Applied and Computational Harmonic Analysis on Graphs and Networks: Tools and Applications

Jeff Irion & Naoki Saito

Department of Mathematics
University of California, Davis

SPIE Conference on Wavelets and Sparsity XVI
San Diego, CA
August 12, 2015

Outline

Acknowledgment

- Support from Office of Naval Research grant: ONR N00014-12-1-0177
- Support from National Science Foundation grant: DMS-1418779
- Support for Jeff Irion from National Defense Science and Engineering Graduate Fellowship, 32 CFR 168a via AFOSR FA9550-11-C-0028
- The Science News dataset provided by Jeff Solka (George Mason Univ.) via Raphy Coifman (Yale) and Matan Gavish (Hebrew Univ.)

Outline

Motivations: Why Graphs and Networks?

- More and more data are collected in a distributed and irregular manner; they are not organized such as familiar digital signals and images sampled on regular lattices. Examples include:
  - Data from sensor networks
  - Data from social networks, webpages, . . .
  - Data from biological networks
  - . . .
- It is quite important to analyze:
  - Topology of graphs/networks (e.g., how nodes are connected, etc.)
  - Data measured on nodes (e.g., a node = a sensor, then what is an edge?)

Motivations: Why Graphs and Networks?

- More and more data are collected in a distributed and irregular manner; they are not organized such as familiar digital signals and images sampled on regular lattices. Examples include:
  - Data from sensor networks
  - Data from social networks, webpages, . . .
  - Data from biological networks
  - . . .
- It is quite important to analyze:
  - Topology of graphs/networks (e.g., how nodes are connected, etc.)
  - Data measured on nodes (e.g., a node = a sensor, then what is an edge?)

Motivations: Why Graphs and Networks?

- More and more data are collected in a distributed and irregular manner; they are not organized such as familiar digital signals and images sampled on regular lattices. Examples include:
  - Data from sensor networks
  - Data from social networks, webpages, ...
  - Data from biological networks
  - ...
- It is quite important to analyze:
  - Topology of graphs/networks (e.g., how nodes are connected, etc.)
  - Data measured on nodes (e.g., a node = a sensor, then what is an edge?)

Motivations: Why Graphs & Networks?

- Fourier analysis/synthesis and wavelet analysis/synthesis have been 'crown jewels' for data sampled on the regular lattices.
- Hence, we need to *lift* such tools for unorganized and irregularly-sampled datasets including those represented by graphs and networks.
- Moreover, constructing a graph from a usual signal or image and analyzing it can also be very useful! E.g., Nonlocal means image denoising of Buades-Coll-Morel[1]; Signal Segmentation as we will see later.

[1]A. Buades, B. Coll, J.-M. Morel, "A review of image denoising algorithms, with a new one," *Multiscale Model. Simul.*, vol. 4 (2), pp. 490–530, 2005.

Motivations: Why Graphs & Networks?

- Fourier analysis/synthesis and wavelet analysis/synthesis have been 'crown jewels' for data sampled on the regular lattices.
- Hence, we need to *lift* such tools for unorganized and irregularly-sampled datasets including those represented by graphs and networks.
- Moreover, constructing a graph from a usual signal or image and analyzing it can also be very useful! E.g., Nonlocal means image denoising of Buades-Coll-Morel[1]; Signal Segmentation as we will see later.

[1] A. Buades, B. Coll, J.-M. Morel, "A review of image denoising algorithms, with a new one," *Multiscale Model. Simul.*, vol. 4 (2), pp. 490–530, 2005.

Motivations: Why Graphs & Networks?

- Fourier analysis/synthesis and wavelet analysis/synthesis have been 'crown jewels' for data sampled on the regular lattices.
- Hence, we need to *lift* such tools for unorganized and irregularly-sampled datasets including those represented by graphs and networks.
- Moreover, constructing a graph from a usual signal or image and analyzing it can also be very useful! E.g., Nonlocal means image denoising of Buades-Coll-Morel[1]; Signal Segmentation as we will see later.

[1] A. Buades, B. Coll, J.-M. Morel, "A review of image denoising algorithms, 315 with a new one," *Multiscale Model. Simul.*, vol. 4 (2), pp. 490–530, 2005.

Today's Goals

- Briefly review some basic concepts and terminology of graph theory and *graph Laplacians*
- Introduce the tools we recently developed:
  - Hierarchical Graph Laplacian Eigen Transform (HGLET) ≈ *Hierarchical Block Discrete Cosine Transforms on graphs*;
  - Generalized Haar-Walsh Transform (GHWT) = *Haar-Walsh Wavelet Packet Dictionary for graphs*
- Present some interesting applications using them: *noise removal* (or *denoising*); *signal segmentation*; *matrix data analysis*

Today's Goals

- Briefly review some basic concepts and terminology of graph theory and *graph Laplacians*
- Introduce the tools we recently developed:
  - Hierarchical Graph Laplacian Eigen Transform (HGLET) $\approx$ *Hierarchical Block Discrete Cosine Transforms on graphs*;
  - Generalized Haar-Walsh Transform (GHWT) $=$ *Haar-Walsh Wavelet Packet Dictionary for graphs*
- Present some interesting applications using them: *noise removal* (or *denoising*); *signal segmentation*; *matrix data analysis*

Today's Goals

- Briefly review some basic concepts and terminology of graph theory and *graph Laplacians*
- Introduce the tools we recently developed:
  - Hierarchical Graph Laplacian Eigen Transform (HGLET) ≈ *Hierarchical Block Discrete Cosine Transforms on graphs*;
  - Generalized Haar-Walsh Transform (GHWT) = *Haar-Walsh Wavelet Packet Dictionary for graphs*
- Present some interesting applications using them: *noise removal* (or *denoising*); *signal segmentation*; *matrix data analysis*

Today's Goals

- Briefly review some basic concepts and terminology of graph theory and *graph Laplacians*
- Introduce the tools we recently developed:
  - Hierarchical Graph Laplacian Eigen Transform (HGLET) $\approx$ *Hierarchical Block Discrete Cosine Transforms on graphs*;
  - Generalized Haar-Walsh Transform (GHWT) $=$ *Haar-Walsh Wavelet Packet Dictionary for graphs*
- Present some interesting applications using them: *noise removal* (or *denoising*); *signal segmentation*; *matrix data analysis*

Today's Goals

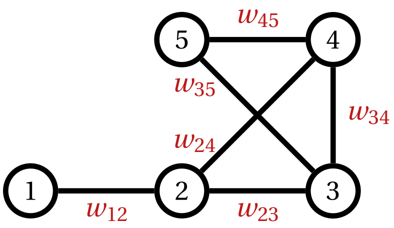
- Briefly review some basic concepts and terminology of graph theory and *graph Laplacians*
- Introduce the tools we recently developed:
    - Hierarchical Graph Laplacian Eigen Transform (HGLET) $\approx$ *Hierarchical Block Discrete Cosine Transforms on graphs*;
    - Generalized Haar-Walsh Transform (GHWT) $=$ *Haar-Walsh Wavelet Packet Dictionary for graphs*
- Present some interesting applications using them: *noise removal* (or *denoising*); *signal segmentation*; *matrix data analysis*

Outline

Definitions and Notation

Let G be a graph.

- $V = V(G) = \{v_1, \dots, v_n\}$ is the set of vertices.
- For simplicity, we often use $1, \dots, n$ instead of $v_1, \dots, v_n$.
- $E = E(G) = \{e_1, \dots, e_m\}$ is the set of edges, where $e_k = (i, j)$ represents an edge (or line segment) connecting between adjacent vertices i, j for some $1 \le i, j \le n$.
- $W = W(G) \in \mathbb{R}^{n \times n}$ is the weight matrix, where w_{ij} denotes the edge weight between vertices i and j.



Definitions and Notation . . .

Note that there are many ways to define w_{ij}.

For example, for *unweighted* graphs, we typically use

$$w_{ij} := \begin{cases} 1 & \text{if } i \sim j \text{ (i.e., } i \text{ and } j \text{ are adjacent);} \\ 0 & \text{otherwise.} \end{cases}$$

This is often referred to as the adjacency matrix and denoted by $A(G)$.

For *weighted* graphs, w_{ij} should reflect the similarity (or affinity) of information at i and j, e.g., if $i \sim j$, then

$$w_{ij} := 1/\mathrm{dist}(i, j) \quad \text{or} \quad \exp(-\mathrm{dist}(i, j)^2/\epsilon^2),$$

where $\mathrm{dist}(\cdot, \cdot)$ is a certain measure of dissimilarity and $\epsilon > 0$ is an appropriate scale parameter.

Definitions and Notation . . .

Note that there are many ways to define w_{ij}.

For example, for *unweighted* graphs, we typically use

$$w_{ij} := \begin{cases} 1 & \text{if } i \sim j \text{ (i.e., } i \text{ and } j \text{ are adjacent);} \\ 0 & \text{otherwise.} \end{cases}$$

This is often referred to as the adjacency matrix and denoted by $A(G)$.

For *weighted* graphs, w_{ij} should reflect the similarity (or affinity) of information at i and j, e.g., if $i \sim j$, then

$$w_{ij} := 1/\mathrm{dist}(i,j) \quad \text{or} \quad \exp(-\mathrm{dist}(i,j)^2/\epsilon^2),$$

where $\mathrm{dist}(\cdot,\cdot)$ is a certain measure of dissimilarity and $\epsilon > 0$ is an appropriate scale parameter.

Our Assumptions

In this talk, we assume that the graph is

- **connected.** Otherwise, we would simply consider the components separately.
- **undirected.** Edges do not have direction, which means that $w_{ij} = w_{ji}$ and thus W is *symmetric*.

The graph may be weighted or unweighted.

Matrices Associated with a Graph

- Let $D = D(G) := \operatorname{diag}(d_1, \ldots, d_n)$ be the degree matrix of G where $d_i := \sum_{j=1}^{n} w_{ij}$ is the degree of the vertex i.
- We can now define several Laplacian matrices of G:

$L(G) := D - W$	Unnormalized
$L_{\mathrm{rw}}(G) := I_n - D^{-1}W = D^{-1}L$	Random-Walk Normalized
$L_{\mathrm{sym}}(G) := I_n - D^{-\frac{1}{2}}WD^{-\frac{1}{2}} = D^{-\frac{1}{2}}LD^{-\frac{1}{2}}$	Symmetrically-Normalized

- Graph Laplacians can also be defined for directed graphs; However, there are many different definitions based on the types/classes of directed graphs, and in general, those matrices are *nonsymmetric*. See, e.g., Fan Chung: "Laplacians and the Cheeger inequality for directed graphs," *Ann. Comb.*, vol. 9, no. 1, pp. 1–19, 2005, for an attempt to symmetrize graph Laplacian matrices for *strongly connected* digraphs.

Matrices Associated with a Graph

- Let $D = D(G) := \operatorname{diag}(d_1, \dots, d_n)$ be the degree matrix of G where $d_i := \sum_{j=1}^{n} w_{ij}$ is the degree of the vertex i.
- We can now define several Laplacian matrices of G:

$$L(G) := D - W \qquad \text{Unnormalized}$$

$$L_{\text{rw}}(G) := I_n - D^{-1}W = D^{-1}L \qquad \text{Random-Walk Normalized}$$

$$L_{\text{sym}}(G) := I_n - D^{-\frac{1}{2}}WD^{-\frac{1}{2}} = D^{-\frac{1}{2}}LD^{-\frac{1}{2}} \qquad \text{Symmetrically-Normalized}$$

- Graph Laplacians can also be defined for directed graphs; However, there are many different definitions based on the types/classes of directed graphs, and in general, those matrices are *nonsymmetric*. See, e.g., Fan Chung: "Laplacians and the Cheeger inequality for directed graphs," *Ann. Comb.*, vol. 9, no. 1, pp. 1–19, 2005, for an attempt to symmetrize graph Laplacian matrices for *strongly connected* digraphs.

Matrices Associated with a Graph

- Let $D = D(G) := \mathrm{diag}(d_1, \ldots, d_n)$ be the degree matrix of G where $d_i := \sum_{j=1}^{n} w_{ij}$ is the degree of the vertex i.
- We can now define several Laplacian matrices of G:

$$L(G) := D - W \qquad \text{Unnormalized}$$

$$L_{\mathrm{rw}}(G) := I_n - D^{-1}W = D^{-1}L \qquad \text{Random-Walk Normalized}$$

$$L_{\mathrm{sym}}(G) := I_n - D^{-\frac{1}{2}}WD^{-\frac{1}{2}} = D^{-\frac{1}{2}}LD^{-\frac{1}{2}} \qquad \text{Symmetrically-Normalized}$$

- Graph Laplacians can also be defined for directed graphs; However, there are many different definitions based on the types/classes of directed graphs, and in general, those matrices are *nonsymmetric*. See, e.g., Fan Chung: "Laplacians and the Cheeger inequality for directed graphs," *Ann. Comb.*, vol. 9, no. 1, pp. 1–19, 2005, for an attempt to symmetrize graph Laplacian matrices for *strongly connected* digraphs.

Graph Laplacians . . .

- Let $f \in \mathbb{R}^n$ be a data vector defined on $V(G)$. Then

$$Lf(i) = d_i f(i) - \sum_{j=1}^{n} w_{ij} f(j) = \sum_{j=1}^{n} w_{ij} \left(f(i) - f(j) \right).$$

  i.e., this is a generalization of the *finite difference approximation* to the Laplace operator.

- On the other hand,

$$L_{\mathrm{rw}} f(i) = f(i) - \sum_{j=1}^{n} p_{ij} f(j) = \frac{1}{d_i} \sum_{j=1}^{n} w_{ij} \left(f(i) - f(j) \right).$$

$$L_{\mathrm{sym}} f(i) = f(i) - \frac{1}{\sqrt{d_i}} \sum_{j=1}^{n} \frac{w_{ij}}{\sqrt{d_j}} f(j) = \frac{1}{\sqrt{d_i}} \sum_{j=1}^{n} w_{ij} \left(\frac{f(i)}{\sqrt{d_i}} - \frac{f(j)}{\sqrt{d_j}} \right).$$

- Note that these definitions of the graph Laplacian corresponds to $-\Delta$ in $\mathbb{R}^d$, i.e., they are nonnegative operators (a.k.a. positive semi-definite matrices).

Graph Laplacians . . .

- Let $f \in \mathbb{R}^n$ be a data vector defined on $V(G)$. Then

$$Lf(i) = d_i f(i) - \sum_{j=1}^{n} w_{ij} f(j) = \sum_{j=1}^{n} w_{ij} \left(f(i) - f(j) \right).$$

  i.e., this is a generalization of the *finite difference approximation* to the Laplace operator.

- On the other hand,

$$L_{\mathrm{rw}} f(i) = f(i) - \sum_{j=1}^{n} p_{ij} f(j) = \frac{1}{d_i} \sum_{j=1}^{n} w_{ij} \left(f(i) - f(j) \right).$$

$$L_{\mathrm{sym}} f(i) = f(i) - \frac{1}{\sqrt{d_i}} \sum_{j=1}^{n} \frac{w_{ij}}{\sqrt{d_j}} f(j) = \frac{1}{\sqrt{d_i}} \sum_{j=1}^{n} w_{ij} \left(\frac{f(i)}{\sqrt{d_i}} - \frac{f(j)}{\sqrt{d_j}} \right).$$

- Note that these definitions of the graph Laplacian corresponds to $-\Delta$ in $\mathbb{R}^d$, i.e., they are nonnegative operators (a.k.a. positive semi-definite matrices).

Graph Laplacians . . .

- Let $f \in \mathbb{R}^n$ be a data vector defined on $V(G)$. Then

$$Lf(i) = d_i f(i) - \sum_{j=1}^{n} w_{ij} f(j) = \sum_{j=1}^{n} w_{ij} \left(f(i) - f(j) \right).$$

  i.e., this is a generalization of the *finite difference approximation* to the Laplace operator.

- On the other hand,

$$L_{\mathrm{rw}} f(i) = f(i) - \sum_{j=1}^{n} p_{ij} f(j) = \frac{1}{d_i} \sum_{j=1}^{n} w_{ij} \left(f(i) - f(j) \right).$$

$$L_{\mathrm{sym}} f(i) = f(i) - \frac{1}{\sqrt{d_i}} \sum_{j=1}^{n} \frac{w_{ij}}{\sqrt{d_j}} f(j) = \frac{1}{\sqrt{d_i}} \sum_{j=1}^{n} w_{ij} \left(\frac{f(i)}{\sqrt{d_i}} - \frac{f(j)}{\sqrt{d_j}} \right).$$

- Note that these definitions of the graph Laplacian corresponds to $-\Delta$ in $\mathbb{R}^d$, i.e., they are nonnegative operators (a.k.a. positive semi-definite matrices).

Why Graph Laplacian Eigenfunctions?

- The graph Laplacian *eigenfunctions* form an orthonormal basis on a graph $\Longrightarrow$
  - can *expand* functions defined on a graph
  - can perform *spectral analysis/synthesis/filtering* of data measured on vertices of a graph
- Can be used for graph partitioning, graph drawing, data analysis, clustering, ... $\Longrightarrow$ Graph Cut, Spectral Clustering
- Less studied than graph Laplacian eigenvalues
- In this talk, I will use the terms "eigenfunctions" and "eigenvectors" interchangeably.
- Also, an eigenvector/function is denoted by $\boldsymbol{\phi}$, and its value at vertex $x \in V$ is denoted by $\boldsymbol{\phi}(x)$.

Why Graph Laplacian Eigenfunctions?

- The graph Laplacian *eigenfunctions* form an orthonormal basis on a graph $\Longrightarrow$
  - can *expand* functions defined on a graph
  - can perform *spectral analysis/synthesis/filtering* of data measured on vertices of a graph
- Can be used for graph partitioning, graph drawing, data analysis, clustering, ... $\Longrightarrow$ Graph Cut, Spectral Clustering
- Less studied than graph Laplacian eigenvalues
- In this talk, I will use the terms "eigenfunctions" and "eigenvectors" interchangeably.
- Also, an eigenvector/function is denoted by ϕ, and its value at vertex $x \in V$ is denoted by $\phi(x)$.

Why Graph Laplacian Eigenfunctions?

- The graph Laplacian *eigenfunctions* form an orthonormal basis on a graph $\Longrightarrow$
  - can *expand* functions defined on a graph
  - can perform *spectral analysis/synthesis/filtering* of data measured on vertices of a graph
- Can be used for graph partitioning, graph drawing, data analysis, clustering, ... $\Longrightarrow$ Graph Cut, Spectral Clustering
- Less studied than graph Laplacian eigenvalues
- In this talk, I will use the terms "eigenfunctions" and "eigenvectors" interchangeably.
- Also, an eigenvector/function is denoted by $\boldsymbol{\phi}$, and its value at vertex $x \in V$ is denoted by $\boldsymbol{\phi}(x)$.

Why Graph Laplacian Eigenfunctions?

- The graph Laplacian *eigenfunctions* form an orthonormal basis on a graph $\Longrightarrow$
  - can *expand* functions defined on a graph
  - can perform *spectral analysis/synthesis/filtering* of data measured on vertices of a graph
- Can be used for graph partitioning, graph drawing, data analysis, clustering, . . . $\Longrightarrow$ Graph Cut, Spectral Clustering
- Less studied than graph Laplacian eigenvalues
- In this talk, I will use the terms "eigenfunctions" and "eigenvectors" interchangeably.
- Also, an eigenvector/function is denoted by $\boldsymbol{\phi}$, and its value at vertex $x \in V$ is denoted by $\boldsymbol{\phi}(x)$.

Why Graph Laplacian Eigenfunctions?

