

MAT 271: Applied & Computational Harmonic Analysis
Lecture 20: *Multiscale Basis Dictionaries on Graphs and
Networks*

Naoki Saito (with help from Jeff Irion)

Department of Mathematics
University of California, Davis

March 13, 2014

- 1 Introductory Remarks
- 2 Motivations: Why Graphs?
- 3 Background
 - Basic Graph Theory Terminology
 - Graph Laplacians
 - Graph Partitioning via Spectral Clustering
- 4 Multiscale Basis Dictionaries
 - Hierarchical Graph Laplacian Eigen Transform (HGLET)
 - Generalized Haar-Walsh Transform (GHWT)
- 5 Best-Basis Algorithm for HGLET & GHWT
- 6 Approximation Experiments
- 7 Summary and Future Work

- For much more details of this part of lecture, please check my course website on “Harmonic Analysis on Graphs & Networks”: <http://www.math.ucdavis.edu/~saito/courses/HarmGraph/> as well as my two short articles with Jeff Irion at <http://www.math.ucdavis.edu/~saito/publications/>.
- We rely on the so-called **graph Laplacians** to construct our multiscale basis dictionaries. Some good references on graph Laplacian **eigenvalues** are:
 - R. B. Bapat: *Graphs and Matrices*, Universitext, Springer, 2010.
 - A. E. Brouwer & W. H. Haemers: *Spectra of Graphs*, Springer, 2012.
 - F. R. K. Chung: *Spectral Graph Theory*, Amer. Math. Soc., 1997.
 - D. Cvetković, P. Rowlinson, & S. Simić: *An Introduction to the Theory of Graph Spectra*, Vol. 75, London Mathematical Society Student Texts, Cambridge Univ. Press, 2010.
 - D. Spielman: “Spectral graph theory,” in *Combinatorial Scientific Computing* (O. Schenk, ed.), Chap. 18, pp. 495–524, CRC Press, 2012.
- As for the graph Laplacian *eigenfunctions*, there are not too many books (although there may be many papers); one of the good books is
 - T. Bıyıkođlu, J. Leydold, & P. F. Stadler, *Laplacian Eigenvectors of Graphs*, Lecture Notes in Mathematics, vol. 1915, Springer, 2007.

- 1 Introductory Remarks
- 2 Motivations: Why Graphs?**
- 3 Background
 - Basic Graph Theory Terminology
 - Graph Laplacians
 - Graph Partitioning via Spectral Clustering
- 4 Multiscale Basis Dictionaries
 - Hierarchical Graph Laplacian Eigen Transform (HGLET)
 - Generalized Haar-Walsh Transform (GHWT)
- 5 Best-Basis Algorithm for HGLET & GHWT
- 6 Approximation Experiments
- 7 Summary and Future Work

Motivations: Why Graphs?

- 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?

- 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?

- 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?

- 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?

- 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?

- 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?

- 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?

- 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?

- **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.

Motivations: Why Graphs?

- **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.

Motivations: Why Graphs?

- **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.

An Example of Sensor Networks

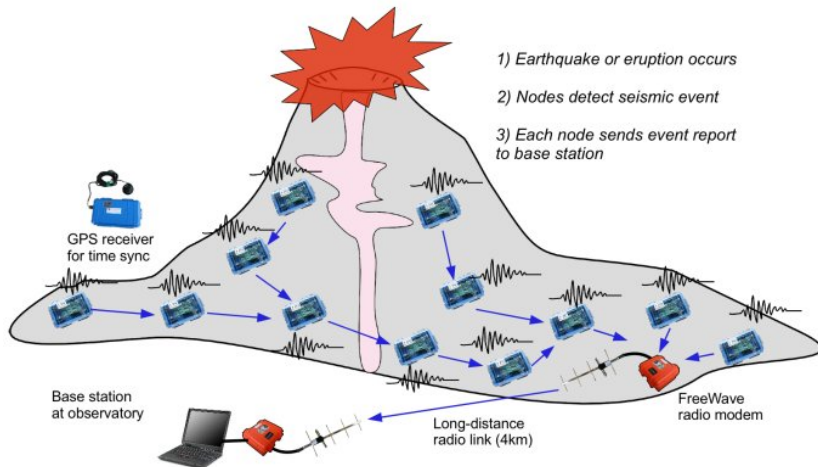


Figure : Volcano monitoring sensor network architecture of Harvard Sensor Networks Lab

An Example of Social Networks

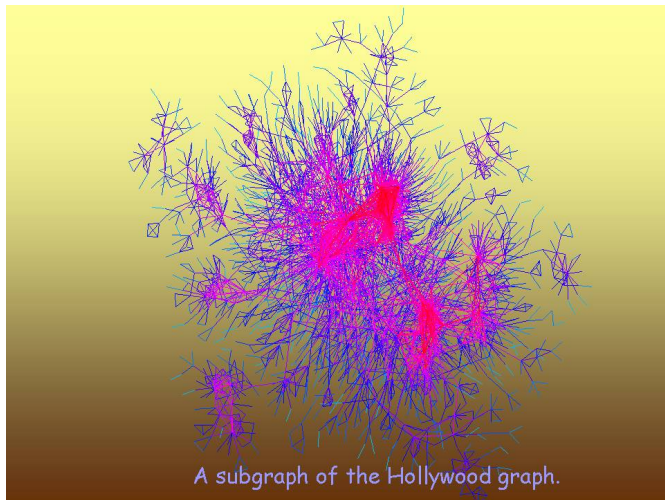


Figure : Through the courtesy of Prof. Fan Chung, UC San Diego

An Example of Biological Networks

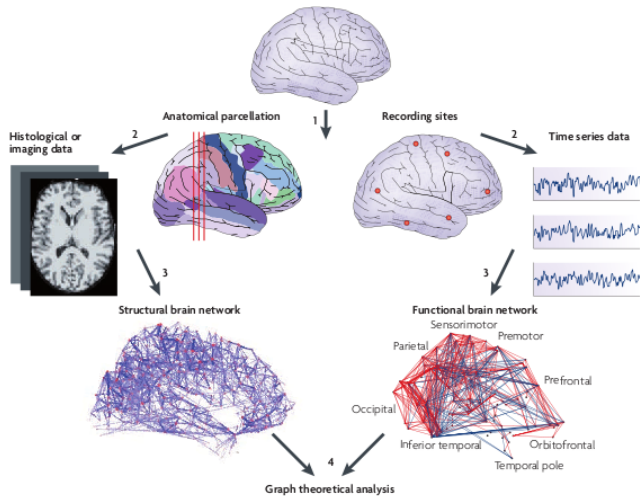
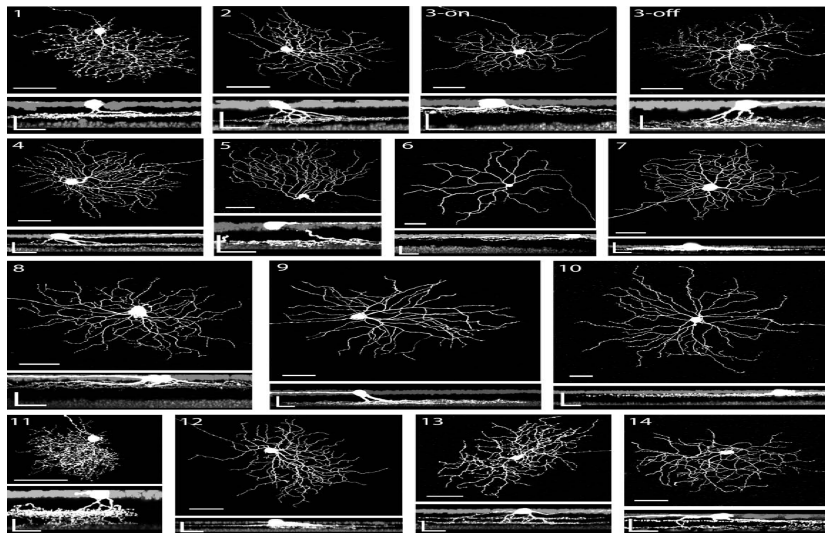
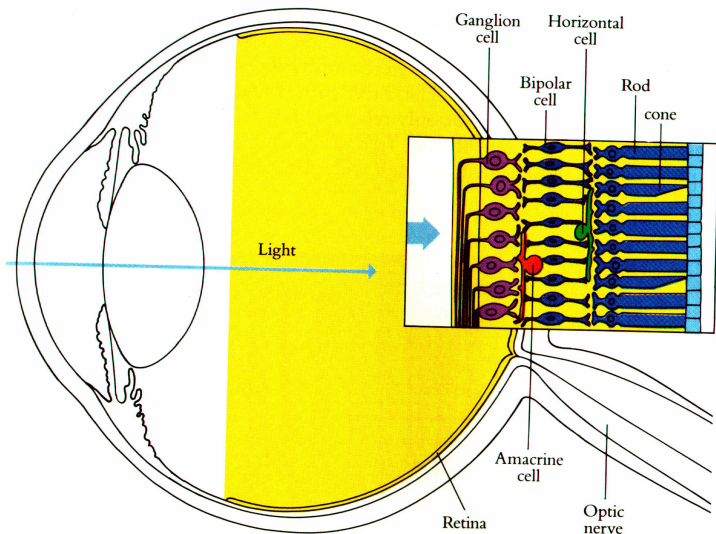


Figure : From E. Bullmore and O. Sporns, *Nature Reviews Neuroscience*, vol. 10, pp.186–198, Mar. 2009.