- The graph Laplacian *eigenfunctions* form an orthonormal basis on a graph $\Longrightarrow$
  - can *expand* functions defined on a graph
  - can perform *spectral analysis/synthesis/filtering* of data measured on vertices of a graph
- Can be used for graph partitioning, graph drawing, data analysis, clustering, ... $\Longrightarrow$ Graph Cut, Spectral Clustering
- Less studied than graph Laplacian eigenvalues
- In this talk, I will use the terms "eigenfunctions" and "eigenvectors" interchangeably.
- Also, an eigenvector/function is denoted by ϕ, and its value at vertex $x \in V$ is denoted by $\phi(x)$.

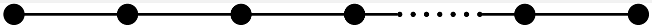
Why Graph Laplacian Eigenfunctions?

- The graph Laplacian *eigenfunctions* form an orthonormal basis on a graph $\Longrightarrow$
  - can *expand* functions defined on a graph
  - can perform *spectral analysis/synthesis/filtering* of data measured on vertices of a graph
- Can be used for graph partitioning, graph drawing, data analysis, clustering, ... $\Longrightarrow$ Graph Cut, Spectral Clustering
- Less studied than graph Laplacian eigenvalues
- In this talk, I will use the terms "eigenfunctions" and "eigenvectors" interchangeably.
- Also, an eigenvector/function is denoted by ϕ, and its value at vertex $x \in V$ is denoted by $\phi(x)$.

Why Graph Laplacian Eigenfunctions?

- The graph Laplacian *eigenfunctions* form an orthonormal basis on a graph $\Longrightarrow$
  - can *expand* functions defined on a graph
  - can perform *spectral analysis/synthesis/filtering* of data measured on vertices of a graph
- Can be used for graph partitioning, graph drawing, data analysis, clustering, . . . $\Longrightarrow$ Graph Cut, Spectral Clustering
- Less studied than graph Laplacian eigenvalues
- In this talk, I will use the terms “eigenfunctions” and “eigenvectors” interchangeably.
- Also, an eigenvector/function is denoted by $\boldsymbol{\phi}$, and its value at vertex $x \in V$ is denoted by $\boldsymbol{\phi}(x)$.

A Simple Yet Important Example: A Path Graph



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

The eigenvectors of this matrix are exactly the DCT Type II basis vectors used for the JPEG image compression standard! (See G. Strang, "The discrete cosine transform," *SIAM Review*, vol. 41, pp. 135–147, 1999).

- $\lambda_k = 4\sin^2(\pi k/2n)$; $\boldsymbol{\phi}_k(\ell) = \sqrt{\frac{2}{n}}\cos\left(\pi k\left(\ell+\frac{1}{2}\right)/n\right)$, $k,\ell = 0,1,\dots,n-1$.
- In this simple case, λ (eigenvalue) is a monotonic function w.r.t. the frequency, which is the eigenvalue index k. However, *the notion of frequency is not well defined on a more general graph!*
- The eigenvectors of $L_{\mathrm{sym}} \equiv D^{1/2}\cdot$ the eigenvectors of L_{rw} $\equiv$ the DCT Type I basis vectors

Outline

A Brief Review of Graph Laplacian Eigenpairs

- In this review part, we only consider undirected graphs and their unnormalized Laplacians $L(G) = D(G) - W(G)$.
- It is a good exercise to see how the statements change for L_{rw}, L_{sym}.
- $L(G)$ is positive semi-definite as was shown earlier. Hence, we can *sort* the eigenvalues of $L(G)$ as $0 = \lambda_0(G) \le \lambda_1(G) \le \cdots \le \lambda_{n-1}(G)$.
- $m_G(\lambda) :=$ the multiplicity of λ.
- $\operatorname{rank} L(G) = n - m_G(0)$ where $m_G(0)$ turns out to be the number of connected components of G. $L(G)$ has $m_G(0)$ diagonal blocks; the eigenspace corresponding to $\lambda = 0$ is spanned by the *indicator* vectors of each connected component.
- In particular, $\lambda_1 \ne 0$, i.e., $m_G(0) = 1$ iff G is connected. Then, the eigenfunction corresponding to $\lambda_0 = 0$ is the constant function $\boldsymbol{\phi}_0 = \mathbf{1}_n$.
- This led M. Fiedler (1973) to define the algebraic connectivity of G by $a(G) := \lambda_1(G)$, viewing it as *a quantitative measure of connectivity*.

A Brief Review of Graph Laplacian Eigenpairs

- In this review part, we only consider undirected graphs and their unnormalized Laplacians $L(G) = D(G) - W(G)$.
- It is a good exercise to see how the statements change for L_{rw}, L_{sym}.
- $L(G)$ is positive semi-definite as was shown earlier. Hence, we can *sort* the eigenvalues of $L(G)$ as $0 = \lambda_0(G) \le \lambda_1(G) \le \cdots \le \lambda_{n-1}(G)$.
- $m_G(\lambda) :=$ the multiplicity of λ.
- $\operatorname{rank} L(G) = n - m_G(0)$ where $m_G(0)$ turns out to be the number of connected components of G. $L(G)$ has $m_G(0)$ diagonal blocks; the eigenspace corresponding to $\lambda = 0$ is spanned by the *indicator* vectors of each connected component.
- In particular, $\lambda_1 \ne 0$, i.e., $m_G(0) = 1$ iff G is connected. Then, the eigenfunction corresponding to $\lambda_0 = 0$ is the constant function $\boldsymbol{\phi}_0 = \mathbf{1}_n$.
- This led M. Fiedler (1973) to define the algebraic connectivity of G by $a(G) := \lambda_1(G)$, viewing it as *a quantitative measure of connectivity*.

A Brief Review of Graph Laplacian Eigenpairs

- In this review part, we only consider undirected graphs and their unnormalized Laplacians $L(G) = D(G) - W(G)$.
- It is a good exercise to see how the statements change for L_{rw}, L_{sym}.
- $L(G)$ is positive semi-definite as was shown earlier. Hence, we can *sort* the eigenvalues of $L(G)$ as $0 = \lambda_0(G) \le \lambda_1(G) \le \cdots \le \lambda_{n-1}(G)$.
- $m_G(\lambda) :=$ the multiplicity of λ.
- $\operatorname{rank} L(G) = n - m_G(0)$ where $m_G(0)$ turns out to be the number of connected components of G. $L(G)$ has $m_G(0)$ diagonal blocks; the eigenspace corresponding to $\lambda = 0$ is spanned by the *indicator* vectors of each connected component.
- In particular, $\lambda_1 \neq 0$, i.e., $m_G(0) = 1$ iff G is connected. Then, the eigenfunction corresponding to $\lambda_0 = 0$ is the constant function $\boldsymbol{\phi}_0 = \mathbf{1}_n$.
- This led M. Fiedler (1973) to define the algebraic connectivity of G by $a(G) := \lambda_1(G)$, viewing it as *a quantitative measure of connectivity*.

A Brief Review of Graph Laplacian Eigenpairs

- In this review part, we only consider undirected graphs and their unnormalized Laplacians $L(G) = D(G) - W(G)$.
- It is a good exercise to see how the statements change for L_{rw}, L_{sym}.
- $L(G)$ is positive semi-definite as was shown earlier. Hence, we can *sort* the eigenvalues of $L(G)$ as $0 = \lambda_0(G) \le \lambda_1(G) \le \cdots \le \lambda_{n-1}(G)$.
- $m_G(\lambda) :=$ the multiplicity of λ.
- $\operatorname{rank} L(G) = n - m_G(0)$ where $m_G(0)$ turns out to be the number of connected components of G. $L(G)$ has $m_G(0)$ diagonal blocks; the eigenspace corresponding to $\lambda = 0$ is spanned by the *indicator* vectors of each connected component.
- In particular, $\lambda_1 \neq 0$, i.e., $m_G(0) = 1$ iff G is connected. Then, the eigenfunction corresponding to $\lambda_0 = 0$ is the constant function $\boldsymbol{\phi}_0 = \mathbf{1}_n$.
- This led M. Fiedler (1973) to define the algebraic connectivity of G by $a(G) := \lambda_1(G)$, viewing it as *a quantitative measure of connectivity*.

A Brief Review of Graph Laplacian Eigenpairs

- In this review part, we only consider undirected graphs and their unnormalized Laplacians $L(G) = D(G) - W(G)$.
- It is a good exercise to see how the statements change for L_{rw}, L_{sym}.
- $L(G)$ is positive semi-definite as was shown earlier. Hence, we can *sort* the eigenvalues of $L(G)$ as $0 = \lambda_0(G) \le \lambda_1(G) \le \cdots \le \lambda_{n-1}(G)$.
- $m_G(\lambda)$:= the multiplicity of λ.
- $\operatorname{rank} L(G) = n - m_G(0)$ where $m_G(0)$ turns out to be the number of connected components of G. $L(G)$ has $m_G(0)$ diagonal blocks; the eigenspace corresponding to $\lambda = 0$ is spanned by the *indicator* vectors of each connected component.
- In particular, $\lambda_1 \neq 0$, i.e., $m_G(0) = 1$ iff G is connected. Then, the eigenfunction corresponding to $\lambda_0 = 0$ is the constant function $\boldsymbol{\phi}_0 = \mathbf{1}_n$.
- This led M. Fiedler (1973) to define the algebraic connectivity of G by $a(G) := \lambda_1(G)$, viewing it as *a quantitative measure of connectivity*.

A Brief Review of Graph Laplacian Eigenpairs

- In this review part, we only consider undirected graphs and their unnormalized Laplacians $L(G) = D(G) - W(G)$.
- It is a good exercise to see how the statements change for L_{rw}, L_{sym}.
- $L(G)$ is positive semi-definite as was shown earlier. Hence, we can *sort* the eigenvalues of $L(G)$ as $0 = \lambda_0(G) \le \lambda_1(G) \le \cdots \le \lambda_{n-1}(G)$.
- $m_G(\lambda)$:= the multiplicity of λ.
- $\operatorname{rank} L(G) = n - m_G(0)$ where $m_G(0)$ turns out to be the number of connected components of G. $L(G)$ has $m_G(0)$ diagonal blocks; the eigenspace corresponding to $\lambda = 0$ is spanned by the *indicator* vectors of each connected component.
- In particular, $\lambda_1 \ne 0$, i.e., $m_G(0) = 1$ iff G is connected. Then, the eigenfunction corresponding to $\lambda_0 = 0$ is the constant function $\boldsymbol{\phi}_0 = \mathbf{1}_n$.
- This led M. Fiedler (1973) to define the algebraic connectivity of G by $a(G) := \lambda_1(G)$, viewing it as *a quantitative measure of connectivity*.

A Brief Review of Graph Laplacian Eigenpairs

- In this review part, we only consider undirected graphs and their unnormalized Laplacians $L(G) = D(G) - W(G)$.
- It is a good exercise to see how the statements change for L_{rw}, L_{sym}.
- $L(G)$ is positive semi-definite as was shown earlier. Hence, we can *sort* the eigenvalues of $L(G)$ as $0 = \lambda_0(G) \le \lambda_1(G) \le \cdots \le \lambda_{n-1}(G)$.
- $m_G(\lambda)$:= the multiplicity of λ.
- $\operatorname{rank} L(G) = n - m_G(0)$ where $m_G(0)$ turns out to be the number of connected components of G. $L(G)$ has $m_G(0)$ diagonal blocks; the eigenspace corresponding to $\lambda = 0$ is spanned by the *indicator* vectors of each connected component.
- In particular, $\lambda_1 \ne 0$, i.e., $m_G(0) = 1$ iff G is connected. Then, the eigenfunction corresponding to $\lambda_0 = 0$ is the constant function $\boldsymbol{\phi}_0 = \mathbf{1}_n$.
- This led M. Fiedler (1973) to define the algebraic connectivity of G by $a(G) := \lambda_1(G)$, viewing it as *a quantitative measure of connectivity*.

Outline

Graph Partitioning via Spectral Clustering

Goal: Split the vertices V into two "good" subsets, X and X^c

Plan: Use the signs of the entries in $\boldsymbol{\phi}_1$ known as the Fiedler vector

Why? Using $\boldsymbol{\phi}_1$ of $L(G)$ to generate X and X^c yields an *approximate* minimizer of the RatioCut function[1]:

$$\mathrm{RatioCut}(X,X^c) := \frac{\mathrm{cut}(X,X^c)}{|X|} + \frac{\mathrm{cut}(X,X^c)}{|X^c|}, \quad \text{where } \mathrm{cut}(X,X^c) := \sum_{\substack{i\in X\\ j\in X^c}} w_{ij}$$

We can also use the signs of $\boldsymbol{\phi}_1$ of L_{rw} (equivalently, L_{sym}) to cut a graph, which yield an *approximate* minimizer of the *Normalized Cut* (or *NCut*) function of Shi and Malik[2]:

$$\mathrm{NCut}(X,X^c) := \frac{\mathrm{cut}(X,X^c)}{\mathrm{vol}(X)} + \frac{\mathrm{cut}(X,X^c)}{\mathrm{vol}(X^c)}, \quad \text{where } \mathrm{vol}(X) := \sum_{i\in X} d_i$$

[1] L. Hagen and A. B. Kahng: "New spectral methods for ratio cut partitioning and clustering," *IEEE Trans. Comput.-Aided Des.*, vol. 11, no. 9, pp. 1074-1085, 1992.

[2] J. Shi & J. Malik: "Normalized cuts and image segmentation", *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, no. 8, pp. 888–905, 2000.

Graph Partitioning via Spectral Clustering

Goal: Split the vertices V into two "good" subsets, X and X^c

Plan: Use the signs of the entries in $\boldsymbol{\phi}_1$ known as the Fiedler vector

Why? Using $\boldsymbol{\phi}_1$ of $L(G)$ to generate X and X^c yields an *approximate* minimizer of the RatioCut function[1]:

$$\mathrm{RatioCut}(X, X^c) := \frac{\mathrm{cut}(X, X^c)}{|X|} + \frac{\mathrm{cut}(X, X^c)}{|X^c|}, \quad \text{where } \mathrm{cut}(X, X^c) := \sum_{\substack{i \in X \\ j \in X^c}} w_{ij}$$

We can also use the signs of $\boldsymbol{\phi}_1$ of L_{rw} (equivalently, L_{sym}) to cut a graph, which yield an *approximate* minimizer of the *Normalized Cut* (or *NCut*) function of Shi and Malik[2]:

$$\mathrm{NCut}(X, X^c) := \frac{\mathrm{cut}(X, X^c)}{\mathrm{vol}(X)} + \frac{\mathrm{cut}(X, X^c)}{\mathrm{vol}(X^c)}, \quad \text{where } \mathrm{vol}(X) := \sum_{i \in X} d_i$$

[1] L. Hagen and A. B. Kahng: "New spectral methods for ratio cut partitioning and clustering," *IEEE Trans. Comput.-Aided Des.*, vol. 11, no. 9, pp. 1074-1085, 1992.

[2] J. Shi & J. Malik: "Normalized cuts and image segmentation", *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, no. 8, pp. 888–905, 2000.

Graph Partitioning via Spectral Clustering

Goal: Split the vertices V into two "good" subsets, X and X^c

Plan: Use the signs of the entries in $\boldsymbol{\phi}_1$ known as the Fiedler vector

Why? Using $\boldsymbol{\phi}_1$ of $L(G)$ to generate X and X^c yields an *approximate* minimizer of the RatioCut function[1]:

$$\mathrm{RatioCut}(X,X^c) := \frac{\mathrm{cut}(X,X^c)}{|X|} + \frac{\mathrm{cut}(X,X^c)}{|X^c|}, \quad \text{where } \mathrm{cut}(X,X^c) := \sum_{\substack{i\in X\\ j\in X^c}} w_{ij}$$

We can also use the signs of $\boldsymbol{\phi}_1$ of L_{rw} (equivalently, L_{sym}) to cut a graph, which yield an *approximate* minimizer of the *Normalized Cut* (or *NCut*) function of Shi and Malik[2]:

$$\mathrm{NCut}(X,X^c) := \frac{\mathrm{cut}(X,X^c)}{\mathrm{vol}(X)} + \frac{\mathrm{cut}(X,X^c)}{\mathrm{vol}(X^c)}, \quad \text{where } \mathrm{vol}(X) := \sum_{i\in X} d_i$$

[1] L. Hagen and A. B. Kahng: "New spectral methods for ratio cut partitioning and clustering," *IEEE Trans. Comput.-Aided Des.*, vol. 11, no. 9, pp. 1074-1085, 1992.

[2] J. Shi & J. Malik: "Normalized cuts and image segmentation", *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, no. 8, pp. 888–905, 2000.

Graph Partitioning via Spectral Clustering

Goal: Split the vertices V into two "good" subsets, X and X^c

Plan: Use the signs of the entries in $\boldsymbol{\phi}_1$ known as the Fiedler vector

Why? Using $\boldsymbol{\phi}_1$ of $L(G)$ to generate X and X^c yields an *approximate* minimizer of the RatioCut function[1]:

$$\mathrm{RatioCut}(X, X^c) := \frac{\mathrm{cut}(X, X^c)}{|X|} + \frac{\mathrm{cut}(X, X^c)}{|X^c|}, \quad \text{where } \mathrm{cut}(X, X^c) := \sum_{\substack{i \in X \\ j \in X^c}} w_{ij}$$

We can also use the signs of $\boldsymbol{\phi}_1$ of L_{rw} (equivalently, L_{sym}) to cut a graph, which yield an *approximate* minimizer of the *Normalized Cut* (or *NCut*) function of Shi and Malik[2]:

$$\mathrm{NCut}(X, X^c) := \frac{\mathrm{cut}(X, X^c)}{\mathrm{vol}(X)} + \frac{\mathrm{cut}(X, X^c)}{\mathrm{vol}(X^c)}, \quad \text{where } \mathrm{vol}(X) := \sum_{i \in X} d_i$$

[1] L. Hagen and A. B. Kahng: "New spectral methods for ratio cut partitioning and clustering," *IEEE Trans. Comput.-Aided Des.*, vol. 11, no. 9, pp. 1074-1085, 1992.

[2] J. Shi & J. Malik: "Normalized cuts and image segmentation", *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, no. 8, pp. 888–905, 2000.

Graph Partitioning via Spectral Clustering

The practice of using the Fiedler vector to partition a graph is supported by the following theory.

Definition (Weak Nodal Domain)

A positive (or negative) weak nodal domain of f on $V(G)$ is a maximal connected induced subgraph of G on vertices $v \in V$ with $f(v) \geq 0$ (or $f(v) \leq 0$) that contains at least one nonzero vertex. The number of weak nodal domains of f is denoted by $\mathfrak{W}(f)$.

Corollary (Fiedler (1975))

If G is connected, then $\mathfrak{W}(\phi_1) = 2$.

Graph Partitioning via Spectral Clustering

The practice of using the Fiedler vector to partition a graph is supported by the following theory.

Definition (Weak Nodal Domain)

A positive (or negative) weak nodal domain of f on $V(G)$ is a maximal connected induced subgraph of G on vertices $v \in V$ with $f(v) \geq 0$ (or $f(v) \leq 0$) that contains at least one nonzero vertex. The number of weak nodal domains of f is denoted by $\mathfrak{W}(f)$.

Corollary (Fiedler (1975))

If G is connected, then $\mathfrak{W}(\phi_1) = 2$.

Graph Partitioning via Spectral Clustering

The practice of using the Fiedler vector to partition a graph is supported by the following theory.

Definition (Weak Nodal Domain)

A positive (or negative) weak nodal domain of f on $V(G)$ is a maximal connected induced subgraph of G on vertices $v \in V$ with $f(v) \geq 0$ (or $f(v) \leq 0$) that contains at least one nonzero vertex. The number of weak nodal domains of f is denoted by $\mathfrak{W}(f)$.

Corollary (Fiedler (1975))

If G is connected, then $\mathfrak{W}(\phi_1) = 2$.

Example of Graph Partitioning

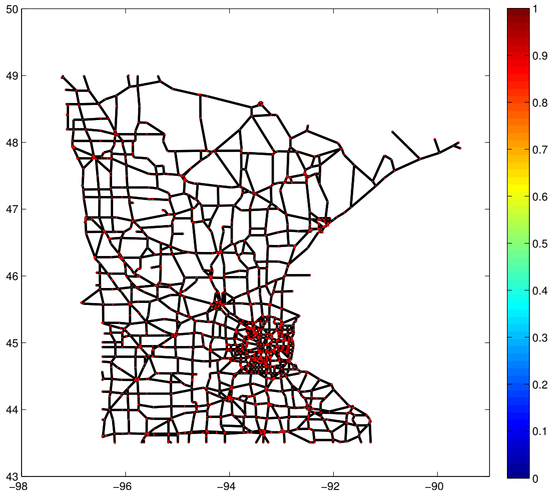


Figure: The MN road network

Example of Graph Partitioning

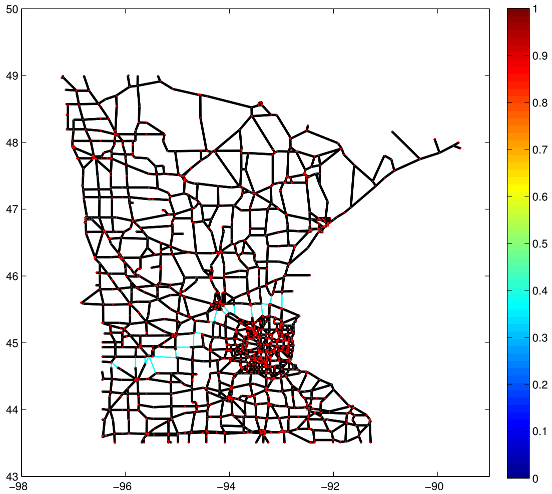
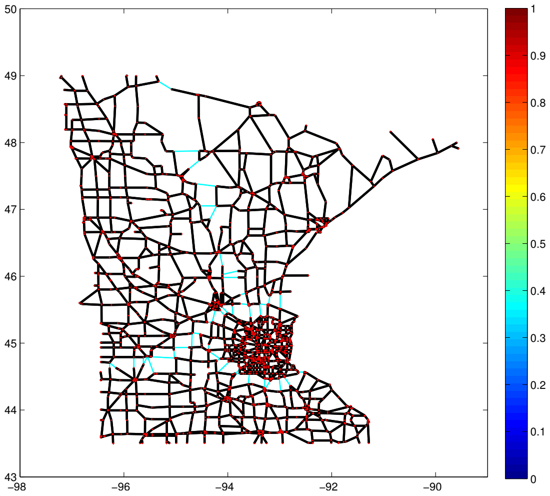


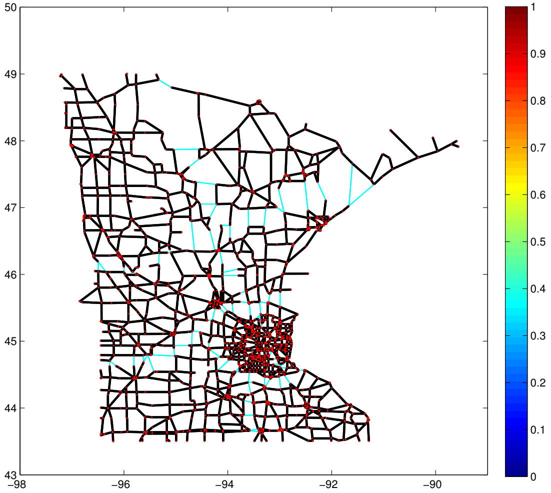
Figure: The MN road network partitioned via the Fiedler vector of L_{rw}

One Can Do This Recursively!



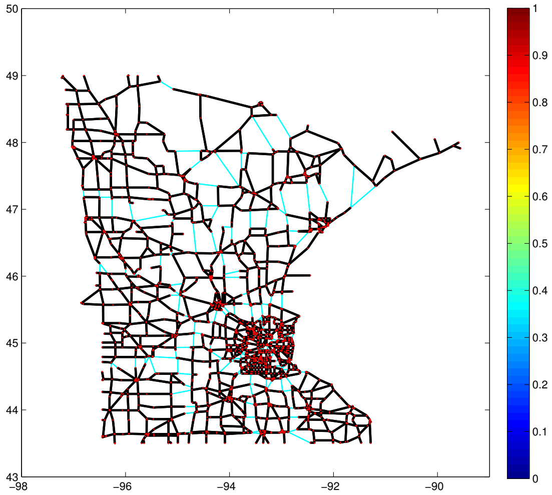
The MN road network recursively partitioned via the Fiedler vectors of L_{rw}'s of subgraphs: $j = 2$

One Can Do This Recursively!



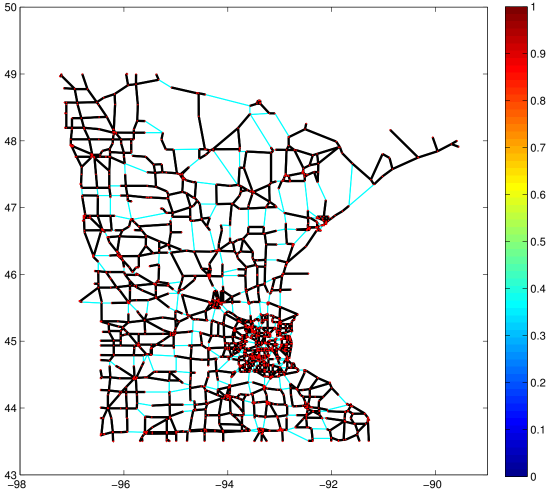
$j = 3$

One Can Do This Recursively!



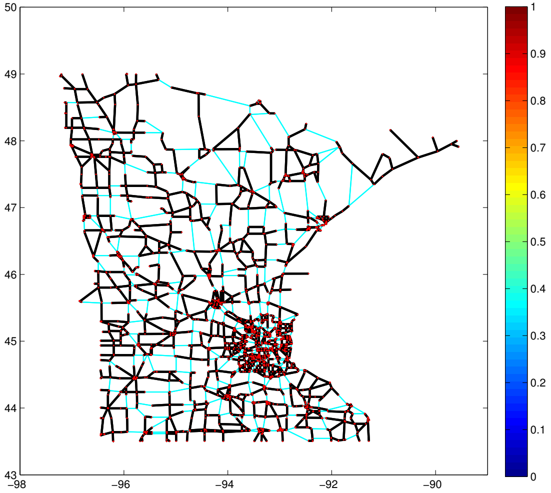
$j = 4$

One Can Do This Recursively!



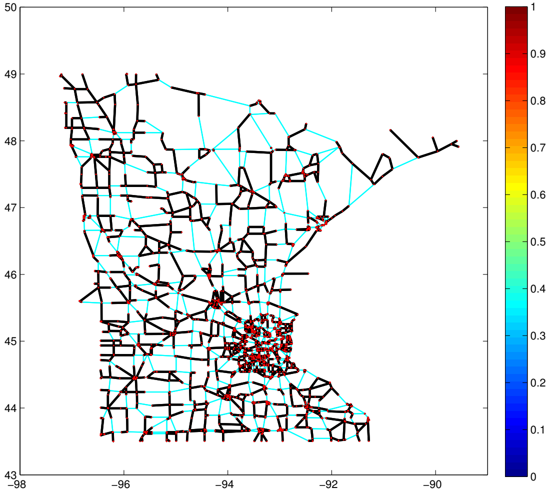
$j = 5$

One Can Do This Recursively!



$j = 6$

One Can Do This Recursively!



$j = 7$

Outline

Motivation: Building Multiscale Basis Dictionaries

- *Wavelets* have been quite successful on regular domains
- They have been extended to irregular domains $\Rightarrow$ "2nd Generation Wavelets" including graphs, e.g.:
  - Coifman and Maggioni (2006): diffusion wavelets; Bremer *et al.* (2006): diffusion wavelet *packets*
  - Jansen, Nason, and Silverman (2008): Adaptation of the *lifting scheme* to graphs
  - Hammond, Vandergheynst, and Gribonval (2011): Spectral graph wavelet transforms (via spectral graph theory)
  - Sharon and Shkolnisky (2015): Laplacian multiwavelet bases (via a combination of spectral graph theory and multiresolution analysis)
  - ...

Motivation: Building Multiscale Basis Dictionaries

- *Wavelets* have been quite successful on regular domains
- They have been extended to irregular domains ⇒ "2nd Generation Wavelets" including graphs, e.g.:
  - Coifman and Maggioni (2006): diffusion wavelets; Bremer *et al.* (2006): diffusion wavelet *packets*
  - Jansen, Nason, and Silverman (2008): Adaptation of the *lifting scheme* to graphs
  - Hammond, Vandergheynst, and Gribonval (2011): Spectral graph wavelet transforms (via spectral graph theory)
  - Sharon and Shkolnisky (2015): Laplacian multiwavelet bases (via a combination of spectral graph theory and multiresolution analysis)
  - . . .

Key Difficulties to Build Wavelets/Wavelet Packets on Graphs

- It has been quite popular to use graph Laplacian eigenvectors as "cosines" or Fourier modes on graphs with eigenvalues as (the square of) their "frequencies"
- However, the notion of *frequency* is ill-defined on general graphs and the Fourier transform is not properly defined on graphs
- Graph Laplacian eigenvectors may also exhibit peculiar behaviors depending on *topology* and *structure* of given graphs!
- For example, eigenvectors corresponding to high eigenvalues may be highly *localized*; see: Y. Nakatsukasa, N. Saito, & E. Woei: "Mysteries around graph Laplacian eigenvalue 4," *Linear Algebra and its Applications*, vol. 438, no. 8, pp. 3231–3246, 2013.
- Hence, building wavelets on graphs based on *the Littlewood-Paley theory* is quite challenging
- Moreover, the notion of *smoothness class* of functions (e.g., Sobolev and Besov spaces) is also difficult to define on graphs $\Longrightarrow$ Can define it on *metric/quantum graphs* . . . *Spaces of homogeneous type*?

Key Difficulties to Build Wavelets/Wavelet Packets on Graphs

- It has been quite popular to use graph Laplacian eigenvectors as "cosines" or Fourier modes on graphs with eigenvalues as (the square of) their "frequencies"
- However, the notion of *frequency* is ill-defined on general graphs and the Fourier transform is not properly defined on graphs
- Graph Laplacian eigenvectors may also exhibit peculiar behaviors depending on *topology* and *structure* of given graphs!
- For example, eigenvectors corresponding to high eigenvalues may be highly *localized*; see: Y. Nakatsukasa, N. Saito, & E. Woei: "Mysteries around graph Laplacian eigenvalue 4," *Linear Algebra and its Applications*, vol. 438, no. 8, pp. 3231–3246, 2013.
- Hence, building wavelets on graphs based on *the Littlewood-Paley theory* is quite challenging
- Moreover, the notion of *smoothness class* of functions (e.g., Sobolev and Besov spaces) is also difficult to define on graphs $\Longrightarrow$ Can define it on *metric/quantum graphs* . . . *Spaces of homogeneous type*?

Key Difficulties to Build Wavelets/Wavelet Packets on Graphs

- It has been quite popular to use graph Laplacian eigenvectors as "cosines" or Fourier modes on graphs with eigenvalues as (the square of) their "frequencies"
- However, the notion of *frequency* is ill-defined on general graphs and the Fourier transform is not properly defined on graphs
- Graph Laplacian eigenvectors may also exhibit peculiar behaviors depending on *topology* and *structure* of given graphs!
- For example, eigenvectors corresponding to high eigenvalues may be highly *localized*; see: Y. Nakatsukasa, N. Saito, & E. Woei: "Mysteries around graph Laplacian eigenvalue 4," *Linear Algebra and its Applications*, vol. 438, no. 8, pp. 3231–3246, 2013.
- Hence, building wavelets on graphs based on *the Littlewood-Paley theory* is quite challenging
- Moreover, the notion of *smoothness class* of functions (e.g., Sobolev and Besov spaces) is also difficult to define on graphs $\Longrightarrow$ Can define it on *metric/quantum graphs* . . . *Spaces of homogeneous type*?

Key Difficulties to Build Wavelets/Wavelet Packets on Graphs

- It has been quite popular to use graph Laplacian eigenvectors as "cosines" or Fourier modes on graphs with eigenvalues as (the square of) their "frequencies"
- However, the notion of *frequency* is ill-defined on general graphs and the Fourier transform is not properly defined on graphs
- Graph Laplacian eigenvectors may also exhibit peculiar behaviors depending on *topology* and *structure* of given graphs!
- For example, eigenvectors corresponding to high eigenvalues may be highly *localized*; see: Y. Nakatsukasa, N. Saito, & E. Woei: "Mysteries around graph Laplacian eigenvalue 4," *Linear Algebra and its Applications*, vol. 438, no. 8, pp. 3231–3246, 2013.
- Hence, building wavelets on graphs based on *the Littlewood-Paley theory* is quite challenging
- Moreover, the notion of *smoothness class* of functions (e.g., Sobolev and Besov spaces) is also difficult to define on graphs $\Longrightarrow$ Can define it on *metric/quantum graphs* . . . *Spaces of homogeneous type*?

Key Difficulties to Build Wavelets/Wavelet Packets on Graphs

- It has been quite popular to use graph Laplacian eigenvectors as "cosines" or Fourier modes on graphs with eigenvalues as (the square of) their "frequencies"
- However, the notion of *frequency* is ill-defined on general graphs and the Fourier transform is not properly defined on graphs
- Graph Laplacian eigenvectors may also exhibit peculiar behaviors depending on *topology* and *structure* of given graphs!
- For example, eigenvectors corresponding to high eigenvalues may be highly *localized*; see: Y. Nakatsukasa, N. Saito, & E. Woei: "Mysteries around graph Laplacian eigenvalue 4," *Linear Algebra and its Applications*, vol. 438, no. 8, pp. 3231–3246, 2013.
- Hence, building wavelets on graphs based on *the Littlewood-Paley theory* is quite challenging
- Moreover, the notion of *smoothness class* of functions (e.g., Sobolev and Besov spaces) is also difficult to define on graphs $\Longrightarrow$ Can define it on *metric/quantum graphs* . . . *Spaces of homogeneous type*?

Key Difficulties to Build Wavelets/Wavelet Packets on Graphs

- It has been quite popular to use graph Laplacian eigenvectors as “cosines” or Fourier modes on graphs with eigenvalues as (the square of) their “frequencies”
- However, the notion of *frequency* is ill-defined on general graphs and the Fourier transform is not properly defined on graphs
- Graph Laplacian eigenvectors may also exhibit peculiar behaviors depending on *topology* and *structure* of given graphs!
- For example, eigenvectors corresponding to high eigenvalues may be highly *localized*; see: Y. Nakatsukasa, N. Saito, & E. Woei: “Mysteries around graph Laplacian eigenvalue 4,” *Linear Algebra and its Applications*, vol. 438, no. 8, pp. 3231–3246, 2013.
- Hence, building wavelets on graphs based on *the Littlewood-Paley theory* is quite challenging
- Moreover, the notion of *smoothness class* of functions (e.g., Sobolev and Besov spaces) is also difficult to define on graphs $\Longrightarrow$ Can define it on *metric/quantum graphs* . . . *Spaces of homogeneous type*?

Our transforms involve 2 main steps:

1. Recursively partition the graph

$\Updownarrow$ These steps can be performed concurrently, or we can fully partition the graph and then generate a set of bases

2. Using the regions on each level of the graph partitioning, generate a set of orthonormal bases for the graph

Our transforms involve 2 main steps:

1. Recursively partition the graph

⇕ These steps can be performed concurrently, or we can fully partition the graph and then generate a set of bases

2. Using the regions on each level of the graph partitioning, generate a set of orthonormal bases for the graph

Outline

Hierarchical Graph Laplacian Eigen Transform (HGLET)

Now we present a novel transform that can be viewed as a generalization of the *block Discrete Cosine Transform*. We refer to this transform as the *Hierarchical Graph Laplacian Eigen Transform (HGLET)*.

The algorithm proceeds as follows...

1. Generate an orthonormal basis for the entire graph ⇒ Laplacian eigenvectors (Notation is $\boldsymbol{\phi}^j_{k,l}$ with $j = 0$)
2. Partition the graph using the Fiedler vector $\boldsymbol{\phi}^j_{k,1}$
3. Generate an orthonormal basis for each of the partitions ⇒ Laplacian eigenvectors
4. Repeat...

$$\begin{bmatrix} \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,n^0_0-1} \end{bmatrix}$$

1. Generate an orthonormal basis for the entire graph ⇒ Laplacian eigenvectors (Notation is $\boldsymbol{\phi}^j_{k,l}$ with $j = 0$)
2. Partition the graph using the Fiedler vector $\boldsymbol{\phi}^j_{k,1}$
3. Generate an orthonormal basis for each of the partitions ⇒ Laplacian eigenvectors
4. Repeat...

$$\begin{bmatrix} \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,n^0_0-1} \end{bmatrix}$$

1. Generate an orthonormal basis for the entire graph ⇒ Laplacian eigenvectors (Notation is $\boldsymbol{\phi}^j_{k,l}$ with $j = 0$)
2. Partition the graph using the Fiedler vector $\boldsymbol{\phi}^j_{k,1}$
3. Generate an orthonormal basis for each of the partitions ⇒ Laplacian eigenvectors
4. Repeat…

$$\begin{bmatrix} \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,n^0_0-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,n^1_0-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^1_{1,0} & \boldsymbol{\phi}^1_{1,1} & \boldsymbol{\phi}^1_{1,2} & \cdots & \boldsymbol{\phi}^1_{1,n^1_1-1} \end{bmatrix}$$

1. Generate an orthonormal basis for the entire graph $\Rightarrow$ Laplacian eigenvectors (Notation is $\boldsymbol{\phi}_{k,l}^{j}$ with $j=0$)
2. Partition the graph using the Fiedler vector $\boldsymbol{\phi}_{k,1}^{j}$
3. Generate an orthonormal basis for each of the partitions $\Rightarrow$ Laplacian eigenvectors
4. Repeat...

$$\begin{bmatrix} \boldsymbol{\phi}_{0,0}^{0} & \boldsymbol{\phi}_{0,1}^{0} & \boldsymbol{\phi}_{0,2}^{0} & \cdots & \boldsymbol{\phi}_{0,n_0^0-1}^{0} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}_{0,0}^{1} & \boldsymbol{\phi}_{0,1}^{1} & \boldsymbol{\phi}_{0,2}^{1} & \cdots & \boldsymbol{\phi}_{0,n_0^1-1}^{1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}_{1,0}^{1} & \boldsymbol{\phi}_{1,1}^{1} & \boldsymbol{\phi}_{1,2}^{1} & \cdots & \boldsymbol{\phi}_{1,n_1^1-1}^{1} \end{bmatrix}$$

1. Generate an orthonormal basis for the entire graph ⇒ Laplacian eigenvectors (Notation is $\boldsymbol{\phi}^{j}_{k,l}$ with $j=0$)
2. Partition the graph using the Fiedler vector $\boldsymbol{\phi}^{j}_{k,1}$
3. Generate an orthonormal basis for each of the partitions ⇒ Laplacian eigenvectors
4. Repeat...