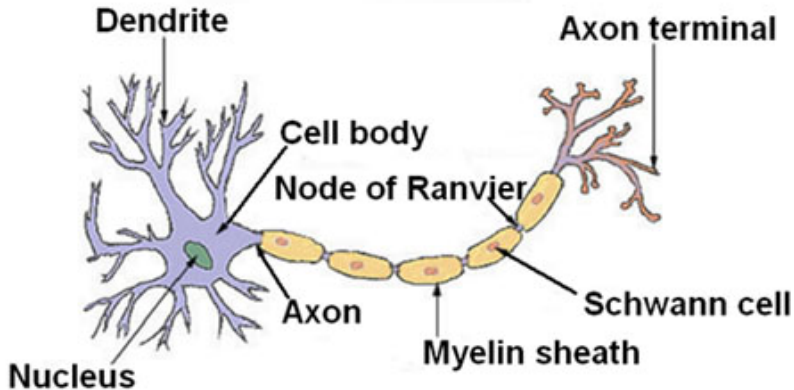
Another Biological Example: Retinal Ganglion Cells



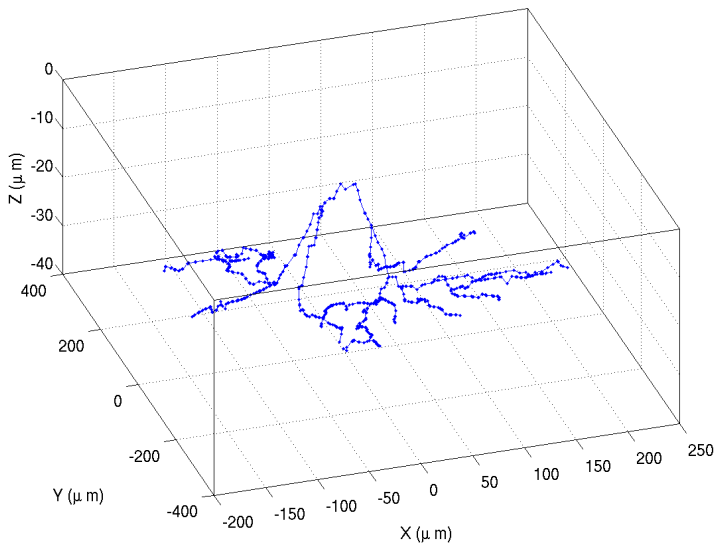
Retinal Ganglion Cells (D. Hubel: *Eye, Brain, & Vision*, '95)

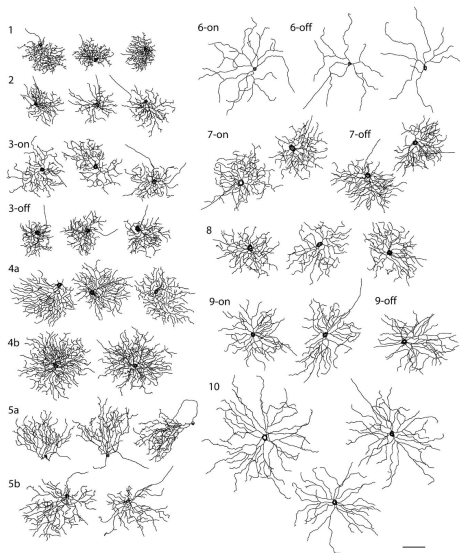
A Typical Neuron (from Wikipedia)

Structure of a Typical Neuron



Mouse's RGC as a Graph



Clustering using Features Derived by Neurolucida[®]

Representing a Regular Image as a Graph

often turns out to be quite useful for various purposes. In particular, **Nonlocal Means Denoising Algorithm** of Buades-Coll-Morel is quite impressive.

- Construct a graph each of whose vertices represents $k \times k$ patch of a given image (k may be 3, 5, ..., etc.) So each vertex represents a point in \mathbb{R}^{k^2} .
- Connect every pair of vertices with the weight $W_{ij} = \exp(-\|\text{patch}_i - \text{patch}_j\|^2 / \epsilon^2)$ with *appropriately chosen* scale parameter $\epsilon > 0$.
- Compute the weighted average of the center pixel of each patch using the normalized weights $W_{ij} / \sum_l W_{il}$. More precisely, the average of the center of the i th patch, $\bar{c}_i = \sum_j W_{ij} c_j / \sum_l W_{il}$.
- See also an interesting work by Daitch-Kelner-Spielman: "Fitting a Graph to Vector Data," *Proc. 26th Intern. Conf. Machine Learning*, 2009.

Representing a Regular Image as a Graph

often turns out to be quite useful for various purposes. In particular, **Nonlocal Means Denoising Algorithm** of Buades-Coll-Morel is quite impressive.

- Construct a graph each of whose vertices represents $k \times k$ patch of a given image (k may be 3, 5, ..., etc.) So each vertex represents a point in \mathbb{R}^{k^2} .
- Connect every pair of vertices with the weight $W_{ij} = \exp(-\|\text{patch}_i - \text{patch}_j\|^2 / \epsilon^2)$ with *appropriately chosen* scale parameter $\epsilon > 0$.
- Compute the weighted average of the center pixel of each patch using the normalized weights $W_{ij} / \sum_l W_{il}$. More precisely, the average of the center of the i th patch, $\bar{c}_i = \sum_j W_{ij} c_j / \sum_l W_{il}$.
- See also an interesting work by Daitch-Kelner-Spielman: "Fitting a Graph to Vector Data," *Proc. 26th Intern. Conf. Machine Learning*, 2009.

Representing a Regular Image as a Graph

often turns out to be quite useful for various purposes. In particular, **Nonlocal Means Denoising Algorithm** of Buades-Coll-Morel is quite impressive.

- Construct a graph each of whose vertices represents $k \times k$ patch of a given image (k may be 3, 5, ..., etc.) So each vertex represents a point in \mathbb{R}^{k^2} .
- Connect every pair of vertices with the weight $W_{ij} = \exp(-\|\text{patch}_i - \text{patch}_j\|^2 / \epsilon^2)$ with *appropriately chosen* scale parameter $\epsilon > 0$.
- Compute the weighted average of the center pixel of each patch using the normalized weights $W_{ij} / \sum_l W_{il}$. More precisely, the average of the center of the i th patch, $\bar{c}_i = \sum_j W_{ij} c_j / \sum_l W_{il}$.
- See also an interesting work by Daitch-Kelner-Spielman: "Fitting a Graph to Vector Data," *Proc. 26th Intern. Conf. Machine Learning*, 2009.

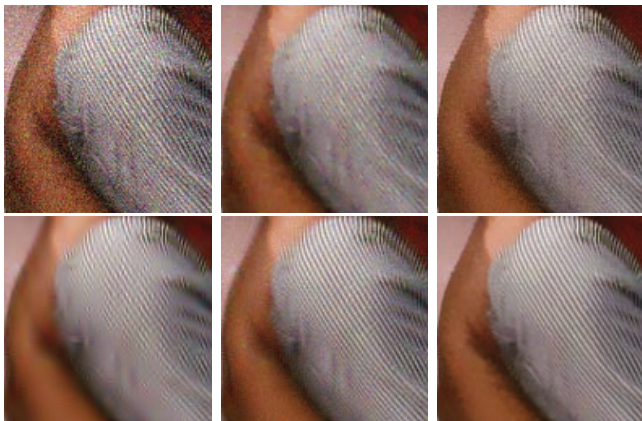
Representing a Regular Image as a Graph

often turns out to be quite useful for various purposes. In particular, **Nonlocal Means Denoising Algorithm** of Buades-Coll-Morel is quite impressive.

- Construct a graph each of whose vertices represents $k \times k$ patch of a given image (k may be 3, 5, ..., etc.) So each vertex represents a point in \mathbb{R}^{k^2} .
- Connect every pair of vertices with the weight $W_{ij} = \exp(-\|\text{patch}_i - \text{patch}_j\|^2 / \epsilon^2)$ with *appropriately chosen* scale parameter $\epsilon > 0$.
- Compute the weighted average of the center pixel of each patch using the normalized weights $W_{ij} / \sum_l W_{il}$. More precisely, the average of the center of the i th patch, $\bar{c}_i = \sum_j W_{ij} c_j / \sum_l W_{il}$.
- See also an interesting work by Daitch-Kelner-Spielman: "Fitting a Graph to Vector Data," *Proc. 26th Intern. Conf. Machine Learning*, 2009.

From: A. Buades, B. Coll, and J.-M. Morel, *SIAM Review*, vol. 52, no. 1, pp. 113–147, 2010.

Noisy Image; Total Variation Denoising; Neighborhood Filter



Trans. Inv. Wavelets; Empirical Wiener; Nonlocal Means

Motivations: Multiscale Basis Dictionary on Graphs

Wavelets

- Have been quite successful on regular domains
- Have been extended to irregular domains \Rightarrow “2nd Generation Wavelets”

For example:

- Hammond, Vandergheynst, and Gribonval (2011): wavelets via spectral graph theory
- Coifman and Maggioni (2006): diffusion wavelets
 \Rightarrow Bremer *et al.* (2006): diffusion wavelet packets

Key difficulty: The notion of *frequency* is ill-defined on graphs \Rightarrow The Fourier transform is not properly defined on graphs

Common strategy: Develop wavelet-*like* multiscale transforms

Key Idea: Use of the **graph Laplacian eigenvectors** as the substitution of the Fourier basis

Motivations: Multiscale Basis Dictionary on Graphs

Wavelets

- Have been quite successful on regular domains
- Have been extended to irregular domains \Rightarrow “2nd Generation Wavelets”

For example:

- Hammond, Vandergheynst, and Gribonval (2011): wavelets via spectral graph theory
- Coifman and Maggioni (2006): diffusion wavelets
 \Rightarrow Bremer *et al.* (2006): diffusion wavelet packets

Key difficulty: The notion of *frequency* is ill-defined on graphs \Rightarrow The Fourier transform is not properly defined on graphs

Common strategy: Develop wavelet-*like* multiscale transforms

Key Idea: Use of the **graph Laplacian eigenvectors** as the substitution of the Fourier basis

Motivations: Multiscale Basis Dictionary on Graphs

Wavelets

- Have been quite successful on regular domains
- Have been extended to irregular domains \Rightarrow “2nd Generation Wavelets”

For example:

- Hammond, Vandergheynst, and Gribonval (2011): wavelets via spectral graph theory
- Coifman and Maggioni (2006): diffusion wavelets
 \Rightarrow Bremer *et al.* (2006): diffusion wavelet packets

Key difficulty: The notion of *frequency* is ill-defined on graphs \Rightarrow The Fourier transform is not properly defined on graphs

Common strategy: Develop wavelet-*like* multiscale transforms

Key Idea: Use of the **graph Laplacian eigenvectors** as the substitution of the Fourier basis

Motivations: Multiscale Basis Dictionary on Graphs

Wavelets

- Have been quite successful on regular domains
- Have been extended to irregular domains \Rightarrow “2nd Generation Wavelets”

For example:

- Hammond, Vandergheynst, and Gribonval (2011): wavelets via spectral graph theory
- Coifman and Maggioni (2006): diffusion wavelets
 \Rightarrow Bremer *et al.* (2006): diffusion wavelet packets

Key difficulty: The notion of *frequency* is ill-defined on graphs \Rightarrow The Fourier transform is not properly defined on graphs

Common strategy: Develop wavelet-*like* multiscale transforms

Key Idea: Use of the **graph Laplacian eigenvectors** as the substitution of the Fourier basis

Motivations: Multiscale Basis Dictionary on Graphs

Wavelets

- Have been quite successful on regular domains
- Have been extended to irregular domains \Rightarrow “2nd Generation Wavelets”

For example:

- Hammond, Vandergheynst, and Gribonval (2011): wavelets via spectral graph theory
- Coifman and Maggioni (2006): diffusion wavelets
 \Rightarrow Bremer *et al.* (2006): diffusion wavelet packets

Key difficulty: The notion of *frequency* is ill-defined on graphs \Rightarrow The Fourier transform is not properly defined on graphs

Common strategy: Develop wavelet-*like* multiscale transforms

Key Idea: Use of the **graph Laplacian eigenvectors** as the substitution of the Fourier basis

Goals

- Develop and implement multiscale transforms for data on graphs and networks; in particular, build *multiscale basis dictionaries* on graphs.
- Investigate their usefulness for a variety of applications including approximation, denoising, classification, and regression on graphs.
- In this lecture, we will focus on how to construct such dictionaries on graphs and demonstrate their usefulness for data approximation on graphs.