$$\begin{bmatrix} \boldsymbol{\phi}^{0}_{0,0} & \boldsymbol{\phi}^{0}_{0,1} & \boldsymbol{\phi}^{0}_{0,2} & \cdots & \boldsymbol{\phi}^{0}_{0,n^0_0-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^{1}_{0,0} & \boldsymbol{\phi}^{1}_{0,1} & \boldsymbol{\phi}^{1}_{0,2} & \cdots & \boldsymbol{\phi}^{1}_{0,n^1_0-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^{1}_{1,0} & \boldsymbol{\phi}^{1}_{1,1} & \boldsymbol{\phi}^{1}_{1,2} & \cdots & \boldsymbol{\phi}^{1}_{1,n^1_1-1} \end{bmatrix}$$

1. Generate an orthonormal basis for the entire graph $\Rightarrow$ Laplacian eigenvectors (Notation is $\boldsymbol{\phi}^{j}_{k,l}$ with $j = 0$)
2. Partition the graph using the Fiedler vector $\boldsymbol{\phi}^{j}_{k,1}$
3. Generate an orthonormal basis for each of the partitions $\Rightarrow$ Laplacian eigenvectors
4. Repeat...

$$\begin{bmatrix} \boldsymbol{\phi}^{0}_{0,0} & \boldsymbol{\phi}^{0}_{0,1} & \boldsymbol{\phi}^{0}_{0,2} & \cdots & \boldsymbol{\phi}^{0}_{0,n^0_0-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^{1}_{0,0} & \boldsymbol{\phi}^{1}_{0,1} & \boldsymbol{\phi}^{1}_{0,2} & \cdots & \boldsymbol{\phi}^{1}_{0,n^1_0-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^{1}_{1,0} & \boldsymbol{\phi}^{1}_{1,1} & \boldsymbol{\phi}^{1}_{1,2} & \cdots & \boldsymbol{\phi}^{1}_{1,n^1_1-1} \end{bmatrix}$$

$$\left[\boldsymbol{\phi}^{2}_{0,0}\,\boldsymbol{\phi}^{2}_{0,1}\cdots\boldsymbol{\phi}^{2}_{0,n^2_0-1}\right]\left[\boldsymbol{\phi}^{2}_{1,0}\,\boldsymbol{\phi}^{2}_{1,1}\cdots\boldsymbol{\phi}^{2}_{1,n^2_1-1}\right]\left[\boldsymbol{\phi}^{2}_{2,0}\,\boldsymbol{\phi}^{2}_{2,1}\cdots\boldsymbol{\phi}^{2}_{2,n^2_2-1}\right]\left[\boldsymbol{\phi}^{2}_{3,0}\,\boldsymbol{\phi}^{2}_{3,1}\cdots\boldsymbol{\phi}^{2}_{3,n^2_3-1}\right]$$

1. Generate an orthonormal basis for the entire graph ⇒ Laplacian eigenvectors (Notation is $\boldsymbol{\phi}^j_{k,l}$ with $j=0$)
2. Partition the graph using the Fiedler vector $\boldsymbol{\phi}^j_{k,1}$
3. Generate an orthonormal basis for each of the partitions ⇒ Laplacian eigenvectors
4. Repeat...

$$\begin{bmatrix} \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,n^0_0-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,n^1_0-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^1_{1,0} & \boldsymbol{\phi}^1_{1,1} & \boldsymbol{\phi}^1_{1,2} & \cdots & \boldsymbol{\phi}^1_{1,n^1_1-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^2_{0,0}\,\boldsymbol{\phi}^2_{0,1}\cdots\boldsymbol{\phi}^2_{0,n^2_0-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{1,0}\,\boldsymbol{\phi}^2_{1,1}\cdots\boldsymbol{\phi}^2_{1,n^2_1-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{2,0}\,\boldsymbol{\phi}^2_{2,1}\cdots\boldsymbol{\phi}^2_{2,n^2_2-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{3,0}\,\boldsymbol{\phi}^2_{3,1}\cdots\boldsymbol{\phi}^2_{3,n^2_3-1} \end{bmatrix}$$

1. Generate an orthonormal basis for the entire graph ⇒ Laplacian eigenvectors (Notation is $\boldsymbol{\phi}^{j}_{k,l}$ with $j = 0$)
2. Partition the graph using the Fiedler vector $\boldsymbol{\phi}^{j}_{k,1}$
3. Generate an orthonormal basis for each of the partitions ⇒ Laplacian eigenvectors
4. Repeat...

$$\begin{bmatrix} \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,n^0_0-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,n^1_0-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^1_{1,0} & \boldsymbol{\phi}^1_{1,1} & \boldsymbol{\phi}^1_{1,2} & \cdots & \boldsymbol{\phi}^1_{1,n^1_1-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^2_{0,0}\, \boldsymbol{\phi}^2_{0,1} \cdots \boldsymbol{\phi}^2_{0,n^2_0-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{1,0}\, \boldsymbol{\phi}^2_{1,1} \cdots \boldsymbol{\phi}^2_{1,n^2_1-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{2,0}\, \boldsymbol{\phi}^2_{2,1} \cdots \boldsymbol{\phi}^2_{2,n^2_2-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{3,0}\, \boldsymbol{\phi}^2_{3,1} \cdots \boldsymbol{\phi}^2_{3,n^2_3-1} \end{bmatrix}$$

$$\vdots$$

Remarks

- For an unweighted path graph, this exactly yields a *dictionary of the multiscale BDCT-II*
- Similar to wavelet packet or local cosine dictionaries in that it generates a dictionary of bases (i.e., an *overcomplete system*) from which we can select a particular basis useful for the task at hand $\Rightarrow$ best-basis algorithm, local discriminant basis algorithm, ...
  - A union of bases on disjoint subsets is obviously orthonormal

Remarks

- For an unweighted path graph, this exactly yields a *dictionary of the multiscale BDCT-II*
- Similar to wavelet packet or local cosine dictionaries in that it generates a dictionary of bases (i.e., an *overcomplete system*) from which we can select a particular basis useful for the task at hand $\Rightarrow$ best-basis algorithm, local discriminant basis algorithm, . . .
  - A union of bases on disjoint subsets is obviously orthonormal

Remarks

- For an unweighted path graph, this exactly yields a *dictionary of the multiscale BDCT-II*
- Similar to wavelet packet or local cosine dictionaries in that it generates a dictionary of bases (i.e., an *overcomplete system*) from which we can select a particular basis useful for the task at hand $\Rightarrow$ best-basis algorithm, local discriminant basis algorithm, ...
  - A union of bases on disjoint subsets is obviously orthonormal

Remarks

- For an unweighted path graph, this exactly yields a *dictionary of the multiscale BDCT-II*
- Similar to wavelet packet or local cosine dictionaries in that it generates a dictionary of bases (i.e., an *overcomplete system*) from which we can select a particular basis useful for the task at hand ⇒ best-basis algorithm, local discriminant basis algorithm, ...
  - A union of bases on disjoint subsets is obviously orthonormal

$$\begin{bmatrix} \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,n^0_0-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,n^1_0-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^1_{1,0} & \boldsymbol{\phi}^1_{1,1} & \boldsymbol{\phi}^1_{1,2} & \cdots & \boldsymbol{\phi}^1_{1,n^1_1-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^2_{0,0} & \cdots & \boldsymbol{\phi}^2_{0,n^2_0-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{1,0} & \cdots & \boldsymbol{\phi}^2_{1,n^2_1-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{2,0} & \cdots & \boldsymbol{\phi}^2_{2,n^2_2-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{3,0} & \cdots & \boldsymbol{\phi}^2_{3,n^2_3-1} \end{bmatrix}$$

Remarks

- For an unweighted path graph, this exactly yields a *dictionary of the multiscale BDCT-II*
- Similar to wavelet packet or local cosine dictionaries in that it generates a dictionary of bases (i.e., an *overcomplete system*) from which we can select a particular basis useful for the task at hand $\Rightarrow$ best-basis algorithm, local discriminant basis algorithm, . . .
  - A union of bases on disjoint subsets is obviously orthonormal

$$\left[\begin{array}{ccccc} \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,n^0_0-1} \end{array}\right]$$

$$\left[\boldsymbol{\phi}^1_{0,0} \quad \boldsymbol{\phi}^1_{0,1} \quad \boldsymbol{\phi}^1_{0,2} \quad \cdots \quad \boldsymbol{\phi}^1_{0,n^1_0-1}\right]\left[\boldsymbol{\phi}^1_{1,0} \quad \boldsymbol{\phi}^1_{1,1} \quad \boldsymbol{\phi}^1_{1,2} \quad \cdots \quad \boldsymbol{\phi}^1_{1,n^1_1-1}\right]$$

$$\left[\boldsymbol{\phi}^2_{0,0} \quad \cdots \quad \boldsymbol{\phi}^2_{0,n^2_0-1}\right]\left[\boldsymbol{\phi}^2_{1,0} \quad \cdots \quad \boldsymbol{\phi}^2_{1,n^2_1-1}\right]\left[\boldsymbol{\phi}^2_{2,0} \quad \cdots \quad \boldsymbol{\phi}^2_{2,n^2_2-1}\right]\left[\boldsymbol{\phi}^2_{3,0} \quad \cdots \quad \boldsymbol{\phi}^2_{3,n^2_3-1}\right]$$

Remarks

- For an unweighted path graph, this exactly yields a *dictionary of the multiscale BDCT-II*
- Similar to wavelet packet or local cosine dictionaries in that it generates a dictionary of bases (i.e., an *overcomplete system*) from which we can select a particular basis useful for the task at hand ⇒ best-basis algorithm, local discriminant basis algorithm, . . .
  - A union of bases on disjoint subsets is obviously orthonormal

$$\left[\ \boldsymbol{\phi}^0_{0,0} \quad \boldsymbol{\phi}^0_{0,1} \quad \boldsymbol{\phi}^0_{0,2} \quad \cdots \quad \boldsymbol{\phi}^0_{0,n^0_0-1}\ \right]$$

$$\left[\boldsymbol{\phi}^1_{0,0} \quad \boldsymbol{\phi}^1_{0,1} \quad \boldsymbol{\phi}^1_{0,2} \quad \cdots \quad \boldsymbol{\phi}^1_{0,n^1_0-1}\right]\left[\boldsymbol{\phi}^1_{1,0} \quad \boldsymbol{\phi}^1_{1,1} \quad \boldsymbol{\phi}^1_{1,2} \quad \cdots \quad \boldsymbol{\phi}^1_{1,n^1_1-1}\right]$$

$$\left[\boldsymbol{\phi}^2_{0,0} \ \cdots \ \boldsymbol{\phi}^2_{0,n^2_0-1}\right]\left[\boldsymbol{\phi}^2_{1,0} \ \cdots \ \boldsymbol{\phi}^2_{1,n^2_1-1}\right]\left[\boldsymbol{\phi}^2_{2,0} \ \cdots \ \boldsymbol{\phi}^2_{2,n^2_2-1}\right]\left[\boldsymbol{\phi}^2_{3,0} \ \cdots \ \boldsymbol{\phi}^2_{3,n^2_3-1}\right]$$

Remarks

- For an unweighted path graph, this exactly yields a *dictionary of the multiscale BDCT-II*
- Similar to wavelet packet or local cosine dictionaries in that it generates a dictionary of bases (i.e., an *overcomplete system*) from which we can select a particular basis useful for the task at hand ⇒ best-basis algorithm, local discriminant basis algorithm, . . .
    - A union of bases on disjoint subsets is obviously orthonormal

$$\begin{bmatrix} \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,n^0_0-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,n^1_0-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^1_{1,0} & \boldsymbol{\phi}^1_{1,1} & \boldsymbol{\phi}^1_{1,2} & \cdots & \boldsymbol{\phi}^1_{1,n^1_1-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^2_{0,0} & \cdots & \boldsymbol{\phi}^2_{0,n^2_0-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{1,0} & \cdots & \boldsymbol{\phi}^2_{1,n^2_1-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{2,0} & \cdots & \boldsymbol{\phi}^2_{2,n^2_2-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{3,0} & \cdots & \boldsymbol{\phi}^2_{3,n^2_3-1} \end{bmatrix}$$

Remarks

- For an unweighted path graph, this exactly yields a *dictionary of the multiscale BDCT-II*
- Similar to wavelet packet or local cosine dictionaries in that it generates a dictionary of bases (i.e., an *overcomplete system*) from which we can select a particular basis useful for the task at hand $\Rightarrow$ best-basis algorithm, local discriminant basis algorithm, . . .
  - A union of bases on disjoint subsets is obviously orthonormal

$$\left[\quad \boldsymbol{\phi}^0_{0,0} \qquad \boldsymbol{\phi}^0_{0,1} \qquad \boldsymbol{\phi}^0_{0,2} \qquad \cdots \qquad \boldsymbol{\phi}^0_{0,n^0_0-1} \quad\right]$$

$$\left[\boldsymbol{\phi}^1_{0,0} \quad \boldsymbol{\phi}^1_{0,1} \quad \boldsymbol{\phi}^1_{0,2} \quad \cdots \quad \boldsymbol{\phi}^1_{0,n^1_0-1}\right]\left[\boldsymbol{\phi}^1_{1,0} \quad \boldsymbol{\phi}^1_{1,1} \quad \boldsymbol{\phi}^1_{1,2} \quad \cdots \quad \boldsymbol{\phi}^1_{1,n^1_1-1}\right]$$

$$\left[\boldsymbol{\phi}^2_{0,0} \ \cdots \ \boldsymbol{\phi}^2_{0,n^2_0-1}\right]\left[\boldsymbol{\phi}^2_{1,0} \ \cdots \ \boldsymbol{\phi}^2_{1,n^2_1-1}\right]\left[\boldsymbol{\phi}^2_{2,0} \ \cdots \ \boldsymbol{\phi}^2_{2,n^2_2-1}\right]\left[\boldsymbol{\phi}^2_{3,0} \ \cdots \ \boldsymbol{\phi}^2_{3,n^2_3-1}\right]$$

Remarks

- For an unweighted path graph, this exactly yields a *dictionary of the multiscale BDCT-II*
- Similar to wavelet packet or local cosine dictionaries in that it generates a dictionary of bases (i.e., an *overcomplete system*) from which we can select a particular basis useful for the task at hand ⇒ best-basis algorithm, local discriminant basis algorithm, . . .
    - A union of bases on disjoint subsets is obviously orthonormal

$$\begin{bmatrix} \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,n^0_0-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,n^1_0-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^1_{1,0} & \boldsymbol{\phi}^1_{1,1} & \boldsymbol{\phi}^1_{1,2} & \cdots & \boldsymbol{\phi}^1_{1,n^1_1-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^2_{0,0} & \cdots & \boldsymbol{\phi}^2_{0,n^2_0-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{1,0} & \cdots & \boldsymbol{\phi}^2_{1,n^2_1-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{2,0} & \cdots & \boldsymbol{\phi}^2_{2,n^2_2-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{3,0} & \cdots & \boldsymbol{\phi}^2_{3,n^2_3-1} \end{bmatrix}$$

Outline

Generalized Haar-Walsh Transform (GHWT)

HGLET is a generalization of the block DCT, and it generates basis vectors that are *smooth on their support*.

The Generalized Haar-Walsh Transform (GHWT) is a generalization of the classical Haar and Walsh-Hadamard Transforms, and it generates basis vectors that are *piecewise-constant on their support*.

The algorithm can be summarized as follows...

Generalized Haar-Walsh Transform (GHWT)

HGLET is a generalization of the block DCT, and it generates basis vectors that are *smooth on their support*.

The Generalized Haar-Walsh Transform (GHWT) is a generalization of the classical Haar and Walsh-Hadamard Transforms, and it generates basis vectors that are *piecewise-constant on their support*.

The algorithm can be summarized as follows...

1. Generate a full recursive partitioning of the graph ⇒ Fiedler vectors
2. Generate an orthonormal basis for level $j_{\max}$ (the finest level) ⇒ ***scaling vectors*** on the single-node regions
   - As with HGLET, the notation is $\boldsymbol{\psi}^j_{k,l}$
3. Using the basis for level $j_{\max}$, generate an orthonormal basis for level $j_{\max}-1$ ⇒ ***scaling*** and ***Haar-like*** vectors
4. Repeat... Using the basis for level j, generate an orthonormal basis for level $j-1$ ⇒ ***scaling***, ***Haar-like***, and ***Walsh-like*** vectors

1. Generate a full recursive partitioning of the graph ⇒ Fiedler vectors
2. Generate an orthonormal basis for level $j_{\max}$ (the finest level) ⇒ ***scaling vectors*** on the single-node regions
    - As with HGLET, the notation is $\boldsymbol{\psi}^{j}_{k,l}$
3. Using the basis for level $j_{\max}$, generate an orthonormal basis for level $j_{\max}-1$ ⇒ ***scaling*** and ***Haar-like*** vectors
4. Repeat... Using the basis for level j, generate an orthonormal basis for level $j-1$ ⇒ ***scaling***, ***Haar-like***, and ***Walsh-like*** vectors

$$\left[\ \boldsymbol{\psi}^{j_{\max}}_{0,0}\ \right]\quad \left[\ \boldsymbol{\psi}^{j_{\max}}_{1,0}\ \right]\quad \left[\ \boldsymbol{\psi}^{j_{\max}}_{2,0}\ \right]\quad \left[\ \boldsymbol{\psi}^{j_{\max}}_{3,0}\ \right]\ \cdots\ \left[\ \boldsymbol{\psi}^{j_{\max}}_{K^{j_{\max}}-2,0}\ \right]\qquad \left[\ \boldsymbol{\psi}^{j_{\max}}_{K^{j_{\max}}-1,0}\ \right]$$

1. Generate a full recursive partitioning of the graph ⇒ Fiedler vectors
2. Generate an orthonormal basis for level $j_{\max}$ (the finest level) ⇒ ***scaling vectors*** on the single-node regions
   - As with HGLET, the notation is $\boldsymbol{\psi}^{j}_{k,l}$
3. Using the basis for level $j_{\max}$, generate an orthonormal basis for level $j_{\max}-1$ ⇒ ***scaling*** and ***Haar-like*** vectors
4. Repeat... Using the basis for level j, generate an orthonormal basis for level $j-1$ ⇒ *scaling*, *Haar-like*, and *Walsh-like* vectors

$$\left[\ \boldsymbol{\psi}^{j_{\max}-1}_{0,0}\quad \boldsymbol{\psi}^{j_{\max}-1}_{0,1}\ \right]\ \left[\ \boldsymbol{\psi}^{j_{\max}-1}_{1,0}\quad \boldsymbol{\psi}^{j_{\max}-1}_{1,1}\ \right]\ \cdots\ \left[\ \boldsymbol{\psi}^{j_{\max}-1}_{K^{j_{\max}-1}-1,0}\quad \boldsymbol{\psi}^{j_{\max}-1}_{K^{j_{\max}-1}-1,1}\ \right]$$

$$\left[\ \boldsymbol{\psi}^{j_{\max}}_{0,0}\ \right]\ \left[\ \boldsymbol{\psi}^{j_{\max}}_{1,0}\ \right]\ \left[\ \boldsymbol{\psi}^{j_{\max}}_{2,0}\ \right]\ \left[\ \boldsymbol{\psi}^{j_{\max}}_{3,0}\ \right]\ \cdots\ \left[\ \boldsymbol{\psi}^{j_{\max}}_{K^{j_{\max}}-2,0}\ \right]\quad \left[\ \boldsymbol{\psi}^{j_{\max}}_{K^{j_{\max}}-1,0}\ \right]$$

1. Generate a full recursive partitioning of the graph $\Rightarrow$ Fiedler vectors
2. Generate an orthonormal basis for level $j_{\max}$ (the finest level) $\Rightarrow$ ***scaling vectors*** on the single-node regions
   - As with HGLET, the notation is $\boldsymbol{\psi}^{j}_{k,l}$
3. Using the basis for level $j_{\max}$, generate an orthonormal basis for level $j_{\max}-1$ $\Rightarrow$ ***scaling*** and ***Haar-like*** vectors
4. Repeat... Using the basis for level j, generate an orthonormal basis for level $j-1$ $\Rightarrow$ ***scaling***, ***Haar-like***, and ***Walsh-like*** vectors

$$\left[\ \boldsymbol{\psi}^{0}_{0,0} \quad \boldsymbol{\psi}^{0}_{0,1} \quad \boldsymbol{\psi}^{0}_{0,2} \quad \boldsymbol{\psi}^{0}_{0,3} \quad \cdots \quad \boldsymbol{\psi}^{0}_{0,n-2} \quad \boldsymbol{\psi}^{0}_{0,n-1}\ \right]$$

$$\vdots$$

$$\left[\ \boldsymbol{\psi}^{j_{\max}-1}_{0,0} \quad \boldsymbol{\psi}^{j_{\max}-1}_{0,1}\ \right] \left[\ \boldsymbol{\psi}^{j_{\max}-1}_{1,0} \quad \boldsymbol{\psi}^{j_{\max}-1}_{1,1}\ \right] \cdots \left[\ \boldsymbol{\psi}^{j_{\max}-1}_{K^{j_{\max}-1}-1,0} \quad \boldsymbol{\psi}^{j_{\max}-1}_{K^{j_{\max}-1}-1,1}\ \right]$$

$$\left[\ \boldsymbol{\psi}^{j_{\max}}_{0,0}\ \right] \left[\ \boldsymbol{\psi}^{j_{\max}}_{1,0}\ \right] \left[\ \boldsymbol{\psi}^{j_{\max}}_{2,0}\ \right] \left[\ \boldsymbol{\psi}^{j_{\max}}_{3,0}\ \right] \cdots \left[\ \boldsymbol{\psi}^{j_{\max}}_{K^{j_{\max}}-2,0}\ \right] \left[\ \boldsymbol{\psi}^{j_{\max}}_{K^{j_{\max}}-1,0}\ \right]$$

Notation

- $j \in [0, j_{\text{max}}]$ denotes **levels**, where $j = 0$ is the coarsest level (i.e., the entire graph) and $j = j_{\text{max}}$ is the finest level (each region is a single node)
- $k \in [0, K^j)$ indexes the **regions** on level j
- n_k^j denote the number of nodes in region k on level j
- If $n_k^j > 1$ then we let k' and $k' + 1$ denote the indices of the children regions on level $j + 1$

Notation

- $j \in [0, j_{\max}]$ denotes **levels**, where $j = 0$ is the coarsest level (i.e., the entire graph) and $j = j_{\max}$ is the finest level (each region is a single node)
- $k \in [0, K^j)$ indexes the **regions** on level j
- n_k^j denote the number of nodes in region k on level j
- If $n_k^j > 1$ then we let k' and $k' + 1$ denote the indices of the children regions on level $j + 1$

Notation

- $j \in [0, j_{\max}]$ denotes **levels**, where $j = 0$ is the coarsest level (i.e., the entire graph) and $j = j_{\max}$ is the finest level (each region is a single node)
- $k \in [0, K^j)$ indexes the **regions** on level j
- n_k^j denote the number of nodes in region k on level j
- If $n_k^j > 1$ then we let k' and $k'+1$ denote the indices of the children regions on level $j+1$

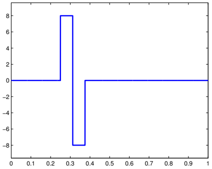
Notation

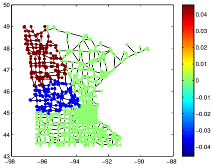
- $j \in [0, j_{\max}]$ denotes **levels**, where $j = 0$ is the coarsest level (i.e., the entire graph) and $j = j_{\max}$ is the finest level (each region is a single node)
- $k \in [0, K^j)$ indexes the **regions** on level j
- n_k^j denote the number of nodes in region k on level j
- If $n_k^j > 1$ then we let k' and $k' + 1$ denote the indices of the children regions on level $j + 1$

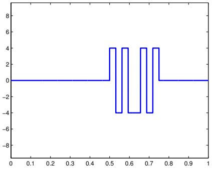
Basis Vector & Coefficient Notation

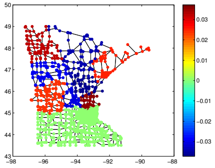
GHWT basis vectors and coefficients are written as $\boldsymbol{\psi}^j_{k,\ell}$ and $c^j_{k,\ell}$, respectively, where j and k correspond to level and region and ℓ is the **tag**.

- $\ell = 0 \Rightarrow$ **scaling coefficient/basis vector**
- $\ell = 1 \Rightarrow$ **Haar-like coefficient/basis vector**
- $\ell \geq 2 \Rightarrow$ **Walsh-like coefficient/basis vector**


(a) Haar function on $\mathbb{R}$


(b) Haar-like vector $\boldsymbol{\psi}^2_{0,1}$


(c) Haar-Walsh wavelet packet on $\mathbb{R}$

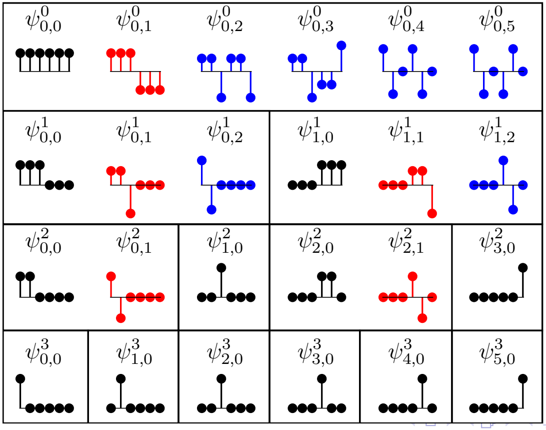

(d) Walsh-like vector $\boldsymbol{\psi}^1_{0,5}$

Remarks

- For an unweighted path graph, this yields a dictionary of Haar-Walsh functions
- As with the HGLET, we can select an orthonormal basis for the entire graph by taking the union of orthonormal bases on disjoint regions

Remarks

- For an unweighted path graph, this yields a dictionary of Haar-Walsh functions
- As with the HGLET, we can select an orthonormal basis for the entire graph by taking the union of orthonormal bases on disjoint regions



Remarks

- We can also reorder and regroup the vectors on each level of the GHWT dictionary according to their type (**scaling**, **Haar-like**, or **Walsh-like**)

Remarks

- We can also reorder and regroup the vectors on each level of the GHWT dictionary according to their type (**scaling**, **Haar-like**, or **Walsh-like**)

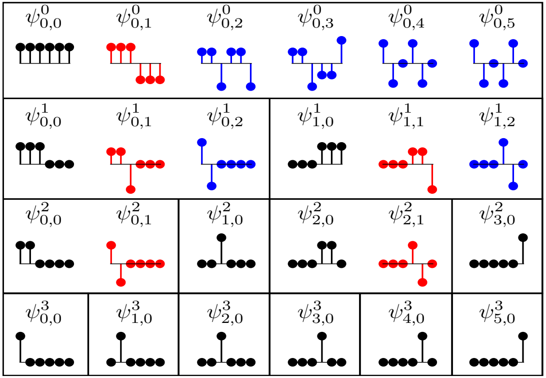


Figure: Default dictionary; i.e., coarse-to-fine

Remarks

- We can also reorder and regroup the vectors on each level of the GHWT dictionary according to their type (**scaling**, **Haar-like**, or **Walsh-like**)

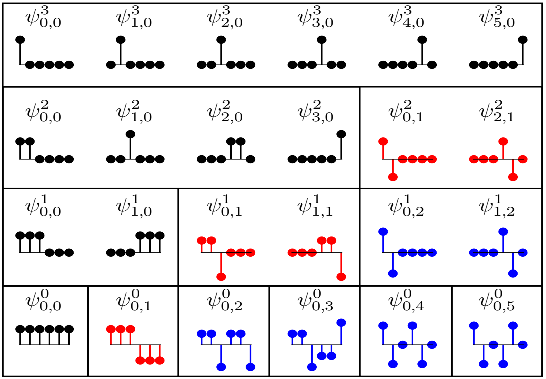


Figure: Reordered & regrouped dictionary; i.e., fine-to-coarse

Remarks

- We can also reorder and regroup the vectors on each level of the GHWT dictionary according to their type (**scaling**, **Haar-like**, or **Walsh-like**)

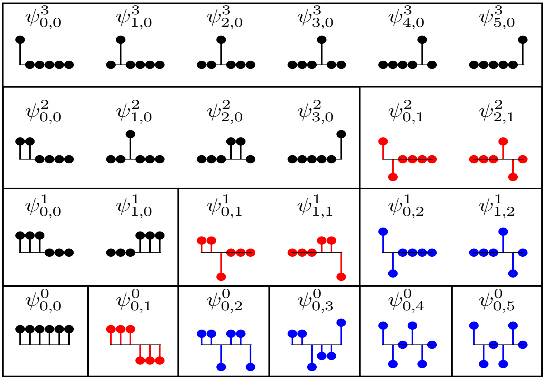


Figure: Reordered & regrouped dictionary; i.e., fine-to-coarse

- This reorganization gives us *more options for choosing a good basis*

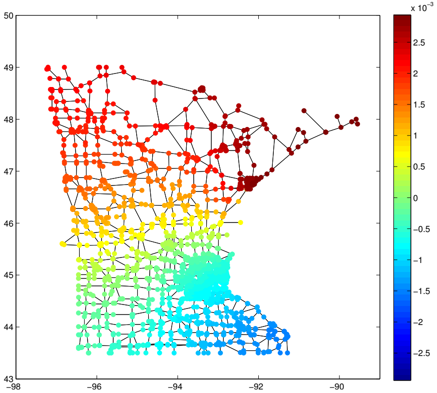
HGLET vs. GHWT

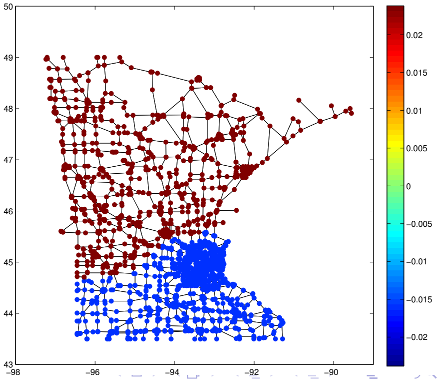
Here we display some of the basis vectors generated by our HGLET (left) and GHWT (right) schemes on the MN road network. (Note: $j = 0$ is the coarsest scale, $j = 14$ is the finest.)

HGLET vs. GHWT

Here we display some of the basis vectors generated by our HGLET (left) and GHWT (right) schemes on the MN road network. (Note: $j = 0$ is the coarsest scale, $j = 14$ is the finest.)

Level $j = 0$, Region $k = 0$, $l = 1$

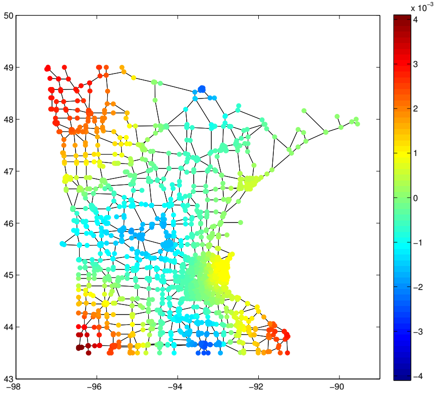


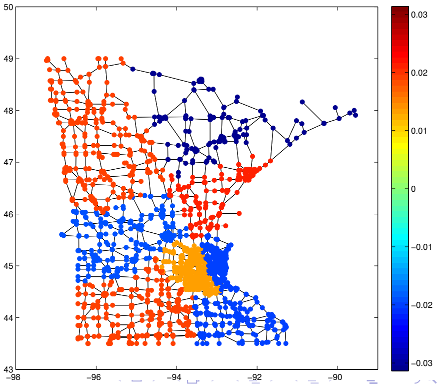


HGLET vs. GHWT

Here we display some of the basis vectors generated by our HGLET (left) and GHWT (right) schemes on the MN road network. (Note: $j = 0$ is the coarsest scale, $j = 14$ is the finest.)

Level $j = 0$,   Region $k = 0$, $l = 7$

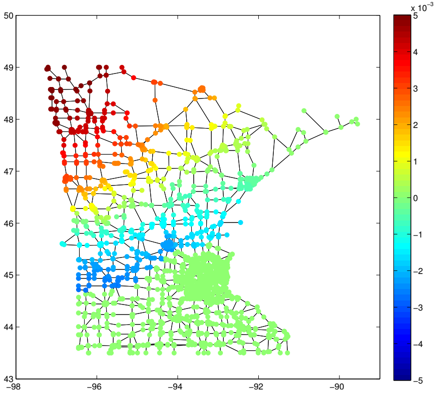


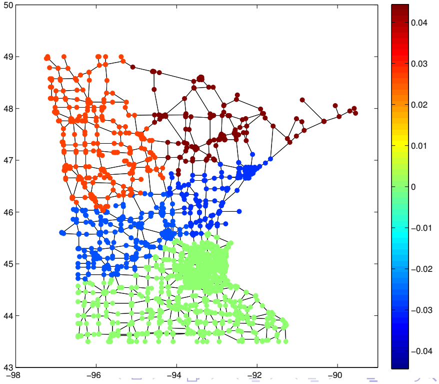


HGLET vs. GHWT

Here we display some of the basis vectors generated by our HGLET (left) and GHWT (right) schemes on the MN road network. (Note: $j = 0$ is the coarsest scale, $j = 14$ is the finest.)

Level $j = 1$,   Region $k = 0$, $l = 2$

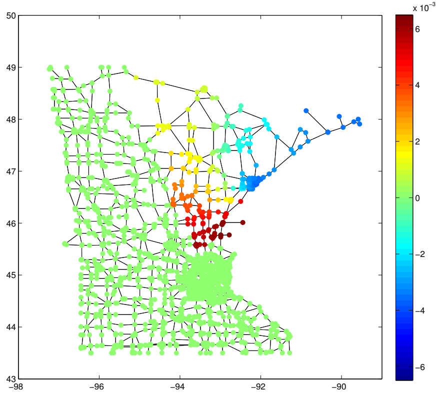


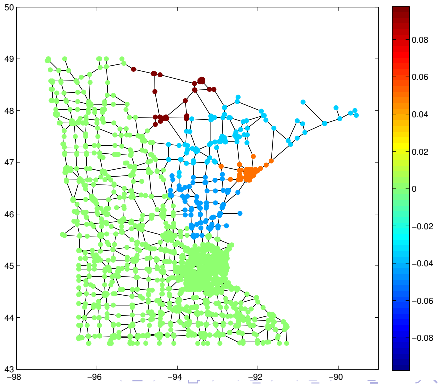


HGLET vs. GHWT

Here we display some of the basis vectors generated by our HGLET (left) and GHWT (right) schemes on the MN road network. (Note: $j = 0$ is the coarsest scale, $j = 14$ is the finest.)

Level $j = 2$, Region $k = 1$, $l = 2$

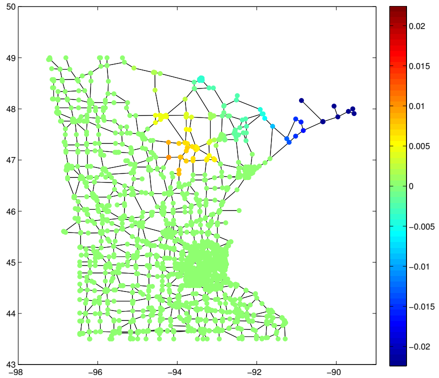


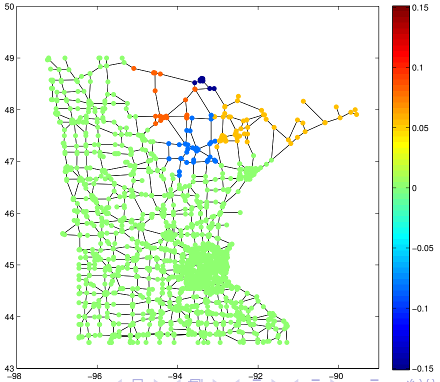


HGLET vs. GHWT

Here we display some of the basis vectors generated by our HGLET (left) and GHWT (right) schemes on the MN road network. (Note: $j = 0$ is the coarsest scale, $j = 14$ is the finest.)

Level $j = 3$, Region $k = 2$, $l = 2$





Computational Complexity: HGLET vs. GHWT

- Recursive Partitioning (RP) via Fiedler vectors costs from $O(n \log n)$ to $O(n^2)$ depending on an input graph
- Given a recursive partitioning with $O(\log n)$ levels, the computational cost of the GHWT is $O(n \log n)$ whereas that of the HGLET is $O(n^3)$
- The following table shows the results of our numerical experiments computed on a desktop PC (CPU: 16 GB RAM, 3.2 GHz Clock Speed):

Dataset	n	$j_{\max}$	RP	HGLET	GHWT
Dendritic Tree	1154	13	0.49 s	0.99 s	0.07 s
MN Road Network	2640	14	0.76 s	10.57 s	0.18 s
Facebook Graph	4039	46	18.10 s	57.15 s	1.17 s
Brain Mesh Data	127083	21	164.18 s	N/A	11.59 s

Computational Complexity: HGLET vs. GHWT

- Recursive Partitioning (RP) via Fiedler vectors costs from $O(n \log n)$ to $O(n^2)$ depending on an input graph
- Given a recursive partitioning with $O(\log n)$ levels, the computational cost of the GHWT is $O(n \log n)$ whereas that of the HGLET is $O(n^3)$
- The following table shows the results of our numerical experiments computed on a desktop PC (CPU: 16 GB RAM, 3.2 GHz Clock Speed):

Dataset	n	$j_{\max}$	RP	HGLET	GHWT
Dendritic Tree	1154	13	0.49 s	0.99 s	0.07 s
MN Road Network	2640	14	0.76 s	10.57 s	0.18 s
Facebook Graph	4039	46	18.10 s	57.15 s	1.17 s
Brain Mesh Data	127083	21	164.18 s	N/A	11.59 s

Computational Complexity: HGLET vs. GHWT

- Recursive Partitioning (RP) via Fiedler vectors costs from $O(n \log n)$ to $O(n^2)$ depending on an input graph
- Given a recursive partitioning with $O(\log n)$ levels, the computational cost of the GHWT is $O(n \log n)$ whereas that of the HGLET is $O(n^3)$
- The following table shows the results of our numerical experiments computed on a desktop PC (CPU: 16 GB RAM, 3.2 GHz Clock Speed):

Dataset	n	$j_{\max}$	RP	HGLET	GHWT
Dendritic Tree	1154	13	0.49 s	0.99 s	0.07 s
MN Road Network	2640	14	0.76 s	10.57 s	0.18 s
Facebook Graph	4039	46	18.10 s	57.15 s	1.17 s
Brain Mesh Data	127083	21	164.18 s	N/A	11.59 s

Related Work

The following articles also discussed the Haar-like transform on graphs and trees, but *neither the Walsh-Hadamard transform nor Wavelet Packets* on them are discussed:

1. A. D. Szlam, M. Maggioni, R. R. Coifman, and J. C. Bremer, Jr., "Diffusion-driven multiscale analysis on manifolds and graphs: top-down and bottom-up constructions," in *Wavelets XI* (M. Papadakis et al. eds.), *Proc. SPIE 5914*, Paper # 59141D, 2005.
2. F. Murtagh, "The Haar wavelet transform of a dendrogram," *J. Classification*, vol. 24, pp. 3–32, 2007.
3. A. Lee, B. Nadler, and L. Wasserman, "Treelets–an adaptive multi-scale basis for sparse unordered data," *Ann. Appl. Stat.*, vol. 2, pp. 435–471, 2008.
4. M. Gavish, B. Nadler, and R. Coifman, "Multiscale wavelets on trees, graphs and high dimensional data: Theory and applications to semi supervised learning," in *Proc. 27th Intern. Conf. Machine Learning* (J. Fürnkranz et al. eds.), pp. 367–374, Omnipress, Haifa, 2010.