Goals

- Develop and implement multiscale transforms for data on graphs and networks; in particular, build *multiscale basis dictionaries* on graphs.
- Investigate their usefulness for a variety of applications including approximation, denoising, classification, and regression on graphs.
- In this lecture, we will focus on how to construct such dictionaries on graphs and demonstrate their usefulness for data approximation on graphs.

Goals

- Develop and implement multiscale transforms for data on graphs and networks; in particular, build *multiscale basis dictionaries* on graphs.
- Investigate their usefulness for a variety of applications including approximation, denoising, classification, and regression on graphs.
- In this lecture, we will focus on how to construct such dictionaries on graphs and demonstrate their usefulness for data approximation on graphs.

- 1 Introductory Remarks
- 2 Motivations: Why Graphs?
- 3 Background**
 - **Basic Graph Theory Terminology**
 - Graph Laplacians
 - Graph Partitioning via Spectral Clustering
- 4 Multiscale Basis Dictionaries
 - Hierarchical Graph Laplacian Eigen Transform (HGLET)
 - Generalized Haar-Walsh Transform (GHWT)
- 5 Best-Basis Algorithm for HGLET & GHWT
- 6 Approximation Experiments
- 7 Summary and Future Work

Definitions and Notation

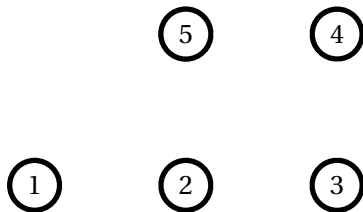
Let G be a **graph**.

- $V = V(G) = \{v_1, \dots, v_N\}$ is the set of **vertices**.
- $E = E(G) = \{e_1, \dots, e_N\}$ is the set of **edges**, where $e_k = (v_i, v_j)$ represents an edge (or line segment) connecting between adjacent vertices v_i, v_j for some $1 \leq i, j \leq N$.
- $W = W(G) \in \mathbb{R}^{N \times N}$ is the **weight matrix**, where w_{ij} denotes the edge weight between vertices i and j .

Definitions and Notation

Let G be a **graph**.

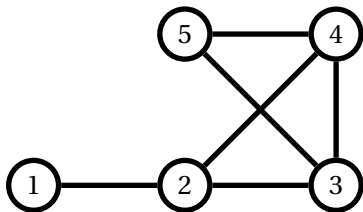
- $V = V(G) = \{v_1, \dots, v_N\}$ is the set of **vertices**.
- $E = E(G) = \{e_1, \dots, e_N\}$ is the set of **edges**, where $e_k = (v_i, v_j)$ represents an edge (or line segment) connecting between adjacent vertices v_i, v_j for some $1 \leq i, j \leq N$.
- $W = W(G) \in \mathbb{R}^{N \times N}$ is the **weight matrix**, where w_{ij} denotes the edge weight between vertices i and j .



Definitions and Notation

Let G be a **graph**.

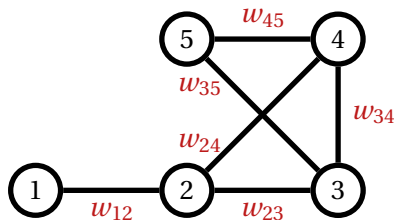
- $V = V(G) = \{v_1, \dots, v_N\}$ is the set of **vertices**.
- $E = E(G) = \{e_1, \dots, e_N\}$ is the set of **edges**, where $e_k = (v_i, v_j)$ represents an edge (or line segment) connecting between adjacent vertices v_i, v_j for some $1 \leq i, j \leq N$.
- $W = W(G) \in \mathbb{R}^{N \times N}$ is the **weight matrix**, where w_{ij} denotes the edge weight between vertices i and j .



Definitions and Notation

Let G be a **graph**.

- $V = V(G) = \{v_1, \dots, v_N\}$ is the set of **vertices**.
- $E = E(G) = \{e_1, \dots, e_N\}$ is the set of **edges**, where $e_k = (v_i, v_j)$ represents an edge (or line segment) connecting between adjacent vertices v_i, v_j for some $1 \leq i, j \leq N$.
- $W = W(G) \in \mathbb{R}^{N \times N}$ 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

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

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 v_i and v_j , e.g., if $v_i \sim v_j$, then

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

where $\text{dist}(\cdot, \cdot)$ 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

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

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 v_i and v_j , e.g., if $v_i \sim v_j$, then

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

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

Our Assumptions

In this lecture, 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.

Our Assumptions

In this lecture, 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.

Our Assumptions

In this lecture, 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.

Our Assumptions

In this lecture, 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.

- 1 Introductory Remarks
- 2 Motivations: Why Graphs?
- 3 Background**
 - Basic Graph Theory Terminology
 - Graph Laplacians**
 - Graph Partitioning via Spectral Clustering
- 4 Multiscale Basis Dictionaries
 - Hierarchical Graph Laplacian Eigen Transform (HGLET)
 - Generalized Haar-Walsh Transform (GHWT)
- 5 Best-Basis Algorithm for HGLET & GHWT
- 6 Approximation Experiments
- 7 Summary and Future Work

Graph Laplacians

$$\left\{ \begin{array}{ll} W(G) = (w_{ij}) & \text{the weight matrix} \\ D(G) := \text{diag}(d_{v_1}, \dots, d_{v_n}) & \text{the degree matrix, where } d_{v_i} := \sum_{j=1}^N w_{ij}. \\ L(G) := D(G) - W(G) & \text{the Laplacian matrix} \end{array} \right.$$

We have:

- sorted eigenvalues $0 = \lambda_0 < \lambda_1 \leq \dots \leq \lambda_{N-1}$
- associated eigenvectors $\phi_0, \phi_1, \dots, \phi_{N-1}$

The eigenvectors form a basis for \mathbb{R}^N . In particular:

- since L is symmetric, the eigenvectors form an orthonormal basis
- $\phi_0 = \mathbf{1}/\sqrt{N}$

Graph Laplacians

$$\left\{ \begin{array}{ll} W(G) = (w_{ij}) & \text{the weight matrix} \\ D(G) := \text{diag}(d_{v_1}, \dots, d_{v_n}) & \text{the degree matrix, where } d_{v_i} := \sum_{j=1}^N w_{ij}. \\ L(G) := D(G) - W(G) & \text{the Laplacian matrix} \end{array} \right.$$

We have:

- sorted eigenvalues $0 = \lambda_0 < \lambda_1 \leq \dots \leq \lambda_{N-1}$
- associated eigenvectors $\boldsymbol{\phi}_0, \boldsymbol{\phi}_1, \dots, \boldsymbol{\phi}_{N-1}$

The eigenvectors form a basis for \mathbb{R}^N . In particular:

- since L is symmetric, the eigenvectors form an orthonormal basis
- $\boldsymbol{\phi}_0 = \mathbf{1}/\sqrt{N}$

Graph Laplacians

$$\begin{cases} W(G) = (w_{ij}) & \text{the weight matrix} \\ D(G) := \text{diag}(d_{v_1}, \dots, d_{v_n}) & \text{the degree matrix, where } d_{v_i} := \sum_{j=1}^N w_{ij}. \\ L(G) := D(G) - W(G) & \text{the Laplacian matrix} \end{cases}$$

We have:

- sorted eigenvalues $0 = \lambda_0 < \lambda_1 \leq \dots \leq \lambda_{N-1}$
- associated eigenvectors $\boldsymbol{\phi}_0, \boldsymbol{\phi}_1, \dots, \boldsymbol{\phi}_{N-1}$

The eigenvectors form a basis for \mathbb{R}^N . In particular:

- since L is symmetric, the eigenvectors form an orthonormal basis
- $\boldsymbol{\phi}_0 = \mathbf{1}/\sqrt{N}$

Why Graph Laplacians?

- Let $f \in \mathbb{R}^N$. Then

$$Lf(v_i) = d_{v_i}f(v_i) - \sum_{j \neq i} w_{ij}f(v_j).$$

This is a generalization of *the finite difference approximation to the Laplace operator*.

- After all, *sines (cosines)* are the *eigenfunctions of the Laplacian* on the rectangular domain with Dirichlet (Neumann) boundary conditions.
- Hence, the expansion of data measured at the vertices w.r.t. the eigenvectors of a graph Laplacian can be viewed as *Fourier (or spectral) analysis of the data on that graph*.

Why Graph Laplacians?

- Let $f \in \mathbb{R}^N$. Then

$$Lf(v_i) = d_{v_i}f(v_i) - \sum_{j \neq i} w_{ij}f(v_j).$$

This is a generalization of *the finite difference approximation to the Laplace operator*.

- After all, *sines (cosines)* are the **eigenfunctions of the Laplacian** on the rectangular domain with Dirichlet (Neumann) boundary conditions.
- Hence, the expansion of data measured at the vertices w.r.t. the eigenvectors of a graph Laplacian can be viewed as **Fourier (or spectral) analysis of the data on that graph**.

Why Graph Laplacians?


- Let $f \in \mathbb{R}^N$. Then

$$Lf(v_i) = d_{v_i}f(v_i) - \sum_{j \neq i} w_{ij}f(v_j).$$

This is a generalization of *the finite difference approximation to the Laplace operator*.

- After all, *sines (cosines)* are the **eigenfunctions of the Laplacian** on the rectangular domain with Dirichlet (Neumann) boundary conditions.
- Hence, the expansion of data measured at the vertices w.r.t. the eigenvectors of a graph Laplacian can be viewed as **Fourier (or spectral) analysis of the data on that graph**.

A Simple Yet Important Example: A Path Graph




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

The eigenvectors of this matrix are exactly the *DCT Type II* basis vectors used for the JPEG image compression standard! (See e.g., Strang, SIAM Review, 1999).