Outline

Best-Basis Algorithms for HGLET & GHWT

Coifman and Wickerhauser (1992) developed the best-basis algorithm as a means of selecting the basis from a dictionary of wavelet packets that is "best" for approximation/compression.

We generalize this approach, developing and implementing an algorithm for selecting the basis from the dictionary of HGLET / GHWT bases that is "best" for approximation.

We require a cost functional $\mathscr{J}$. For example:

$$\mathscr{J}(\boldsymbol{x}) = \|\boldsymbol{x}\|_p := \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p} \qquad 0 < p \leq 1$$

Another example cost functional is based on ...

Best-Basis Algorithms for HGLET & GHWT

Coifman and Wickerhauser (1992) developed the best-basis algorithm as a means of selecting the basis from a dictionary of wavelet packets that is "best" for approximation/compression.

We generalize this approach, developing and implementing an algorithm for selecting the basis from the dictionary of HGLET / GHWT bases that is "best" for approximation.

We require a cost functional $\mathscr{J}$. For example:

$$\mathscr{J}(\boldsymbol{x}) = \|\boldsymbol{x}\|_p := \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p} \qquad 0 < p \leq 1$$

Another example cost functional is based on ...

Best-Basis Algorithms for HGLET & GHWT

Coifman and Wickerhauser (1992) developed the best-basis algorithm as a means of selecting the basis from a dictionary of wavelet packets that is "best" for approximation/compression.

We generalize this approach, developing and implementing an algorithm for selecting the basis from the dictionary of HGLET / GHWT bases that is "best" for approximation.

We require a cost functional $\mathscr{J}$. For example:

$$\mathscr{J}(\boldsymbol{x}) = \|\boldsymbol{x}\|_p := \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p} \qquad 0 < p \leq 1$$

Another example cost functional is based on ...

The Minimum Description Length (MDL) Criterion

Given two or more competing models that are supposed to generate the observed data, choose the model that describes the data and the model itself with the least amount of bits. The basic idea behind the MDL principle: The more you can compress a sequence of data, the more regularity you have detected in the data, hence the more you have learned from the data.

⇒ Need to specify the model class for a given signal

⇒ No time today to go over the details of the MDL philosophy and the actual cost functional we use; More details can be found in:

- J. Rissanen, *Information and Complexity in Statistical Modeling*, Springer, 2007.
- P. D. Grünwald, *The Minimum Description Length Principle*, The MIT Press, 2007.
- N. Saito, "Simultaneous noise suppression and signal compression using a library of orthonormal bases and the minimum description length criterion," in *Wavelets in Geophysics* (E. Foufoula-Georgiou and P. Kumar, eds.), Chap. XI, pp. 299–324, Academic Press, 1994.
- N. Saito and E. Woei, "Simultaneous segmentation, compression, and denoising of signals using polyharmonic local sine transform and minimum description length criterion," *2005 IEEE/SP 13th Workshop on Statistical Signal Processing*, pp. 315–320, 2005.

The Minimum Description Length (MDL) Criterion

Given two or more competing models that are supposed to generate the observed data, choose the model that describes the data and the model itself with the least amount of bits. The basic idea behind the MDL principle: The more you can compress a sequence of data, the more regularity you have detected in the data, hence the more you have learned from the data.

⇒ Need to specify the model class for a given signal

⇒ No time today to go over the details of the MDL philosophy and the actual cost functional we use; More details can be found in:

- J. Rissanen, *Information and Complexity in Statistical Modeling*, Springer, 2007.
- P. D. Grünwald, *The Minimum Description Length Principle*, The MIT Press, 2007.
- N. Saito, "Simultaneous noise suppression and signal compression using a library of orthonormal bases and the minimum description length criterion," in *Wavelets in Geophysics* (E. Foufoula-Georgiou and P. Kumar, eds.), Chap. XI, pp. 299–324, Academic Press, 1994.
- N. Saito and E. Woei, "Simultaneous segmentation, compression, and denoising of signals using polyharmonic local sine transform and minimum description length criterion," *2005 IEEE/SP 13th Workshop on Statistical Signal Processing*, pp. 315–320, 2005.

The Minimum Description Length (MDL) Criterion

Given two or more competing models that are supposed to generate the observed data, choose the model that describes the data and the model itself with the least amount of bits. The basic idea behind the MDL principle: The more you can compress a sequence of data, the more regularity you have detected in the data, hence the more you have learned from the data.

$\Rightarrow$ Need to specify the model class for a given signal

$\Rightarrow$ No time today to go over the details of the MDL philosophy and the actual cost functional we use; More details can be found in:

- J. Rissanen, *Information and Complexity in Statistical Modeling*, Springer, 2007.
- P. D. Grünwald, *The Minimum Description Length Principle*, The MIT Press, 2007.
- N. Saito, "Simultaneous noise suppression and signal compression using a library of orthonormal bases and the minimum description length criterion," in *Wavelets in Geophysics* (E. Foufoula-Georgiou and P. Kumar, eds.), Chap. XI, pp. 299–324, Academic Press, 1994.
- N. Saito and E. Woei, "Simultaneous segmentation, compression, and denoising of signals using polyharmonic local sine transform and minimum description length criterion," *2005 IEEE/SP 13th Workshop on Statistical Signal Processing*, pp. 315–320, 2005.

The Minimum Description Length (MDL) Criterion

Given two or more competing models that are supposed to generate the observed data, choose the model that describes the data and the model itself with the least amount of bits. The basic idea behind the MDL principle: The more you can compress a sequence of data, the more regularity you have detected in the data, hence the more you have learned from the data.

- $\Rightarrow$ Need to specify the model class for a given signal
- $\Rightarrow$ No time today to go over the details of the MDL philosophy and the actual cost functional we use; More details can be found in:

- J. Rissanen, *Information and Complexity in Statistical Modeling*, Springer, 2007.
- P. D. Grünwald, *The Minimum Description Length Principle*, The MIT Press, 2007.
- N. Saito, "Simultaneous noise suppression and signal compression using a library of orthonormal bases and the minimum description length criterion," in *Wavelets in Geophysics* (E. Foufoula-Georgiou and P. Kumar, eds.), Chap. XI, pp. 299–324, Academic Press, 1994.
- N. Saito and E. Woei, "Simultaneous segmentation, compression, and denoising of signals using polyharmonic local sine transform and minimum description length criterion," *2005 IEEE/SP 13th Workshop on Statistical Signal Processing*, pp. 315–320, 2005.

The Minimum Description Length (MDL) Criterion

Given two or more competing models that are supposed to generate the observed data, choose the model that describes the data and the model itself with the least amount of bits. The basic idea behind the MDL principle: The more you can compress a sequence of data, the more regularity you have detected in the data, hence the more you have learned from the data.

- ⇒ Need to specify the model class for a given signal
- ⇒ No time today to go over the details of the MDL philosophy and the actual cost functional we use; More details can be found in:

- J. Rissanen, *Information and Complexity in Statistical Modeling*, Springer, 2007.
- P. D. Grünwald, *The Minimum Description Length Principle*, The MIT Press, 2007.
- N. Saito, "Simultaneous noise suppression and signal compression using a library of orthonormal bases and the minimum description length criterion," in *Wavelets in Geophysics* (E. Foufoula-Georgiou and P. Kumar, eds.), Chap. XI, pp. 299–324, Academic Press, 1994.
- N. Saito and E. Woei, "Simultaneous segmentation, compression, and denoising of signals using polyharmonic local sine transform and minimum description length criterion," *2005 IEEE/SP 13th Workshop on Statistical Signal Processing*, pp. 315–320, 2005.

$$\begin{array}{cccccc} [& \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,n^0_0-1}] \\ & c^0_{0,0} & c^0_{0,1} & c^0_{0,2} & \cdots & c^0_{0,n^0_0-1} \end{array}$$

$$\begin{array}{cc} \left[\boldsymbol{\phi}^1_{0,0} \; \boldsymbol{\phi}^1_{0,1} \; \boldsymbol{\phi}^1_{0,2} \; \cdots \; \boldsymbol{\phi}^1_{0,n^1_0-1} \right] & \left[\boldsymbol{\phi}^1_{1,0} \; \boldsymbol{\phi}^1_{1,1} \; \boldsymbol{\phi}^1_{1,2} \; \cdots \; \boldsymbol{\phi}^1_{1,n^1_1-1} \right] \\ c^1_{0,0} \; c^1_{0,1} \; c^1_{0,2} \; \cdots \; c^1_{0,n^1_0-1} & c^1_{1,0} \; c^1_{1,1} \; c^1_{1,2} \; \cdots \; c^1_{1,n^1_1-1} \end{array}$$

$$\begin{array}{cccc} \left[\boldsymbol{\phi}^2_{0,0} \boldsymbol{\phi}^2_{0,1} \cdots \boldsymbol{\phi}^2_{0,n^2_0-1} \right] & \left[\boldsymbol{\phi}^2_{1,0} \boldsymbol{\phi}^2_{1,1} \cdots \boldsymbol{\phi}^2_{1,n^2_1-1} \right] & \left[\boldsymbol{\phi}^2_{2,0} \boldsymbol{\phi}^2_{2,1} \cdots \boldsymbol{\phi}^2_{2,n^2_2-1} \right] & \left[\boldsymbol{\phi}^2_{3,0} \boldsymbol{\phi}^2_{3,1} \cdots \boldsymbol{\phi}^2_{3,n^2_3-1} \right] \\ c^2_{0,0} \; c^2_{0,1} \cdots c^2_{0,n^2_0-1} & c^2_{1,0} \; c^2_{1,1} \cdots c^2_{1,n^2_1-1} & c^2_{2,0} \; c^2_{2,1} \cdots c^2_{2,n^2_2-1} & c^2_{3,0} \; c^2_{3,1} \cdots c^2_{3,n^2_3-1} \end{array}$$

$$\begin{array}{cccccc} [& \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,n^0_0-1}] \\ & c^0_{0,0} & c^0_{0,1} & c^0_{0,2} & \cdots & c^0_{0,n^0_0-1} \end{array}$$

$$\begin{array}{cc} \begin{bmatrix} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,n^1_0-1} \end{bmatrix} & \begin{bmatrix} \boldsymbol{\phi}^1_{1,0} & \boldsymbol{\phi}^1_{1,1} & \boldsymbol{\phi}^1_{1,2} & \cdots & \boldsymbol{\phi}^1_{1,n^1_1-1} \end{bmatrix} \\ c^1_{0,0} \; c^1_{0,1} \; c^1_{0,2} \; \cdots \; c^1_{0,n^1_0-1} & c^1_{1,0} \; c^1_{1,1} \; c^1_{1,2} \; \cdots \; c^1_{1,n^1_1-1} \end{array}$$

$$\begin{array}{cccc} \begin{bmatrix} \boldsymbol{\phi}^2_{0,0}\, \boldsymbol{\phi}^2_{0,1} \cdots \boldsymbol{\phi}^2_{0,n^2_0-1} \end{bmatrix} & \begin{bmatrix} \boldsymbol{\phi}^2_{1,0}\, \boldsymbol{\phi}^2_{1,1} \cdots \boldsymbol{\phi}^2_{1,n^2_1-1} \end{bmatrix} & \begin{bmatrix} \boldsymbol{\phi}^2_{2,0}\, \boldsymbol{\phi}^2_{2,1} \cdots \boldsymbol{\phi}^2_{2,n^2_2-1} \end{bmatrix} & \begin{bmatrix} \boldsymbol{\phi}^2_{3,0}\, \boldsymbol{\phi}^2_{3,1} \cdots \boldsymbol{\phi}^2_{3,n^2_3-1} \end{bmatrix} \\ c^2_{0,0}\; c^2_{0,1} \cdots c^2_{0,n^2_0-1} & c^2_{1,0}\; c^2_{1,1} \cdots c^2_{1,n^2_1-1} & c^2_{2,0}\; c^2_{2,1} \cdots c^2_{2,n^2_2-1} & c^2_{3,0}\; c^2_{3,1} \cdots c^2_{3,n^2_3-1} \end{array}$$

$$\begin{bmatrix} \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,n^0_0-1} \end{bmatrix}$$
$$c^0_{0,0} \quad c^0_{0,1} \quad c^0_{0,2} \quad \cdots \quad c^0_{0,n^0_0-1}$$

$$\begin{bmatrix} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,n^1_0-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^1_{1,0} & \boldsymbol{\phi}^1_{1,1} & \boldsymbol{\phi}^1_{1,2} & \cdots & \boldsymbol{\phi}^1_{1,n^1_1-1} \end{bmatrix}$$
$$c^1_{0,0} \quad c^1_{0,1} \quad c^1_{0,2} \quad \cdots \quad c^1_{0,n^1_0-1} \qquad c^1_{1,0} \quad c^1_{1,1} \quad c^1_{1,2} \quad \cdots \quad c^1_{1,n^1_1-1}$$

$$\begin{bmatrix} \boldsymbol{\phi}^2_{0,0} \, \boldsymbol{\phi}^2_{0,1} \cdots \boldsymbol{\phi}^2_{0,n^2_0-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{1,0} \, \boldsymbol{\phi}^2_{1,1} \cdots \boldsymbol{\phi}^2_{1,n^2_1-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{2,0} \, \boldsymbol{\phi}^2_{2,1} \cdots \boldsymbol{\phi}^2_{2,n^2_2-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{3,0} \, \boldsymbol{\phi}^2_{3,1} \cdots \boldsymbol{\phi}^2_{3,n^2_3-1} \end{bmatrix}$$
$$c^2_{0,0} \; c^2_{0,1} \cdots c^2_{0,n^2_0-1} \quad c^2_{1,0} \; c^2_{1,1} \cdots c^2_{1,n^2_1-1} \quad c^2_{2,0} \; c^2_{2,1} \cdots c^2_{2,n^2_2-1} \quad c^2_{3,0} \; c^2_{3,1} \cdots c^2_{3,n^2_3-1}$$

$$\mathscr{J}\left(\boldsymbol{c}^1_0\right) \overset{?}{<} \mathscr{J}\left(\boldsymbol{c}^2_0; \boldsymbol{c}^2_1\right)$$

$$\begin{bmatrix} \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,n^0_0-1} \end{bmatrix}$$
$$c^0_{0,0} \quad c^0_{0,1} \quad c^0_{0,2} \quad \cdots \quad c^0_{0,n^0_0-1}$$

$$\begin{bmatrix} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,n^1_0-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^1_{1,0} & \boldsymbol{\phi}^1_{1,1} & \boldsymbol{\phi}^1_{1,2} & \cdots & \boldsymbol{\phi}^1_{1,n^1_1-1} \end{bmatrix}$$
$$c^1_{0,0} \quad c^1_{0,1} \quad c^1_{0,2} \quad \cdots \quad c^1_{0,n^1_0-1} \qquad c^1_{1,0} \quad c^1_{1,1} \quad c^1_{1,2} \quad \cdots \quad c^1_{1,n^1_1-1}$$

$$\begin{bmatrix} \boldsymbol{\phi}^2_{2,0}\, \boldsymbol{\phi}^2_{2,1} \cdots \boldsymbol{\phi}^2_{2,n^2_2-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^2_{3,0}\, \boldsymbol{\phi}^2_{3,1} \cdots \boldsymbol{\phi}^2_{3,n^2_3-1} \end{bmatrix}$$
$$c^2_{2,0}\; c^2_{2,1} \cdots c^2_{2,n^2_2-1} \qquad c^2_{3,0}\; c^2_{3,1} \cdots c^2_{3,n^2_3-1}$$

$$\mathscr{J}\left(\boldsymbol{c}^1_0\right) < \mathscr{J}\left(\boldsymbol{c}^2_0; \boldsymbol{c}^2_1\right)$$

$$\begin{array}{ccccc} \Big[\ \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,n^0_0-1}\ \Big] \\ c^0_{0,0} & c^0_{0,1} & c^0_{0,2} & \cdots & c^0_{0,n^0_0-1} \end{array}$$

$$\begin{array}{ccccc} \Big[\ \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,n^1_0-1}\ \Big] \\ c^1_{0,0} & c^1_{0,1} & c^1_{0,2} & \cdots & c^1_{0,n^1_0-1} \end{array} \quad \begin{array}{ccccc} \Big[\ \boldsymbol{\phi}^1_{1,0} & \boldsymbol{\phi}^1_{1,1} & \boldsymbol{\phi}^1_{1,2} & \cdots & \boldsymbol{\phi}^1_{1,n^1_1-1}\ \Big] \\ c^1_{1,0} & c^1_{1,1} & c^1_{1,2} & \cdots & c^1_{1,n^1_1-1} \end{array}$$

$$\begin{array}{cccc} \Big[\boldsymbol{\phi}^2_{2,0} & \boldsymbol{\phi}^2_{2,1} & \cdots & \boldsymbol{\phi}^2_{2,n^2_2-1}\Big] \\ c^2_{2,0} & c^2_{2,1} & \cdots & c^2_{2,n^2_2-1} \end{array} \quad \begin{array}{cccc} \Big[\boldsymbol{\phi}^2_{3,0} & \boldsymbol{\phi}^2_{3,1} & \cdots & \boldsymbol{\phi}^2_{3,n^2_3-1}\Big] \\ c^2_{3,0} & c^2_{3,1} & \cdots & c^2_{3,n^2_3-1} \end{array}$$

$$\mathscr{J}\left(\boldsymbol{c}^1_1\right) \overset{?}{<} \mathscr{J}\left(\boldsymbol{c}^2_2; \boldsymbol{c}^2_3\right)$$

$$\begin{array}{ccccc} [\ \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,n^0_0-1}\ ] \\ c^0_{0,0} & c^0_{0,1} & c^0_{0,2} & \cdots & c^0_{0,n^0_0-1} \end{array}$$

$$\begin{array}{ccccc} [\ \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,n^1_0-1}\ ] \\ c^1_{0,0} & c^1_{0,1} & c^1_{0,2} & \cdots & c^1_{0,n^1_0-1} \end{array}$$

$$\begin{array}{cccc} [\boldsymbol{\phi}^2_{2,0} & \boldsymbol{\phi}^2_{2,1} & \cdots & \boldsymbol{\phi}^2_{2,n^2_2-1}] \\ c^2_{2,0} & c^2_{2,1} & \cdots & c^2_{2,n^2_2-1} \end{array} \quad \begin{array}{cccc} [\boldsymbol{\phi}^2_{3,0} & \boldsymbol{\phi}^2_{3,1} & \cdots & \boldsymbol{\phi}^2_{3,n^2_3-1}] \\ c^2_{3,0} & c^2_{3,1} & \cdots & c^2_{3,n^2_3-1} \end{array}$$

$$\mathscr{J}\left(\boldsymbol{c}^1_1\right) > \mathscr{J}\left(\boldsymbol{c}^2_2; \boldsymbol{c}^2_3\right)$$

$$\begin{array}{ccccc} \Big[\ \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,n^0_0-1}\ \Big] \\ c^0_{0,0} & c^0_{0,1} & c^0_{0,2} & \cdots & c^0_{0,n^0_0-1} \end{array}$$

$$\begin{array}{ccccc} \Big[\ \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,n^1_0-1}\ \Big] \\ c^1_{0,0} & c^1_{0,1} & c^1_{0,2} & \cdots & c^1_{0,n^1_0-1} \end{array}$$

$$\begin{array}{cccc} \Big[\boldsymbol{\phi}^2_{2,0} & \boldsymbol{\phi}^2_{2,1} & \cdots & \boldsymbol{\phi}^2_{2,n^2_2-1}\Big] \\ c^2_{2,0} & c^2_{2,1} & \cdots & c^2_{2,n^2_2-1} \end{array} \quad \begin{array}{cccc} \Big[\boldsymbol{\phi}^2_{3,0} & \boldsymbol{\phi}^2_{3,1} & \cdots & \boldsymbol{\phi}^2_{3,n^2_3-1}\Big] \\ c^2_{3,0} & c^2_{3,1} & \cdots & c^2_{3,n^2_3-1} \end{array}$$

$$\mathscr{J}\left(\boldsymbol{c}^0_0\right) \overset{?}{<} \mathscr{J}\left(\boldsymbol{c}^1_0; \boldsymbol{c}^2_2; \boldsymbol{c}^2_3\right)$$

$$\begin{bmatrix} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,n^1_0-1} \end{bmatrix}$$
$$c^1_{0,0} \quad c^1_{0,1} \quad c^1_{0,2} \quad \cdots \quad c^1_{0,n^1_0-1}$$

$$\begin{bmatrix} \boldsymbol{\phi}^2_{2,0}\, \boldsymbol{\phi}^2_{2,1} \cdots \boldsymbol{\phi}^2_{2,n^2_2-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^2_{3,0}\, \boldsymbol{\phi}^2_{3,1} \cdots \boldsymbol{\phi}^2_{3,n^2_3-1} \end{bmatrix}$$
$$c^2_{2,0}\; c^2_{2,1} \cdots c^2_{2,n^2_2-1} \qquad c^2_{3,0}\; c^2_{3,1} \cdots c^2_{3,n^2_3-1}$$

$$\mathscr{J}\left(\boldsymbol{c}^0_0\right) > \mathscr{J}\left(\boldsymbol{c}^1_0; \boldsymbol{c}^2_2; \boldsymbol{c}^2_3\right)$$

$$\begin{array}{ccccc} \Big[\, \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,n^1_0-1} \,\Big] \\ c^1_{0,0} & c^1_{0,1} & c^1_{0,2} & \cdots & c^1_{0,n^1_0-1} \end{array}$$

$$\begin{array}{cc} \left[\boldsymbol{\phi}^2_{2,0}\,\boldsymbol{\phi}^2_{2,1}\cdots\boldsymbol{\phi}^2_{2,n^2_2-1}\right] & \left[\boldsymbol{\phi}^2_{3,0}\,\boldsymbol{\phi}^2_{3,1}\cdots\boldsymbol{\phi}^2_{3,n^2_3-1}\right] \\ c^2_{2,0}\;c^2_{2,1}\cdots\;c^2_{2,n^2_2-1} & c^2_{3,0}\;c^2_{3,1}\cdots\;c^2_{3,n^2_3-1} \end{array}$$

$$\mathscr{J}\left(\boldsymbol{c}^0_0\right) > \mathscr{J}\left(\boldsymbol{c}^1_0;\boldsymbol{c}^2_2;\boldsymbol{c}^2_3\right)$$

$$\begin{bmatrix} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,n^1_0-1} \end{bmatrix}$$
$$c^1_{0,0} \quad c^1_{0,1} \quad c^1_{0,2} \quad \cdots \quad c^1_{0,n^1_0-1}$$

$$\begin{bmatrix} \boldsymbol{\phi}^2_{2,0} \, \boldsymbol{\phi}^2_{2,1} \cdots \boldsymbol{\phi}^2_{2,n^2_2-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^2_{3,0} \, \boldsymbol{\phi}^2_{3,1} \cdots \boldsymbol{\phi}^2_{3,n^2_3-1} \end{bmatrix}$$
$$c^2_{2,0} \; c^2_{2,1} \cdots c^2_{2,n^2_2-1} \quad c^2_{3,0} \; c^2_{3,1} \cdots c^2_{3,n^2_3-1}$$

$$\mathscr{J}\left(\boldsymbol{c}^0_0\right) > \mathscr{J}\left(\boldsymbol{c}^1_0; \boldsymbol{c}^2_2; \boldsymbol{c}^2_3\right)$$

According to cost functional $\mathscr{J}$, this is the best basis for approximation.

$$\begin{array}{ccccc} \phi^1_{0,0} & \phi^1_{0,1} & \phi^1_{0,2} & \cdots & \phi^1_{0,n^1_0-1} \\ c^1_{0,0} & c^1_{0,1} & c^1_{0,2} & \cdots & c^1_{0,n^1_0-1} \end{array}$$

$$\begin{array}{cccc} \phi^2_{2,0} & \phi^2_{2,1} & \cdots & \phi^2_{2,n^2_2-1} \\ c^2_{2,0} & c^2_{2,1} & \cdots & c^2_{2,n^2_2-1} \end{array} \quad \begin{array}{cccc} \phi^2_{3,0} & \phi^2_{3,1} & \cdots & \phi^2_{3,n^2_3-1} \\ c^2_{3,0} & c^2_{3,1} & \cdots & c^2_{3,n^2_3-1} \end{array}$$

$$\mathscr{J}\left(\boldsymbol{c}^0_0\right) > \mathscr{J}\left(\boldsymbol{c}^1_0; \boldsymbol{c}^2_2; \boldsymbol{c}^2_3\right)$$

According to cost functional $\mathscr{J}$, this is the best basis for approximation. With the GHWT dictionary, we can run the best-basis algorithm on both the default(*coarse-to-fine*) dictionary and the reorganized (*fine-to-coarse*) dictionary and then compare the results.

Outline

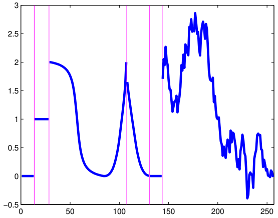
Outline

Motivation

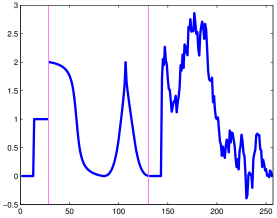
Thanks to the versatility of graphs, graph-based techniques have been used to tackle classical problems, e.g., the nonlocal means algorithm for image denoising can be viewed as a graph-based technique. Here, we demonstrate

the versatility of our graph methods by applying the HGLET and hybrid best-basis algorithm to the problem of denoising and segmenting a 1-D signal sampled on a regular lattice into *meaningful* parts.

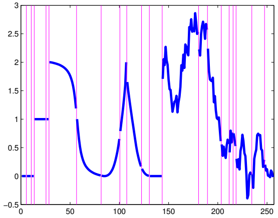
Simply put, the goal is to partition a given 1-D signal into segments based on the characteristics of the signal, which may help interpretation, analysis, compression, etc.



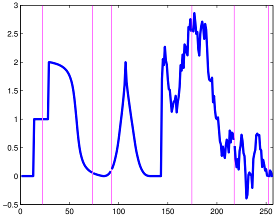
(a) Good



(b) Bad – too few segments



(c) Bad – too many segments



(d) Bad – segmentation lines are poorly placed

Method

We view a 1-D classical signal as signal on an unweighted path graph and proceed as follows.

Iterate until the best basis segmentation converges:

1. **Recursively partition the graph.** We construct a recursive bipartitioning by minimizing *NCut* (*without* using the Fiedler vectors).
2. **Perform the 3 HGLET transforms.** That is, we use the eigenvectors of L, L_{rw}, and L_{sym} of the *unweighted path graph*.
3. **Find the hybrid best basis.** We use the MDL cost functional to search among the coefficients from the 3 HGLET variations.
4. **Modify the graph's edge weights.** If the two adjacent segments in the resulting best basis are represented by the same HGLET transform, then we double the weight of the edge joining them and halve the weights of its two neighboring edges.

Post-process the segments of the final best basis and reconstruct the denoised signal from the quantized coefficients.

Method

We view a 1-D classical signal as signal on an unweighted path graph and proceed as follows.

Iterate until the best basis segmentation converges:

1. **Recursively partition the graph.** We construct a recursive bipartitioning by minimizing *NCut* (*without* using the Fiedler vectors).
2. **Perform the 3 HGLET transforms.** That is, we use the eigenvectors of L, L_{rw}, and L_{sym} of the *unweighted path graph*.
3. **Find the hybrid best basis.** We use the MDL cost functional to search among the coefficients from the 3 HGLET variations.
4. **Modify the graph's edge weights.** If the two adjacent segments in the resulting best basis are represented by the same HGLET transform, then we double the weight of the edge joining them and halve the weights of its two neighboring edges.

Post-process the segments of the final best basis and reconstruct the denoised signal from the quantized coefficients.

Method

We view a 1-D classical signal as signal on an unweighted path graph and proceed as follows.

Iterate until the best basis segmentation converges:

1. **Recursively partition the graph.** We construct a recursive bipartitioning by minimizing *NCut* (*without* using the Fiedler vectors).
2. **Perform the 3 HGLET transforms.** That is, we use the eigenvectors of L, L_{rw}, and L_{sym} of the *unweighted path graph*.
3. **Find the hybrid best basis.** We use the MDL cost functional to search among the coefficients from the 3 HGLET variations.
4. **Modify the graph's edge weights.** If the two adjacent segments in the resulting best basis are represented by the same HGLET transform, then we double the weight of the edge joining them and halve the weights of its two neighboring edges.

Post-process the segments of the final best basis and reconstruct the denoised signal from the quantized coefficients.

Method

We view a 1-D classical signal as signal on an unweighted path graph and proceed as follows.

Iterate until the best basis segmentation converges:

1. **Recursively partition the graph.** We construct a recursive bipartitioning by minimizing *NCut* (*without* using the Fiedler vectors).
2. **Perform the 3 HGLET transforms.** That is, we use the eigenvectors of L, L_{rw}, and L_{sym} of the *unweighted path graph*.
3. **Find the hybrid best basis.** We use the MDL cost functional to search among the coefficients from the 3 HGLET variations.
4. **Modify the graph's edge weights.** If the two adjacent segments in the resulting best basis are represented by the same HGLET transform, then we double the weight of the edge joining them and halve the weights of its two neighboring edges.

Post-process the segments of the final best basis and reconstruct the denoised signal from the quantized coefficients.

Method

We view a 1-D classical signal as signal on an unweighted path graph and proceed as follows.

Iterate until the best basis segmentation converges:

1. **Recursively partition the graph.** We construct a recursive bipartitioning by minimizing *NCut* (*without* using the Fiedler vectors).
2. **Perform the 3 HGLET transforms.** That is, we use the eigenvectors of L, L_{rw}, and L_{sym} of the *unweighted path graph*.
3. **Find the hybrid best basis.** We use the MDL cost functional to search among the coefficients from the 3 HGLET variations.
4. **Modify the graph's edge weights.** If the two adjacent segments in the resulting best basis are represented by the same HGLET transform, then we double the weight of the edge joining them and halve the weights of its two neighboring edges.

Post-process the segments of the final best basis and reconstruct the denoised signal from the quantized coefficients.

Method

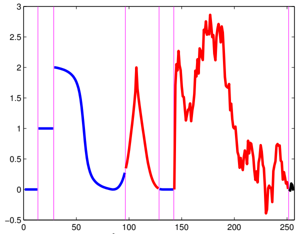
We view a 1-D classical signal as signal on an unweighted path graph and proceed as follows.

Iterate until the best basis segmentation converges:

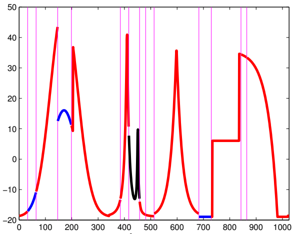
1. **Recursively partition the graph.** We construct a recursive bipartitioning by minimizing *NCut* (*without* using the Fiedler vectors).
2. **Perform the 3 HGLET transforms.** That is, we use the eigenvectors of L, L_{rw}, and L_{sym} of the *unweighted path graph*.
3. **Find the hybrid best basis.** We use the MDL cost functional to search among the coefficients from the 3 HGLET variations.
4. **Modify the graph's edge weights.** If the two adjacent segments in the resulting best basis are represented by the same HGLET transform, then we double the weight of the edge joining them and halve the weights of its two neighboring edges.

Post-process the segments of the final best basis and reconstruct the denoised signal from the quantized coefficients.

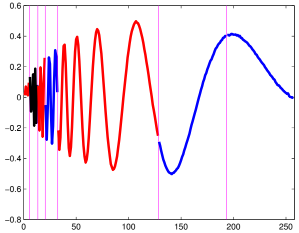
Results



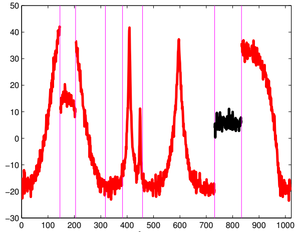
(a) "MSignal" $(n = 256,\ \text{SNR} = 200\ \text{dB})$



(c) "Piece-Regular" $(n = 1024,\ \text{SNR} = 200\ \text{dB})$



(b) "Doppler" $(n = 257,\ \text{SNR} = 40\ \text{dB})$



(d) "Piece-Regular" $(n = 1021,\ \text{SNR} = 20\ \text{dB})$

Figure: **HGLET** L, **HGLET** L_{rw}, and **HGLET** L_{sym} segments.

A Real Signal Example

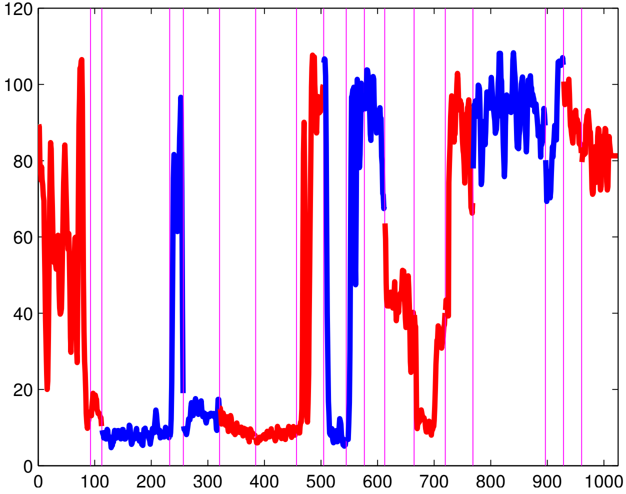
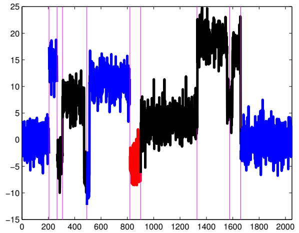
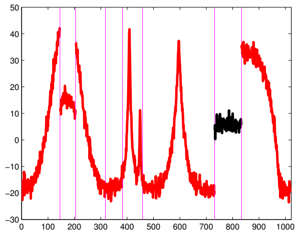
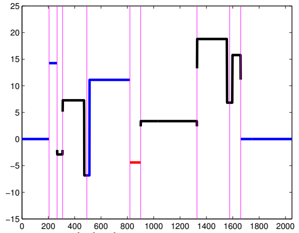


Figure: Gamma-ray log from North Sea subsurface formations. $\Delta z = 6$inches.

Simultaneous Segmentation and Denoising


(a) “Blocks” ($n = 2048$, SNR $= 11.95$ dB)


(c) “Piece-Regular” ($n = 1021$, SNR $= 20$ dB)


(b) Denoised (a) ($n = 2048$, SNR $= 21.60$ dB)

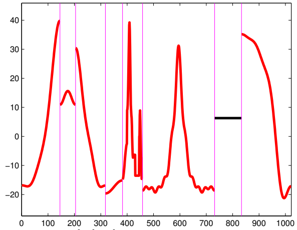

(d) Denoised (c) ($n = 1021$, SNR $= 22.82$ dB)

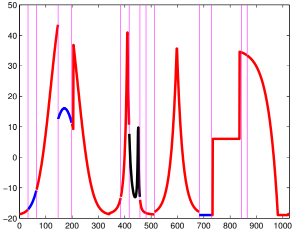
Figure: **HGLET** L, **HGLET** L_{rw}, and **HGLET** L_{sym} segments.

Discussion

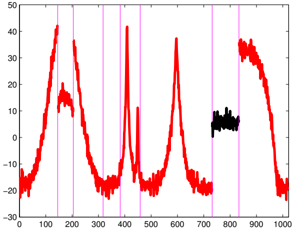
Why does the MDL perform well for this task?

1. The MDL seeks an efficient way to represent the signal ⇒ dissimilar regions are more efficiently represented separately than together
2. Partitions have a cost, and so regions will be merged unless keeping them separate offers a savings in cost that warrants the extra cost of the partition

Discussion



(c) "Piece-Regular" ($n = 1024$, SNR $= 200$ dB)



(d) "Piece-Regular" ($n = 1021$, SNR $= 20$ dB)

What do we learn from this particular example?

- Our method is versatile in that it is compatible with signals of arbitrary length $\Rightarrow$ signals whose length are non-dyadic and even prime are perfectly fine
- Even with the addition of significant noise and changing the length from dyadic to prime, the resulting partitions look similar

Outline

Motivation

There are many examples of data in matrix format:

- Images
- Ratings/Reviews
    - Rows → Netflix users
    - Columns → movies
    - $A(i,j) \rightarrow$ user i's rating of movie j on a 1-5 scale
- Spatiotemporal data
    - Rows → sensors
    - Columns → times
    - $A(i,j) \rightarrow$ sensor i's temperature reading at time j

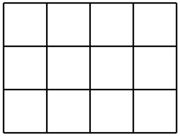
By utilizing graph-based techniques, we can discover and exploit underlying structure in the data for a variety of tasks.

Method

1. Use the matrix data to recursively partition the rows and the columns (explained on next slide)
2. Use the GHWT and best-basis algorithm to analyze the matrix
   i. Analyze along the rows and extract the best basis
   ii. Analyze the row best basis coefficients along the columns and extract the best basis
3. Threshold the expansion coefficients and reconstruct
   - Compare to the tensor Haar basis and, where applicable, to 2-D wavelets

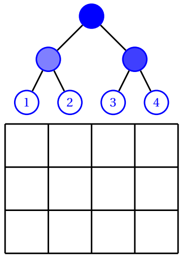
Matrix Partitioning

1. Recursively partition the columns
2. Append row averages on the column clusters
3. Recursively partition the rows (utilizing the appended averages)
4. Append column averages on the row clusters
5. Recursively partition the columns (utilizing the appended averages)
6. Append row averages on the column clusters
7. Recursively partition the rows (utilizing the appended averages)
8. Repeat steps (4)-(7)...



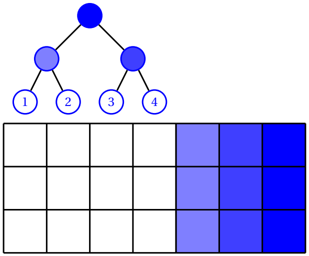
Matrix Partitioning

1. Recursively partition the columns
2. Append row averages on the column clusters
3. Recursively partition the rows (utilizing the appended averages)
4. Append column averages on the row clusters
5. Recursively partition the columns (utilizing the appended averages)
6. Append row averages on the column clusters
7. Recursively partition the rows (utilizing the appended averages)
8. Repeat steps (4)-(7)...