- $\lambda_k = 2 - 2 \cos(\pi k / N) = 4 \sin^2(\pi k / 2N)$, $k = 0, 1, \dots, N - 1$.
- $\phi_k(\ell) = \sqrt{2/N} \cos(\pi k(\ell + \frac{1}{2}) / N)$, $k, \ell = 0, 1, \dots, N - 1$.
- λ (eigenvalue) is a monotonic function w.r.t. k (frequency). However, for general graphs, λ does not have a simple relationship with k .

A Simple Yet Important Example: A Path Graph



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

The eigenvectors of this matrix are exactly the *DCT Type II* basis vectors used for the JPEG image compression standard! (See e.g., Strang, SIAM Review, 1999).

- $\lambda_k = 2 - 2 \cos(\pi k/N) = 4 \sin^2(\pi k/2N)$, $k = 0, 1, \dots, N-1$.
- $\phi_k(\ell) = \sqrt{2/N} \cos(\pi k(\ell + \frac{1}{2})/N)$, $k, \ell = 0, 1, \dots, N-1$.
- λ (eigenvalue) is a monotonic function w.r.t. k (frequency). However, for general graphs, λ does not have a simple relationship with k .

- 1 Introductory Remarks
- 2 Motivations: Why Graphs?
- 3 Background**
 - Basic Graph Theory Terminology
 - Graph Laplacians
 - Graph Partitioning via Spectral Clustering**
- 4 Multiscale Basis Dictionaries
 - Hierarchical Graph Laplacian Eigen Transform (HGLET)
 - Generalized Haar-Walsh Transform (GHWT)
- 5 Best-Basis Algorithm for HGLET & GHWT
- 6 Approximation Experiments
- 7 Summary and Future Work

Goal: split the vertices V into two “good” subsets, X and X^c

Plan: use the signs of the entries in ϕ_1 , which is known as the **Fiedler vector**

Why? Using ϕ_1 to generate X and X^c yields an approximate minimizer of the RatioCut function^{1,2}:

$$\text{RatioCut}(X, X^c) := \frac{\text{cut}(X, X^c)}{|X|} + \frac{\text{cut}(X, X^c)}{|X^c|},$$

where

$$\text{cut}(X, X^c) := \sum_{v_i \in X; v_j \in X^c} W_{ij}$$

- Dividing by the number of nodes ensures that the partitions are of roughly the same size \Rightarrow we do not simply cleave a small number of nodes

¹L. 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.

²We could also use the signs of ϕ_1 for L_{rw} (equivalently, L_{sym}), which yield an approximate minimizer of the popular Normalized Cut function: J. Shi & J. Malik: “Normalized cuts and image segmentation”, *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, no. 8, pp. 888–905, 2000.

Goal: split the vertices V into two “good” subsets, X and X^c

Plan: use the signs of the entries in ϕ_1 , which is known as the **Fiedler vector**

Why? Using ϕ_1 to generate X and X^c yields an approximate minimizer of the RatioCut function^{1,2}:

$$\text{RatioCut}(X, X^c) := \frac{\text{cut}(X, X^c)}{|X|} + \frac{\text{cut}(X, X^c)}{|X^c|},$$

where

$$\text{cut}(X, X^c) := \sum_{v_i \in X; v_j \in X^c} W_{ij}$$

- Dividing by the number of nodes ensures that the partitions are of roughly the same size \Rightarrow we do not simply cleave a small number of nodes

¹L. 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.

²We could also use the signs of ϕ_1 for L_{rw} (equivalently, L_{sym}), which yield an approximate minimizer of the popular Normalized Cut function: J. Shi & J. Malik: “Normalized cuts and image segmentation”, *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, no. 8, pp. 888–905, 2000.

Goal: split the vertices V into two “good” subsets, X and X^c

Plan: use the signs of the entries in ϕ_1 , which is known as the **Fiedler vector**

Why? Using ϕ_1 to generate X and X^c yields an approximate minimizer of the RatioCut function^{1,2}:

$$\text{RatioCut}(X, X^c) := \frac{\text{cut}(X, X^c)}{|X|} + \frac{\text{cut}(X, X^c)}{|X^c|},$$

where

$$\text{cut}(X, X^c) := \sum_{v_i \in X; v_j \in X^c} W_{ij}$$

- Dividing by the number of nodes ensures that the partitions are of roughly the same size \Rightarrow we do not simply cleave a small number of nodes

¹L. 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.

²We could also use the signs of ϕ_1 for L_{rw} (equivalently, L_{sym}), which yield an approximate minimizer of the popular Normalized Cut function: J. Shi & J. Malik: “Normalized cuts and image segmentation”, *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, no. 8, pp. 888–905, 2000.

Goal: split the vertices V into two “good” subsets, X and X^c

Plan: use the signs of the entries in ϕ_1 , which is known as the **Fiedler vector**

Why? Using ϕ_1 to generate X and X^c yields an approximate minimizer of the RatioCut function^{1,2}:

$$\text{RatioCut}(X, X^c) := \frac{\text{cut}(X, X^c)}{|X|} + \frac{\text{cut}(X, X^c)}{|X^c|},$$

where

$$\text{cut}(X, X^c) := \sum_{v_i \in X; v_j \in X^c} W_{ij}$$

- Dividing by the number of nodes ensures that the partitions are of roughly the same size \Rightarrow we do not simply cleave a small number of nodes

¹L. 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.

²We could also use the signs of ϕ_1 for L_{rw} (equivalently, L_{sym}), which yield an approximate minimizer of the popular Normalized Cut function: 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

Let us reformulate the RatioCut minimization problem.

- 1 Define $\mathbf{f} \in \mathbb{R}^N$ as

$$f_i := \begin{cases} \sqrt{\frac{|X^c|}{|X|}} & \text{if } v_i \in X \\ -\sqrt{\frac{|X|}{|X^c|}} & \text{if } v_i \in X^c \end{cases}$$

- 2 The RatioCut problem can be reformulated as

$$\min_{X \subset V} \mathbf{f}^\top L \mathbf{f} \quad \text{subject to } \mathbf{f} \text{ defined as above}$$

Graph Partitioning via Spectral Clustering

Let us reformulate the RatioCut minimization problem.

- 1 Define $\mathbf{f} \in \mathbb{R}^N$ as

$$f_i := \begin{cases} \sqrt{\frac{|X^c|}{|X|}} & \text{if } v_i \in X \\ -\sqrt{\frac{|X|}{|X^c|}} & \text{if } v_i \in X^c \end{cases}$$

- 2 The RatioCut problem can be reformulated as

$$\min_{X \subset V} \mathbf{f}^\top L \mathbf{f} \quad \text{subject to } \mathbf{f} \text{ defined as above}$$

Graph Partitioning via Spectral Clustering

Let us reformulate the RatioCut minimization problem.

- 1 Define $\mathbf{f} \in \mathbb{R}^N$ as

$$f_i := \begin{cases} \sqrt{\frac{|X^c|}{|X|}} & \text{if } v_i \in X \\ -\sqrt{\frac{|X|}{|X^c|}} & \text{if } v_i \in X^c \end{cases}$$

- 2 The RatioCut problem can be reformulated as

$$\min_{X \subset V} \mathbf{f}^T L \mathbf{f} \quad \text{subject to } \mathbf{f} \text{ defined as above}$$

$$\begin{aligned}
\mathbf{f}^\top L \mathbf{f} &= \frac{1}{2} \sum_{i,j=1}^N W_{ij} (f_i - f_j)^2 \\
&= \frac{1}{2} \sum_{\substack{v_i \in X \\ v_j \in X^c}} W_{ij} \left(\sqrt{\frac{|X^c|}{|X|}} + \sqrt{\frac{|X|}{|X^c|}} \right)^2 \\
&\quad + \frac{1}{2} \sum_{\substack{v_i \in X^c \\ v_j \in X}} W_{ij} \left(-\sqrt{\frac{|X^c|}{|X|}} - \sqrt{\frac{|X|}{|X^c|}} \right)^2 \\
&= \text{cut}(X, X^c) \left(\frac{|X^c|}{|X|} + \frac{|X|}{|X^c|} + 2 \right) \\
&= \text{cut}(X, X^c) \left(\frac{|X| + |X^c|}{|X|} + \frac{|X| + |X^c|}{|X^c|} \right) \\
&= |V| \text{RatioCut}(X, X^c)
\end{aligned}$$

Graph Partitioning via Spectral Clustering

Let us reformulate the RatioCut minimization problem.

- 1 Define $\mathbf{f} \in \mathbb{R}^N$ as

$$f_i := \begin{cases} \sqrt{\frac{|X^c|}{|X|}} & \text{if } v_i \in X \\ -\sqrt{\frac{|X|}{|X^c|}} & \text{if } v_i \in X^c \end{cases}$$

- 2 The RatioCut problem can be reformulated as

$$\min_{X \subset V} \mathbf{f}^T L \mathbf{f} \quad \text{subject to } \mathbf{f} \text{ defined as above}$$

Unfortunately, this problem is NP hard...

Graph Partitioning via Spectral Clustering

Let us reformulate the RatioCut minimization problem.

- 1 Define $\mathbf{f} \in \mathbb{R}^N$ as

$$f_i := \begin{cases} \sqrt{\frac{|X^c|}{|X|}} & \text{if } v_i \in X \\ -\sqrt{\frac{|X|}{|X^c|}} & \text{if } v_i \in X^c \end{cases}$$

- 2 The RatioCut problem can be reformulated as

$$\min_{X \subset V} \mathbf{f}^T L \mathbf{f} \quad \text{subject to } \mathbf{f} \text{ defined as above}$$

Unfortunately, this problem is NP hard... **Relax!**

Graph Partitioning via Spectral Clustering

A couple things to note about \mathbf{f} :

- $\mathbf{f} \perp \mathbf{1} \Leftrightarrow \sum f_i = 0$

$$\begin{aligned} \sum_{i=1}^N f_i &= \sum_{v_i \in X} \sqrt{\frac{|X^c|}{|X|}} - \sum_{v_i \in X^c} \sqrt{\frac{|X|}{|X^c|}} \\ &= |X| \sqrt{\frac{|X^c|}{|X|}} - |X^c| \sqrt{\frac{|X|}{|X^c|}} = 0 \end{aligned}$$

- $\|\mathbf{f}\| = \sqrt{N}$

$$\begin{aligned} \|\mathbf{f}\|^2 &= \sum_{i=1}^N f_i^2 \\ &= |X| \frac{|X^c|}{|X|} + |X^c| \frac{|X|}{|X^c|} \\ &= |X| + |X^c| = N \end{aligned}$$

Graph Partitioning via Spectral Clustering

A couple things to note about f :

- $f \perp \mathbf{1} \Leftrightarrow \sum f_i = 0$

$$\begin{aligned} \sum_{i=1}^N f_i &= \sum_{v_i \in X} \sqrt{\frac{|X^c|}{|X|}} - \sum_{v_i \in X^c} \sqrt{\frac{|X|}{|X^c|}} \\ &= |X| \sqrt{\frac{|X^c|}{|X|}} - |X^c| \sqrt{\frac{|X|}{|X^c|}} = 0 \end{aligned}$$

- $\|f\| = \sqrt{N}$

$$\begin{aligned} \|f\|^2 &= \sum_{i=1}^N f_i^2 \\ &= |X| \frac{|X^c|}{|X|} + |X^c| \frac{|X|}{|X^c|} \\ &= |X| + |X^c| = N \end{aligned}$$

Graph Partitioning via Spectral Clustering

A couple things to note about \mathbf{f} :

- $\mathbf{f} \perp \mathbf{1} \Leftrightarrow \sum f_i = 0$

$$\begin{aligned} \sum_{i=1}^N f_i &= \sum_{v_i \in X} \sqrt{\frac{|X^c|}{|X|}} - \sum_{v_i \in X^c} \sqrt{\frac{|X|}{|X^c|}} \\ &= |X| \sqrt{\frac{|X^c|}{|X|}} - |X^c| \sqrt{\frac{|X|}{|X^c|}} = 0 \end{aligned}$$

- $\|\mathbf{f}\| = \sqrt{N}$

$$\begin{aligned} \|\mathbf{f}\|^2 &= \sum_{i=1}^N f_i^2 \\ &= |X| \frac{|X^c|}{|X|} + |X^c| \frac{|X|}{|X^c|} \\ &= |X| + |X^c| = N \end{aligned}$$

Graph Partitioning via Spectral Clustering

- If we relax our previous definition of \mathbf{f} and simply require that (i) $\mathbf{f} \perp \mathbf{1}$ and (ii) $\|\mathbf{f}\| = \sqrt{N}$, then we get the relaxed minimization problem¹:

$$\min_{\mathbf{f} \in \mathbb{R}^N} \mathbf{f}^T L \mathbf{f} \quad \text{subject to } \mathbf{f} \perp \mathbf{1}, \|\mathbf{f}\| = \sqrt{N}$$

- By the Rayleigh-Ritz Theorem, the solution is given by ϕ_1 (scaled as necessary), where ϕ_1 is the eigenvector corresponding to the second smallest eigenvalue of L .
- ϕ_1 is known as the **Fiedler vector** and is often used to partition a graph into two subsets.
- von Luxburg recommends the use of the *random-walk* version of the Laplacian matrix, $L_{\text{rw}} := I - D^{-1}W$, over the usual Laplacian matrix L , which leads to the *NCut* and the generalized eigenvalue problem:
 $L\phi = \lambda D\phi$.

¹U. von Luxburg: "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp.395-416, 2007.

Graph Partitioning via Spectral Clustering

- If we relax our previous definition of \mathbf{f} and simply require that (i) $\mathbf{f} \perp \mathbf{1}$ and (ii) $\|\mathbf{f}\| = \sqrt{N}$, then we get the relaxed minimization problem¹:

$$\min_{\mathbf{f} \in \mathbb{R}^N} \mathbf{f}^T L \mathbf{f} \quad \text{subject to } \mathbf{f} \perp \mathbf{1}, \|\mathbf{f}\| = \sqrt{N}$$

- By the Rayleigh-Ritz Theorem, the solution is given by ϕ_1 (scaled as necessary), where ϕ_1 is the eigenvector corresponding to the second smallest eigenvalue of L .
- ϕ_1 is known as the **Fiedler vector** and is often used to partition a graph into two subsets.
- von Luxburg recommends the use of the *random-walk* version of the Laplacian matrix, $L_{\text{rw}} := I - D^{-1}W$, over the usual Laplacian matrix L , which leads to the *NCut* and the generalized eigenvalue problem:
 $L\phi = \lambda D\phi$.

¹U. von Luxburg: "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp.395-416, 2007.

Graph Partitioning via Spectral Clustering

- If we relax our previous definition of \mathbf{f} and simply require that (i) $\mathbf{f} \perp \mathbf{1}$ and (ii) $\|\mathbf{f}\| = \sqrt{N}$, then we get the relaxed minimization problem¹:

$$\min_{\mathbf{f} \in \mathbb{R}^N} \mathbf{f}^T L \mathbf{f} \quad \text{subject to } \mathbf{f} \perp \mathbf{1}, \|\mathbf{f}\| = \sqrt{N}$$

- By the Rayleigh-Ritz Theorem, the solution is given by ϕ_1 (scaled as necessary), where ϕ_1 is the eigenvector corresponding to the second smallest eigenvalue of L .
- ϕ_1 is known as the **Fiedler vector** and is often used to partition a graph into two subsets.
- von Luxburg recommends the use of the *random-walk* version of the Laplacian matrix, $L_{\text{rw}} := I - D^{-1}W$, over the usual Laplacian matrix L , which leads to the *NCut* and the generalized eigenvalue problem:
 $L\phi = \lambda D\phi$.

¹U. von Luxburg: "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp.395-416, 2007.

Graph Partitioning via Spectral Clustering

- If we relax our previous definition of \mathbf{f} and simply require that (i) $\mathbf{f} \perp \mathbf{1}$ and (ii) $\|\mathbf{f}\| = \sqrt{N}$, then we get the relaxed minimization problem¹:

$$\min_{\mathbf{f} \in \mathbb{R}^N} \mathbf{f}^T L \mathbf{f} \quad \text{subject to } \mathbf{f} \perp \mathbf{1}, \|\mathbf{f}\| = \sqrt{N}$$

- By the Rayleigh-Ritz Theorem, the solution is given by ϕ_1 (scaled as necessary), where ϕ_1 is the eigenvector corresponding to the second smallest eigenvalue of L .
- ϕ_1 is known as the **Fiedler vector** and is often used to partition a graph into two subsets.
- von Luxburg recommends the use of the *random-walk* version of the Laplacian matrix, $L_{\text{rw}} := I - D^{-1}W$, over the usual Laplacian matrix L , which leads to the *NCut* and the generalized eigenvalue problem:
 $L\phi = \lambda D\phi$.

¹U. von Luxburg: "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp.395-416, 2007.

Graph Partitioning via Spectral Clustering

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

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 $\mathfrak{W}(f)$.

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

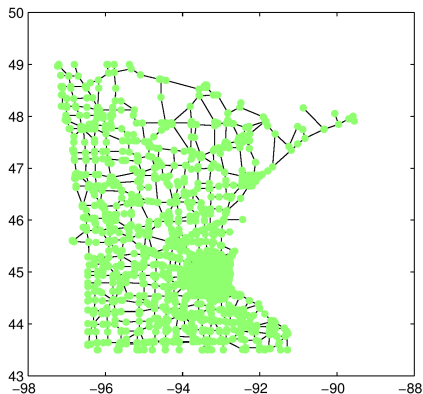


Figure : The MN road network

Example of Graph Partitioning

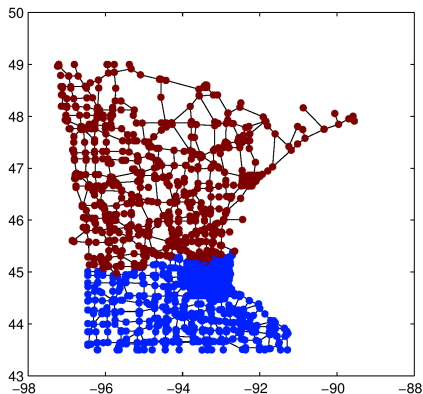


Figure : The MN road network partitioned via the Fiedler vector of L

- 1 Introductory Remarks
- 2 Motivations: Why Graphs?
- 3 Background
 - Basic Graph Theory Terminology
 - Graph Laplacians
 - Graph Partitioning via Spectral Clustering
- 4 Multiscale Basis Dictionaries**
 - Hierarchical Graph Laplacian Eigen Transform (HGLET)
 - Generalized Haar-Walsh Transform (GHWT)
- 5 Best-Basis Algorithm for HGLET & GHWT
- 6 Approximation Experiments
- 7 Summary and Future Work

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

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

- 1 Introductory Remarks
- 2 Motivations: Why Graphs?
- 3 Background
 - Basic Graph Theory Terminology
 - Graph Laplacians
 - Graph Partitioning via Spectral Clustering
- 4 Multiscale Basis Dictionaries**
 - Hierarchical Graph Laplacian Eigen Transform (HGLET)**
 - Generalized Haar-Walsh Transform (GHWT)
- 5 Best-Basis Algorithm for HGLET & GHWT
- 6 Approximation Experiments
- 7 Summary and Future Work

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 \Rightarrow Laplacian eigenvectors (Notation is $\phi_{k,l}^j$ with $j = 0$)
- 2 Partition the graph using the Fiedler vector $\phi_{k,1}^j$
- 3 Generate an orthonormal basis for each of the partitions \Rightarrow Laplacian eigenvectors
- 4 Repeat...
- 5 Select an orthonormal basis from this collection of orthonormal bases

$$\left[\begin{array}{cccccc} \phi_{0,0}^0 & \phi_{0,1}^0 & \phi_{0,2}^0 & \cdots & \phi_{0,N_0^0-1}^0 \end{array} \right]$$

- 1 Generate an orthonormal basis for the entire graph \Rightarrow Laplacian eigenvectors (Notation is $\phi_{k,l}^j$ with $j = 0$)
- 2 Partition the graph using the Fiedler vector $\phi_{k,1}^j$
- 3 Generate an orthonormal basis for each of the partitions \Rightarrow Laplacian eigenvectors
- 4 Repeat...
- 5 Select an orthonormal basis from this collection of orthonormal bases

$$\left[\begin{array}{cccccc} \phi_{0,0}^0 & \phi_{0,1}^0 & \phi_{0,2}^0 & \cdots & \phi_{0,N_0^0-1}^0 \end{array} \right]$$

- 1 Generate an orthonormal basis for the entire graph \Rightarrow **Laplacian eigenvectors** (Notation is $\phi_{k,l}^j$ with $j = 0$)
- 2 Partition the graph using the **Fiedler vector** $\phi_{k,1}^j$
- 3 Generate an orthonormal basis for each of the partitions \Rightarrow **Laplacian eigenvectors**
- 4 Repeat...
- 5 Select an orthonormal basis from this collection of orthonormal bases

$$\left[\phi_{0,0}^0 \quad \phi_{0,1}^0 \quad \phi_{0,2}^0 \quad \cdots \quad \phi_{0,N_0^0-1}^0 \right]$$

$$\left[\phi_{0,0}^1 \quad \phi_{0,1}^1 \quad \phi_{0,2}^1 \quad \cdots \quad \phi_{0,N_0^1-1}^1 \right] \quad \left[\phi_{1,0}^1 \quad \phi_{1,1}^1 \quad \phi_{1,2}^1 \quad \cdots \quad \phi_{1,N_1^1-1}^1 \right]$$