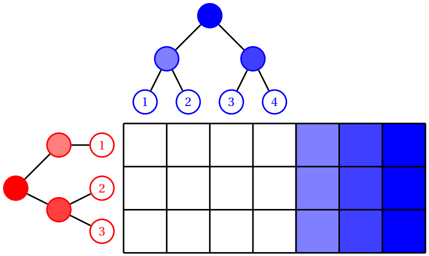
Matrix Partitioning

1. Recursively partition the columns
2. Append row averages on the column clusters
3. Recursively partition the rows (utilizing the appended averages)
4. Append column averages on the row clusters
5. Recursively partition the columns (utilizing the appended averages)
6. Append row averages on the column clusters
7. Recursively partition the rows (utilizing the appended averages)
8. Repeat steps (4)-(7)...



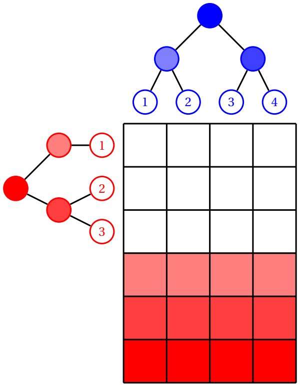
Matrix Partitioning

1. Recursively partition the columns
2. Append row averages on the column clusters
3. Recursively partition the rows (utilizing the appended averages)
4. Append column averages on the row clusters
5. Recursively partition the columns (utilizing the appended averages)
6. Append row averages on the column clusters
7. Recursively partition the rows (utilizing the appended averages)
8. Repeat steps (4)-(7)...



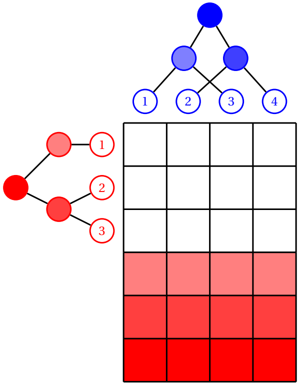
Matrix Partitioning

1. Recursively partition the columns
2. Append row averages on the column clusters
3. Recursively partition the rows (utilizing the appended averages)
4. Append column averages on the row clusters
5. Recursively partition the columns (utilizing the appended averages)
6. Append row averages on the column clusters
7. Recursively partition the rows (utilizing the appended averages)
8. Repeat steps (4)-(7)...



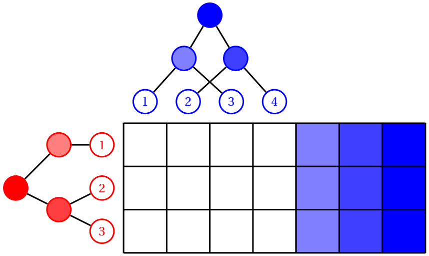
Matrix Partitioning

1. Recursively partition the columns
2. Append row averages on the column clusters
3. Recursively partition the rows (utilizing the appended averages)
4. Append column averages on the row clusters
5. Recursively partition the columns (utilizing the appended averages)
6. Append row averages on the column clusters
7. Recursively partition the rows (utilizing the appended averages)
8. Repeat steps (4)-(7)...



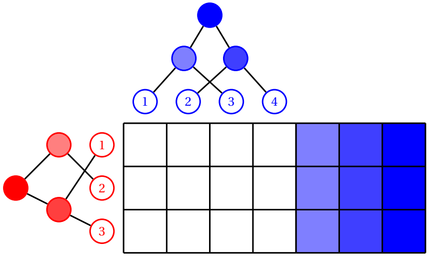
Matrix Partitioning

1. Recursively partition the columns
2. Append row averages on the column clusters
3. Recursively partition the rows (utilizing the appended averages)
4. Append column averages on the row clusters
5. Recursively partition the columns (utilizing the appended averages)
6. Append row averages on the column clusters
7. Recursively partition the rows (utilizing the appended averages)
8. Repeat steps (4)-(7)...



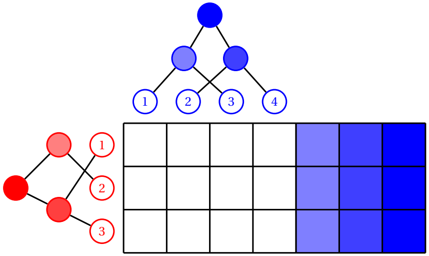
Matrix Partitioning

1. Recursively partition the columns
2. Append row averages on the column clusters
3. Recursively partition the rows (utilizing the appended averages)
4. Append column averages on the row clusters
5. Recursively partition the columns (utilizing the appended averages)
6. Append row averages on the column clusters
7. Recursively partition the rows (utilizing the appended averages)
8. Repeat steps (4)-(7)...



Matrix Partitioning

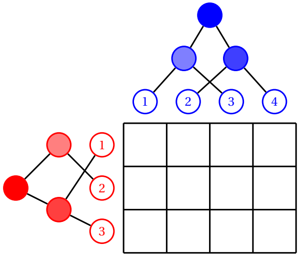
1. Recursively partition the columns
2. Append row averages on the column clusters
3. Recursively partition the rows (utilizing the appended averages)
4. Append column averages on the row clusters
5. Recursively partition the columns (utilizing the appended averages)
6. Append row averages on the column clusters
7. Recursively partition the rows (utilizing the appended averages)
8. Repeat steps (4)-(7)...



Matrix Partitioning

1. Recursively partition the columns
2. Append row averages on the column clusters
3. Recursively partition the rows (utilizing the appended averages)
4. Append column averages on the row clusters
5. Recursively partition the columns (utilizing the appended averages)
6. Append row averages on the column clusters
7. Recursively partition the rows (utilizing the appended averages)
8. Repeat steps (4)-(7)...

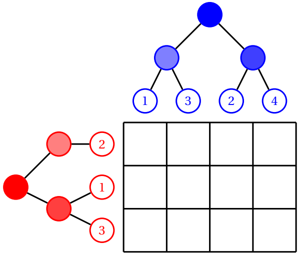
Result: recursive partitioning trees on the rows and columns.



Matrix Partitioning

1. Recursively partition the columns
2. Append row averages on the column clusters
3. Recursively partition the rows (utilizing the appended averages)
4. Append column averages on the row clusters
5. Recursively partition the columns (utilizing the appended averages)
6. Append row averages on the column clusters
7. Recursively partition the rows (utilizing the appended averages)
8. Repeat steps (4)-(7)...

Result: recursive partitioning trees on the rows and columns.



See also: R. R. Coifman & M. Gavish, "Harmonic analysis of digital data bases," in: *Wavelets and Multiscale Analysis* (J. Cohen & A. I. Zayed, eds.), Birkhäuser, New York, pp. 161–197, 2011.

Example 1

Dataset: the Science News database (1047×1000)

- Rows → article abstracts from 8 fields: Anthropology; Astronomy; Behavioral Sciences; Earth Sciences; Life Sciences; Math & CS; Medicine; Physics
- Columns → (appropriately chosen) words
- $A(i, j)$ → the number of times word j appears in abstract i

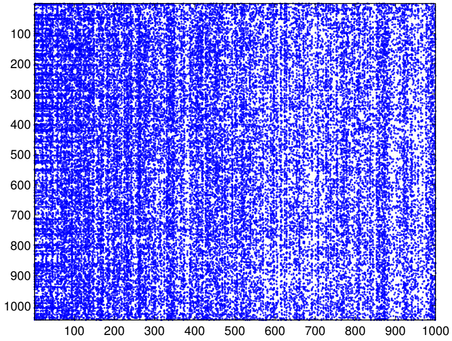


Figure: Science News database (original order).

Example 1

Dataset: the Science News database (1047×1000)

- Rows → article abstracts from 8 fields: Anthropology; Astronomy; Behavioral Sciences; Earth Sciences; Life Sciences; Math & CS; Medicine; Physics
- Columns → (appropriately chosen) words
- $A(i, j)$ → the number of times word j appears in abstract i

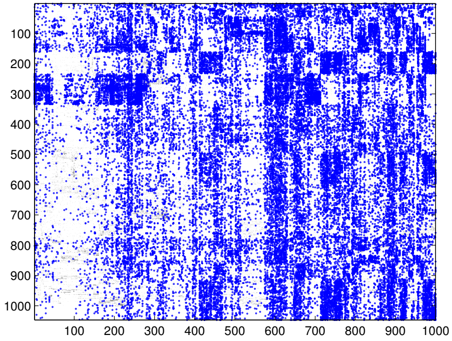


Figure: Science News database (reordered rows and columns).

Example 1

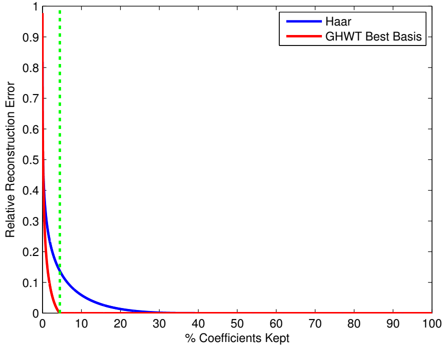


Figure: Haar basis vs. GHWT best basis approximation results. The horizontal line denotes the percentage of nonzero entries in the matrix (**4.5%**).

- Cost functional: 1-norm
- Total number of orthonormal bases searched: $> 10^{300}$
- The best basis performs *much* better than the Haar basis!
- **39.1%** of the Haar coefficients must be kept to achieve perfect reconstruction

$\Rightarrow$ The Haar basis could not efficiently capture the underlying structure of this Science News dataset under the current matrix partitioning strategy (including that of Coifman and Gavish)!

Example 1: Top 3 Basis Functions

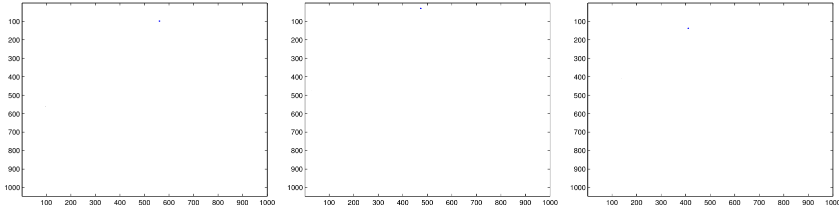
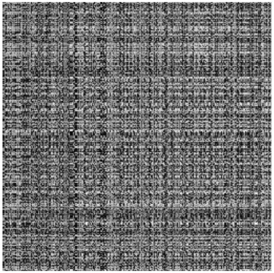
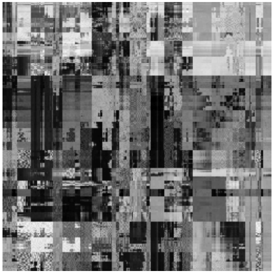


Figure: The support of these basis functions is very localized, i.e., only a few nonzero entries.

Example 2

Dataset: the "Barbara" image with the rows and columns shuffled.





- **Left:** the original Barbara image
- **Middle:** the shuffled Barbara image
- **Right:** the shuffled image reordered according to the recursive partitioning

Example 2

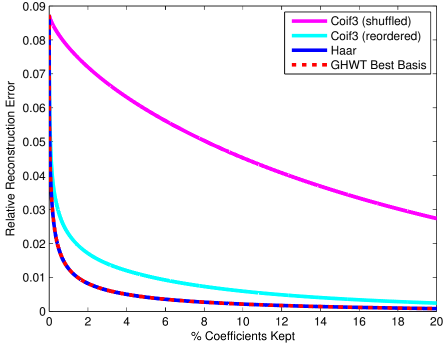
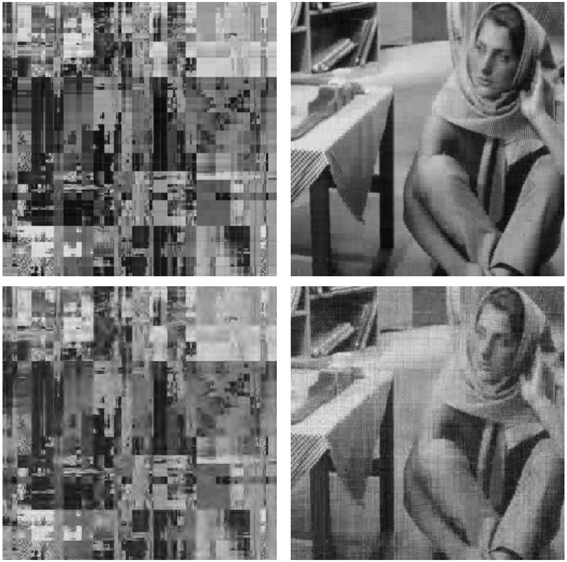


Figure: Approximation results. For "Coif3 (shuffled)" the shuffled image (middle figure on previous page) was analyzed, and for "Coif3 (reordered)" the reordered image (figure on the right) was analyzed.

- Cost functional: 1-norm
- Total number of orthonormal bases searched: 1.36×10^{87}
- The GHWT best basis *exactly matches* the Haar basis
    - As long as the cost functional is chosen correctly for the task at hand (e.g., approximation, denoising, etc.), the GHWT best basis will always perform at least as well as the Haar basis
- The GHWT best basis (Haar basis) performs much better than the Coiflet, which yields a fixed basis that cannot account for the geometry of the data

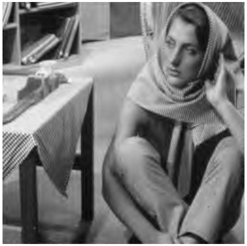
Example 2



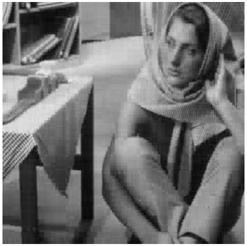
Reconstructions with 10% of the coefficients for the GHWT best basis / Haar basis (top row) and the Coiflet on the reordered image (bottom row).

Example 2 . . .

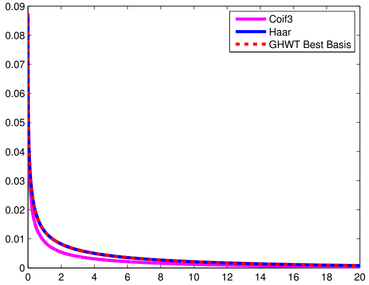
Of course, the Coiflet performs great when the image is not shuffled.



(a) Coiflet reconstruction (10% coefficients kept)



(b) GHWT reconstruction (10% coefficients kept)



(c) Relative Reconstruction Error

But as seen on the previous slides, the GHWT excels when the structure of the matrix data is not known a priori.

Discussion

- Originally developed for signals on graphs, here we have shown the effectiveness of the GHWT for analyzing matrix data
- The GHWT best-basis algorithm searches over an immense number of orthonormal bases, including the graph Haar basis
- When selected using an appropriate cost functional, the GHWT best basis equals or outperforms the graph Haar basis
  - In Example 1, the best-basis algorithm found a basis that performed much better than the Haar basis
  - In Example 2, the best-basis algorithm chose the Haar basis as the best basis
- This demonstrates the importance/advantage of *data-adaptive basis dictionary* from which one can select the most suitable basis for one's task at hand!
- Still need to examine the *meaning of the best basis vectors*, e.g., what combinations of words and articles are well captured by the top basis vectors selected as the best basis?

Outline

Summary

- Although graph Laplacian eigenvectors have been popular as replacement of the Fourier (or DCT) basis on a graph, the analogy takes us only so far due to their sensitivity to the geometry and topology of underlying graphs.
- We developed multiscale basis dictionaries on graphs and networks: *HGLET* and *GHWT*. We also developed a corresponding *best-basis algorithm*.
- The HGLET is a generalization of *Hierarchical Block Discrete Cosine Transforms* originally developed for regularly-sampled signals and images.
- The GHWT is a generalization of the *Haar-Walsh Wavelet Packet Transforms*.
- Both of these transforms allow us to choose an orthonormal basis suitable for the task at hand: approximation, classification, regression, matrix data analysis, . . .
- They are also useful for regularly-sampled signals, e.g., can deal with signals of non-dyadic length; adaptive signal segmentation, . . .
- Developing harmonic analysis tools for directed graphs will be challenging yet important $\Longrightarrow$ our idea: use *integral operator/distance matrix + SVD* instead of *differential operator/graph Laplacian matrix + EIG* (with Eugene Shvarts)
- Still many things to do: generalization to *image* segmentation; better *quantization* strategies for MDL computation; . . .

References

Laplacian Eigenfunction Resource Page
http://www.math.ucdavis.edu/~saito/lapeig/ contains:

- My Course Note (elementary) on "Laplacian Eigenfunctions: Theory, Applications, and Computations"
- My Course Slides on "Harmonic Analysis on Graphs and Networks"
- Talk slides of the minisymposia on Laplacian Eigenfunctions at: ICIAM 2007, Zürich (Organizers: NS, Mauro Maggioni); SIAM Imaging Science Conference 2008, San Diego (Organizers: NS, Xiaomin Huo); IPAM 5-day Workshop 2009, UCLA (Organizers: Peter Jones, Denis Grebenkov, NS); SIAM Annual Meeting 2013, San Diego (Organizers: Chiu-Yen Kao, Braxton Osting, NS); BIRS 5-day Workshop 2015, Banff (Organizers: Peter Jones, Denis Grebenkov, NS).

Jeff Irion disseminates the codes for HGLET/GHWT and related tools at
https://github.com/JeffLIrion/MTSG_Toolbox

References

Laplacian Eigenfunction Resource Page
http://www.math.ucdavis.edu/~saito/lapeig/ contains:

- My Course Note (elementary) on "Laplacian Eigenfunctions: Theory, Applications, and Computations"
- My Course Slides on "Harmonic Analysis on Graphs and Networks"
- Talk slides of the minisymposia on Laplacian Eigenfunctions at: ICIAM 2007, Zürich (Organizers: NS, Mauro Maggioni); SIAM Imaging Science Conference 2008, San Diego (Organizers: NS, Xiaomin Huo); IPAM 5-day Workshop 2009, UCLA (Organizers: Peter Jones, Denis Grebenkov, NS); SIAM Annual Meeting 2013, San Diego (Organizers: Chiu-Yen Kao, Braxton Osting, NS); BIRS 5-day Workshop 2015, Banff (Organizers: Peter Jones, Denis Grebenkov, NS).

Jeff Irion disseminates the codes for HGLET/GHWT and related tools at
https://github.com/JeffLIrion/MTSG_Toolbox

The following articles (and the other related ones) are available at http://www.math.ucdavis.edu/~saito/publications/

- N. Saito: "Data analysis and representation using eigenfunctions of Laplacian on a general domain," *Applied & Computational Harmonic Analysis*, vol. 25, no. 1, pp. 68–97, 2008.
- N. Saito & E. Woei: "Analysis of neuronal dendrite patterns using eigenvalues of graph Laplacians," *JSIAM Letters*, vol. 1, pp. 13–16, 2009.
- Y. Nakatsukasa, N. Saito, & E. Woei: "Mysteries around graph Laplacian eigenvalue 4," *Linear Algebra and its Applications*, vol. 438, no. 8, pp. 3231–3246, 2013.
- J. Irion & N. Saito: "Hierarchical graph Laplacian eigen transforms," *JSIAM Letters*, vol. 6, pp. 21–24, 2014.
- J. Irion & N. Saito: "The generalized Haar-Walsh transform," *Proc. 2014 IEEE Workshop on Statistical Signal Processing*, pp. 488-491, 2014.
- N. Saito & E. Woei: "Tree simplification and the 'plateaux' phenomenon of graph Laplacian eigenvalues," *Linear Algebra and its Applications*, vol. 481, pp. 263–279, 2015.

Thank you very much for your attention!