- 1 Generate an orthonormal basis for the entire graph \Rightarrow **Laplacian eigenvectors** (Notation is $\phi_{k,l}^j$ with $j = 0$)
- 2 Partition the graph using the **Fiedler vector** $\phi_{k,1}^j$
- 3 Generate an orthonormal basis for each of the partitions \Rightarrow **Laplacian eigenvectors**
- 4 Repeat...
- 5 Select an orthonormal basis from this collection of orthonormal bases

$$\left[\phi_{0,0}^0 \quad \phi_{0,1}^0 \quad \phi_{0,2}^0 \quad \cdots \quad \phi_{0,N_0^0-1}^0 \right]$$

$$\left[\phi_{0,0}^1 \quad \phi_{0,1}^1 \quad \phi_{0,2}^1 \quad \cdots \quad \phi_{0,N_0^1-1}^1 \right] \quad \left[\phi_{1,0}^1 \quad \phi_{1,1}^1 \quad \phi_{1,2}^1 \quad \cdots \quad \phi_{1,N_1^1-1}^1 \right]$$

- 1 Generate an orthonormal basis for the entire graph \Rightarrow **Laplacian eigenvectors** (Notation is $\phi_{k,l}^j$ with $j = 0$)
- 2 Partition the graph using the **Fiedler vector** $\phi_{k,1}^j$
- 3 Generate an orthonormal basis for each of the partitions \Rightarrow **Laplacian eigenvectors**
- 4 Repeat...
- 5 Select an orthonormal basis from this collection of orthonormal bases

$$\left[\phi_{0,0}^0 \quad \phi_{0,1}^0 \quad \phi_{0,2}^0 \quad \cdots \quad \phi_{0,N_0^0-1}^0 \right]$$

$$\left[\phi_{0,0}^1 \quad \phi_{0,1}^1 \quad \phi_{0,2}^1 \quad \cdots \quad \phi_{0,N_0^1-1}^1 \right] \quad \left[\phi_{1,0}^1 \quad \phi_{1,1}^1 \quad \phi_{1,2}^1 \quad \cdots \quad \phi_{1,N_1^1-1}^1 \right]$$

- 1 Generate an orthonormal basis for the entire graph \Rightarrow **Laplacian eigenvectors** (Notation is $\phi_{k,l}^j$ with $j = 0$)
- 2 Partition the graph using the **Fiedler vector** $\phi_{k,1}^j$
- 3 Generate an orthonormal basis for each of the partitions \Rightarrow **Laplacian eigenvectors**
- 4 Repeat...
- 5 Select an orthonormal basis from this collection of orthonormal bases

$$\left[\phi_{0,0}^0 \quad \phi_{0,1}^0 \quad \phi_{0,2}^0 \quad \cdots \quad \phi_{0,N_0^0-1}^0 \right]$$

$$\left[\phi_{0,0}^1 \quad \phi_{0,1}^1 \quad \phi_{0,2}^1 \quad \cdots \quad \phi_{0,N_0^1-1}^1 \right] \quad \left[\phi_{1,0}^1 \quad \phi_{1,1}^1 \quad \phi_{1,2}^1 \quad \cdots \quad \phi_{1,N_1^1-1}^1 \right]$$

$$\left[\phi_{0,0}^2 \quad \phi_{0,1}^2 \quad \cdots \quad \phi_{0,N_0^2-1}^2 \right] \left[\phi_{1,0}^2 \quad \phi_{1,1}^2 \quad \cdots \quad \phi_{1,N_1^2-1}^2 \right] \left[\phi_{2,0}^2 \quad \phi_{2,1}^2 \quad \cdots \quad \phi_{2,N_2^2-1}^2 \right] \left[\phi_{3,0}^2 \quad \phi_{3,1}^2 \quad \cdots \quad \phi_{3,N_3^2-1}^2 \right]$$

- 1 Generate an orthonormal basis for the entire graph \Rightarrow **Laplacian eigenvectors** (Notation is $\phi_{k,l}^j$ with $j = 0$)
- 2 Partition the graph using the **Fiedler vector** $\phi_{k,1}^j$
- 3 Generate an orthonormal basis for each of the partitions \Rightarrow **Laplacian eigenvectors**
- 4 Repeat...
- 5 Select an orthonormal basis from this collection of orthonormal bases

$$\left[\phi_{0,0}^0 \quad \phi_{0,1}^0 \quad \phi_{0,2}^0 \quad \cdots \quad \phi_{0,N_0^0-1}^0 \right]$$

$$\left[\phi_{0,0}^1 \quad \phi_{0,1}^1 \quad \phi_{0,2}^1 \quad \cdots \quad \phi_{0,N_0^1-1}^1 \right] \quad \left[\phi_{1,0}^1 \quad \phi_{1,1}^1 \quad \phi_{1,2}^1 \quad \cdots \quad \phi_{1,N_1^1-1}^1 \right]$$

$$\left[\phi_{0,0}^2 \quad \phi_{0,1}^2 \quad \cdots \quad \phi_{0,N_0^2-1}^2 \right] \left[\phi_{1,0}^2 \quad \phi_{1,1}^2 \quad \cdots \quad \phi_{1,N_1^2-1}^2 \right] \left[\phi_{2,0}^2 \quad \phi_{2,1}^2 \quad \cdots \quad \phi_{2,N_2^2-1}^2 \right] \left[\phi_{3,0}^2 \quad \phi_{3,1}^2 \quad \cdots \quad \phi_{3,N_3^2-1}^2 \right]$$

- 1 Generate an orthonormal basis for the entire graph \Rightarrow **Laplacian eigenvectors** (Notation is $\phi_{k,l}^j$ with $j = 0$)
- 2 Partition the graph using the **Fiedler vector** $\phi_{k,1}^j$
- 3 Generate an orthonormal basis for each of the partitions \Rightarrow **Laplacian eigenvectors**
- 4 Repeat...
- 5 Select an orthonormal basis from this collection of orthonormal bases

$$\left[\phi_{0,0}^0 \quad \phi_{0,1}^0 \quad \phi_{0,2}^0 \quad \cdots \quad \phi_{0,N_0^0-1}^0 \right]$$

$$\left[\phi_{0,0}^1 \quad \phi_{0,1}^1 \quad \phi_{0,2}^1 \quad \cdots \quad \phi_{0,N_0^1-1}^1 \right] \quad \left[\phi_{1,0}^1 \quad \phi_{1,1}^1 \quad \phi_{1,2}^1 \quad \cdots \quad \phi_{1,N_1^1-1}^1 \right]$$

$$\left[\phi_{0,0}^2 \quad \phi_{0,1}^2 \quad \cdots \quad \phi_{0,N_0^2-1}^2 \right] \left[\phi_{1,0}^2 \quad \phi_{1,1}^2 \quad \cdots \quad \phi_{1,N_1^2-1}^2 \right] \left[\phi_{2,0}^2 \quad \phi_{2,1}^2 \quad \cdots \quad \phi_{2,N_2^2-1}^2 \right] \left[\phi_{3,0}^2 \quad \phi_{3,1}^2 \quad \cdots \quad \phi_{3,N_3^2-1}^2 \right]$$

$$\vdots$$

- 1 Generate an orthonormal basis for the entire graph \Rightarrow **Laplacian eigenvectors** (Notation is $\phi_{k,l}^j$ with $j = 0$)
- 2 Partition the graph using the **Fiedler vector** $\phi_{k,1}^j$
- 3 Generate an orthonormal basis for each of the partitions \Rightarrow **Laplacian eigenvectors**
- 4 Repeat...
- 5 Select an orthonormal basis from this collection of orthonormal bases

$$\left[\phi_{0,0}^0 \quad \phi_{0,1}^0 \quad \phi_{0,2}^0 \quad \cdots \quad \phi_{0,N_0^0-1}^0 \right]$$

$$\left[\phi_{0,0}^1 \quad \phi_{0,1}^1 \quad \phi_{0,2}^1 \quad \cdots \quad \phi_{0,N_0^1-1}^1 \right] \quad \left[\phi_{1,0}^1 \quad \phi_{1,1}^1 \quad \phi_{1,2}^1 \quad \cdots \quad \phi_{1,N_1^1-1}^1 \right]$$

$$\left[\phi_{0,0}^2 \quad \phi_{0,1}^2 \quad \cdots \quad \phi_{0,N_0^2-1}^2 \right] \left[\phi_{1,0}^2 \quad \phi_{1,1}^2 \quad \cdots \quad \phi_{1,N_1^2-1}^2 \right] \left[\phi_{2,0}^2 \quad \phi_{2,1}^2 \quad \cdots \quad \phi_{2,N_2^2-1}^2 \right] \left[\phi_{3,0}^2 \quad \phi_{3,1}^2 \quad \cdots \quad \phi_{3,N_3^2-1}^2 \right]$$

$$\vdots$$

Remarks

- For an unweighted path graph, this yields a dictionary of the block DCT-II
- Similar to wavelet packet or local cosine dictionaries in that it generates an *overcomplete basis* from which we can select a 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 yields a dictionary of the block DCT-II
- Similar to wavelet packet or local cosine dictionaries in that it generates an *overcomplete basis* from which we can select a 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 yields a dictionary of the block DCT-II
- Similar to wavelet packet or local cosine dictionaries in that it generates an *overcomplete basis* from which we can select a 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 yields a dictionary of the block DCT-II
- Similar to wavelet packet or local cosine dictionaries in that it generates an *overcomplete basis* from which we can select a 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

$$\begin{bmatrix}
 \phi_{0,0}^0 & & \phi_{0,1}^0 & & \phi_{0,2}^0 & & \cdots & & \phi_{0,N_0-1}^0 & & \\
 \phi_{0,0}^1 & \phi_{0,1}^1 & \phi_{0,2}^1 & \cdots & \phi_{0,N_0-1}^1 & & & & & & \\
 \phi_{0,0}^2 & \cdots & \phi_{0,N_0-1}^2 & & \phi_{1,0}^2 & \cdots & \phi_{1,N_1-1}^2 & & & & \\
 \phi_{1,0}^1 & \phi_{1,1}^1 & \phi_{1,2}^1 & \cdots & \phi_{1,N_1-1}^1 & & & & & & \\
 \phi_{2,0}^2 & \cdots & \phi_{2,N_2-1}^2 & & \phi_{3,0}^2 & \cdots & \phi_{3,N_3-1}^2 & & & &
 \end{bmatrix}$$

Remarks

- For an unweighted path graph, this yields a dictionary of the block DCT-II
- Similar to wavelet packet or local cosine dictionaries in that it generates an *overcomplete basis* from which we can select a 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

$$\left[\begin{array}{cccccc}
 \phi_{0,0}^0 & & \phi_{0,1}^0 & & \phi_{0,2}^0 & & \dots & & \phi_{0,N_0^0-1}^0 & & \\
 \left[\begin{array}{cccccc}
 \phi_{0,0}^1 & \phi_{0,1}^1 & \phi_{0,2}^1 & \dots & \phi_{0,N_0^1-1}^1 & \\
 \left[\begin{array}{cccc}
 \phi_{0,0}^2 & \dots & \phi_{0,N_0^2-1}^2 & \\
 \left[\begin{array}{ccc}
 \phi_{1,0}^2 & \dots & \phi_{1,N_1^2-1}^2 & \\
 \left[\begin{array}{ccc}
 \phi_{2,0}^2 & \dots & \phi_{2,N_2^2-1}^2 & \\
 \left[\begin{array}{ccc}
 \phi_{3,0}^2 & \dots & \phi_{3,N_3^2-1}^2 &
 \end{array} \right] \right] \right] \right] \right] \right]
 \end{array} \right]
 \end{array} \right]
 \end{array} \right]
 \end{array}$$

Remarks

- For an unweighted path graph, this yields a dictionary of the block DCT-II
- Similar to wavelet packet or local cosine dictionaries in that it generates an *overcomplete basis* from which we can select a 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

$$\begin{bmatrix}
 \boldsymbol{\phi}_{0,0}^0 & & \boldsymbol{\phi}_{0,1}^0 & & \boldsymbol{\phi}_{0,2}^0 & & \cdots & & \boldsymbol{\phi}_{0,N_0^0-1}^0 & & \\
 \boldsymbol{\phi}_{0,0}^1 & \boldsymbol{\phi}_{0,1}^1 & \boldsymbol{\phi}_{0,2}^1 & \cdots & \boldsymbol{\phi}_{0,N_0^1-1}^1 & & & & & & \\
 \boldsymbol{\phi}_{1,0}^1 & \boldsymbol{\phi}_{1,1}^1 & \boldsymbol{\phi}_{1,2}^1 & \cdots & \boldsymbol{\phi}_{1,N_1^1-1}^1 & & & & & & \\
 \boldsymbol{\phi}_{0,0}^2 & \cdots & \boldsymbol{\phi}_{0,N_0^2-1}^2 & & \boldsymbol{\phi}_{1,0}^2 & \cdots & \boldsymbol{\phi}_{1,N_1^2-1}^2 & & \boldsymbol{\phi}_{2,0}^2 & \cdots & \boldsymbol{\phi}_{2,N_2^2-1}^2 & & \boldsymbol{\phi}_{3,0}^2 & \cdots & \boldsymbol{\phi}_{3,N_3^2-1}^2
 \end{bmatrix}$$

Remarks

- For an unweighted path graph, this yields a dictionary of the block DCT-II
- Similar to wavelet packet or local cosine dictionaries in that it generates an *overcomplete basis* from which we can select a 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

$$\begin{bmatrix}
 \phi_{0,0}^0 & & \phi_{0,1}^0 & & \phi_{0,2}^0 & & \cdots & & \phi_{0,N_0-1}^0 & & \\
 \phi_{0,0}^1 & \phi_{0,1}^1 & \phi_{0,2}^1 & \cdots & \phi_{0,N_0-1}^1 & & & & & & \\
 \phi_{1,0}^1 & \phi_{1,1}^1 & \phi_{1,2}^1 & \cdots & \phi_{1,N_1-1}^1 & & & & & & \\
 \phi_{2,0}^2 & \cdots & \phi_{2,N_2-1}^2 & & \phi_{3,0}^2 & \cdots & \phi_{3,N_3-1}^2 & & & &
 \end{bmatrix}$$

Remarks

- For an unweighted path graph, this yields a dictionary of the block DCT-II
- Similar to wavelet packet or local cosine dictionaries in that it generates an *overcomplete basis* from which we can select a 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

$$\begin{bmatrix}
 \phi_{0,0}^0 & & \phi_{0,1}^0 & & \phi_{0,2}^0 & & \cdots & & \phi_{0,N_0-1}^0 & & \\
 \phi_{0,0}^1 & \phi_{0,1}^1 & \phi_{0,2}^1 & \cdots & \phi_{0,N_0-1}^1 & & & & & & \\
 \phi_{0,0}^2 & \cdots & \phi_{0,N_0-1}^2 & & \phi_{1,0}^2 & \cdots & \phi_{1,N_1-1}^2 & & & & \\
 \phi_{1,0}^1 & \phi_{1,1}^1 & \phi_{1,2}^1 & \cdots & \phi_{1,N_1-1}^1 & & & & & & \\
 \phi_{2,0}^2 & \cdots & \phi_{2,N_2-1}^2 & & \phi_{3,0}^2 & \cdots & \phi_{3,N_3-1}^2 & & & &
 \end{bmatrix}$$

Remarks

- For an unweighted path graph, this yields a dictionary of the block DCT-II
- Similar to wavelet packet or local cosine dictionaries in that it generates an *overcomplete basis* from which we can select a 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

$$\begin{bmatrix} \boldsymbol{\phi}_{0,0}^0 & & \boldsymbol{\phi}_{0,1}^0 & & \boldsymbol{\phi}_{0,2}^0 & & \cdots & & \boldsymbol{\phi}_{0,N_0^0-1}^0 \end{bmatrix} \\
 \left[\boldsymbol{\phi}_{0,0}^1 \quad \boldsymbol{\phi}_{0,1}^1 \quad \boldsymbol{\phi}_{0,2}^1 \quad \cdots \quad \boldsymbol{\phi}_{0,N_0^1-1}^1 \right] \left[\boldsymbol{\phi}_{1,0}^1 \quad \boldsymbol{\phi}_{1,1}^1 \quad \boldsymbol{\phi}_{1,2}^1 \quad \cdots \quad \boldsymbol{\phi}_{1,N_1^1-1}^1 \right] \\
 \left[\boldsymbol{\phi}_{0,0}^2 \quad \cdots \quad \boldsymbol{\phi}_{0,N_0^2-1}^2 \right] \left[\boldsymbol{\phi}_{1,0}^2 \quad \cdots \quad \boldsymbol{\phi}_{1,N_1^2-1}^2 \right] \left[\boldsymbol{\phi}_{2,0}^2 \quad \cdots \quad \boldsymbol{\phi}_{2,N_2^2-1}^2 \right] \left[\boldsymbol{\phi}_{3,0}^2 \quad \cdots \quad \boldsymbol{\phi}_{3,N_3^2-1}^2 \right]$$

Related Work

The following work also proposed a similar strategy to construct a multiscale basis dictionary, i.e., *local cosine dictionary on a graph*:

- 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.

Related Work

The following work also proposed a similar strategy to construct a multiscale basis dictionary, i.e., *local cosine dictionary on a graph*:

- 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.

However, in our opinion, the generalization of the folding/unfolding operations (originally used in the construction of the local cosine transforms on a regular domain) to the graph setting may be harmful. We believe that such operations are not necessary for most tasks in practice. If one needs smoother and overlapping basis vectors, then a better partitioning scheme other than the folding/unfolding operations is called for.

Computational Complexity: HGLET

	Computational Complexity	Run Time for MN^1
HGLET (redundant)	$O(N^3)$	67 sec

¹Computations performed on a personal laptop (4.00 GB RAM, 2.26 GHz), $N = 2640$ and $\text{nnz}(W) = 6604$.

- 1 Introductory Remarks
- 2 Motivations: Why Graphs?
- 3 Background
 - Basic Graph Theory Terminology
 - Graph Laplacians
 - Graph Partitioning via Spectral Clustering
- 4 Multiscale Basis Dictionaries**
 - Hierarchical Graph Laplacian Eigen Transform (HGLET)
 - Generalized Haar-Walsh Transform (GHWT)**
- 5 Best-Basis Algorithm for HGLET & GHWT
- 6 Approximation Experiments
- 7 Summary and Future Work

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 proceeds 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 proceeds as follows...

- 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 $\psi_{k,l}^j$
- 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
- 5 Select an orthonormal basis from this collection of orthonormal bases

- 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 $\psi_{k,l}^j$
- 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
- 5 Select an orthonormal basis from this collection of orthonormal bases

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

- 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 $\psi_{k,l}^j$
- 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
- ⑤ Select an orthonormal basis from this collection of orthonormal bases

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

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

- 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 $\psi_{k,l}^j$
- 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
- 5 Select an orthonormal basis from this collection of orthonormal bases

$$\left[\psi_{0,0}^0 \quad \psi_{0,1}^0 \quad \psi_{0,2}^0 \quad \psi_{0,3}^0 \quad \cdots \quad \psi_{0,N-2}^0 \quad \psi_{0,N-1}^0 \right]$$

$$\vdots$$

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

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

- 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 $\psi_{k,l}^j$
- 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
- ⑤ Select an orthonormal basis from this collection of orthonormal bases

$$\left[\psi_{0,0}^0 \quad \psi_{0,1}^0 \quad \psi_{0,2}^0 \quad \psi_{0,3}^0 \quad \cdots \quad \psi_{0,N-2}^0 \quad \psi_{0,N-1}^0 \right]$$

$$\vdots$$

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

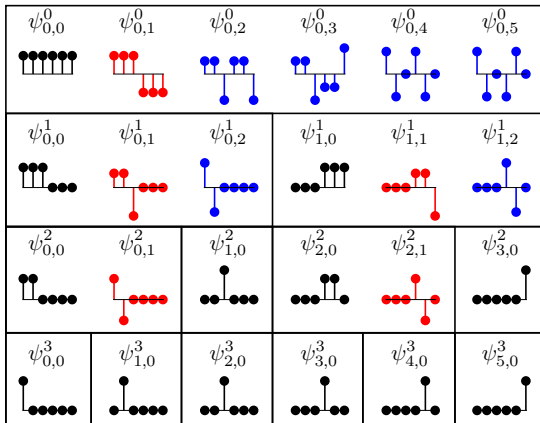
$$\left[\psi_{0,0}^{j_{\max}} \right] \left[\psi_{1,0}^{j_{\max}} \right] \left[\psi_{2,0}^{j_{\max}} \right] \left[\psi_{3,0}^{j_{\max}} \right] \cdots \left[\psi_{K^{j_{\max}}-2,0}^{j_{\max}} \right] \left[\psi_{K^{j_{\max}}-1,0}^{j_{\max}} \right]$$

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**)

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

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**)

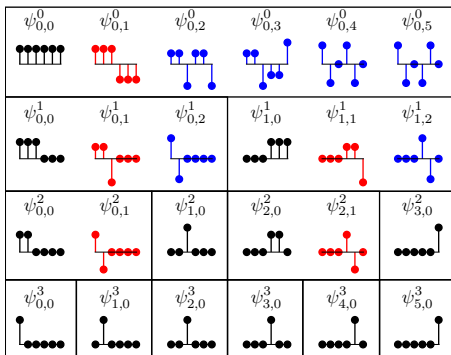


Figure : Default dictionary; i.e., coarse-to-fine

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

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**)

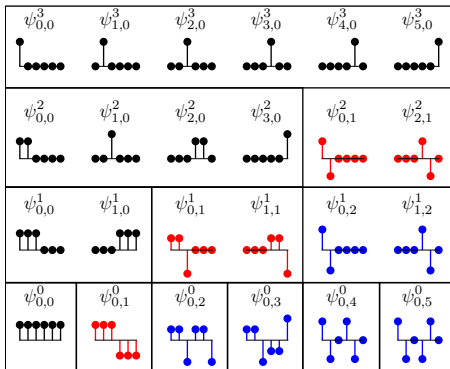


Figure : Reordered & regrouped dictionary; i.e., fine-to-coarse

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

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**)

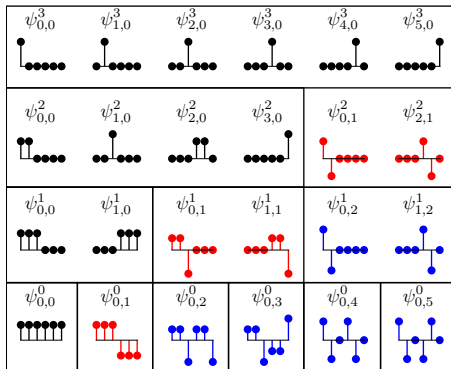


Figure : Reordered & regrouped dictionary; i.e., fine-to-coarse

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

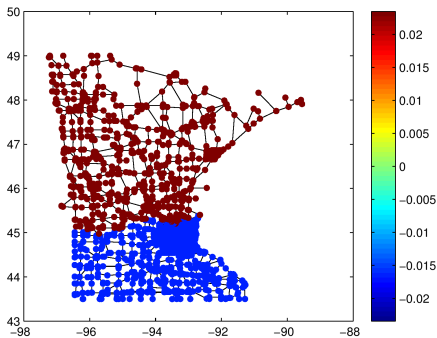
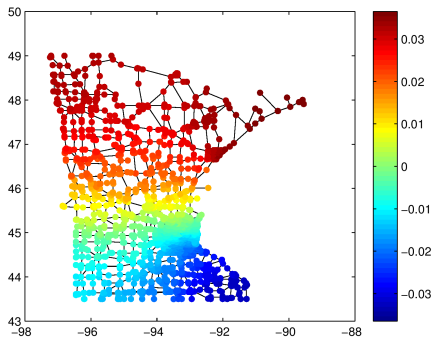
HGLET vs. GHWT

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.)

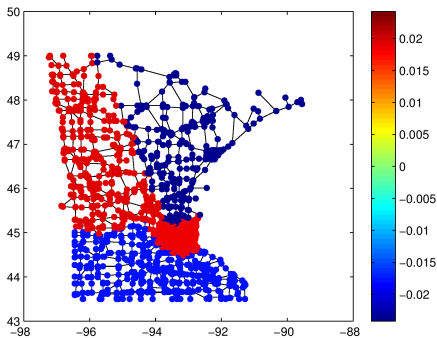
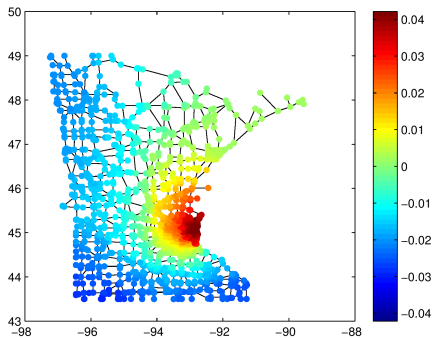
Level $j = 0$, Region $k = 0$, $l = 1$



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.)

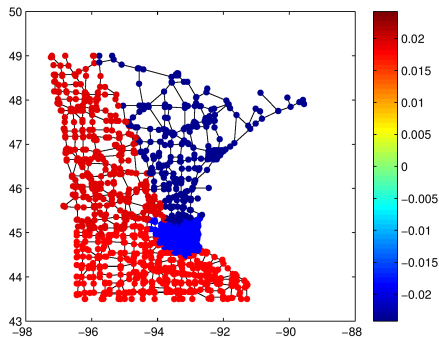
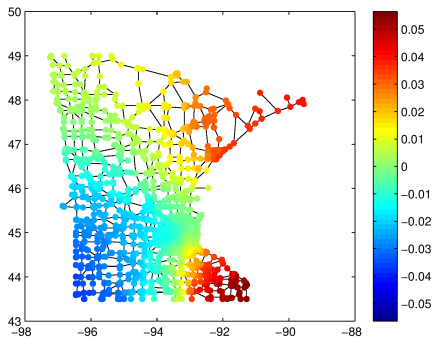
Level $j = 0$, Region $k = 0$, $l = 2$



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.)

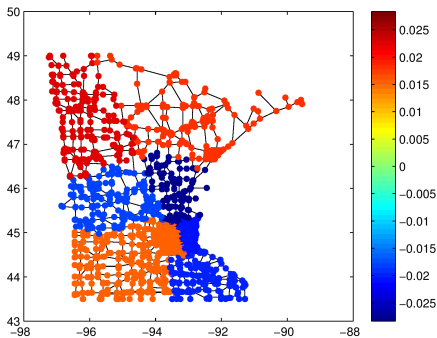
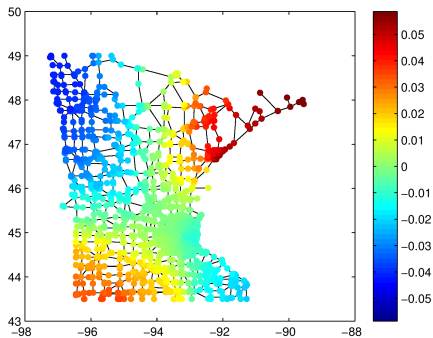
Level $j = 0$, Region $k = 0$, $l = 3$



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.)

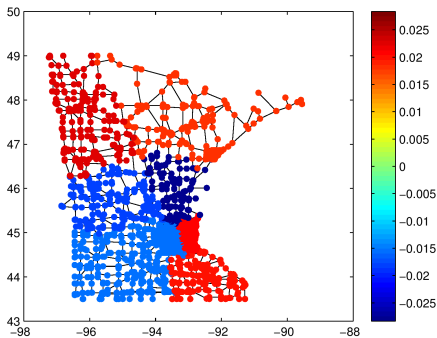
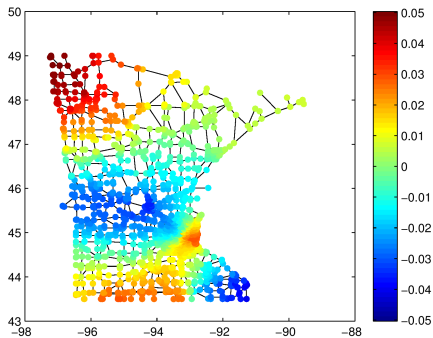
Level $j = 0$, Region $k = 0$, $l = 4$



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.)

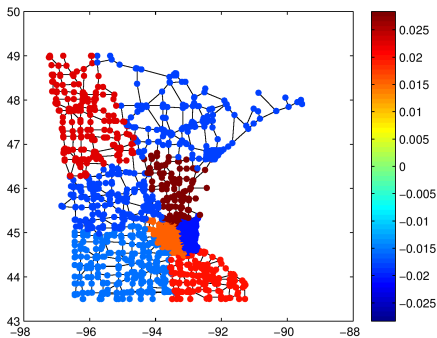
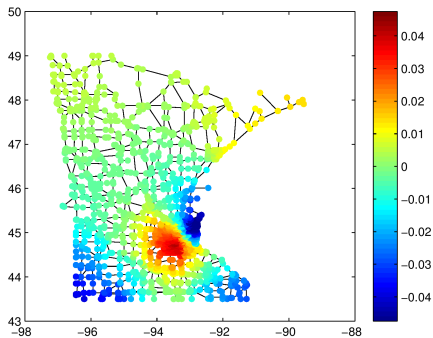
Level $j = 0$, Region $k = 0$, $l = 5$



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.)

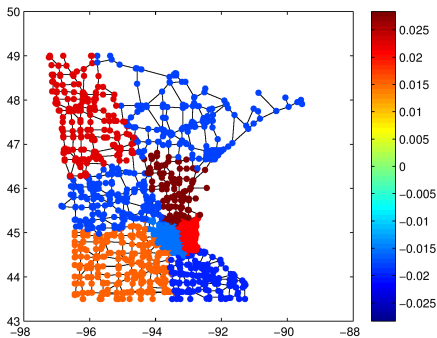
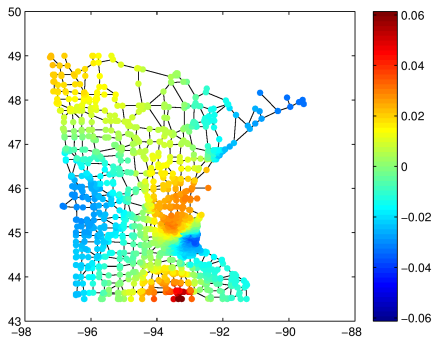
Level $j = 0$, Region $k = 0$, $l = 6$



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.)

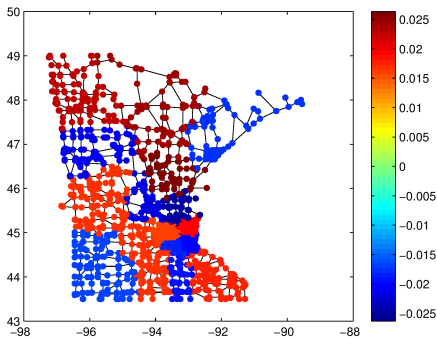
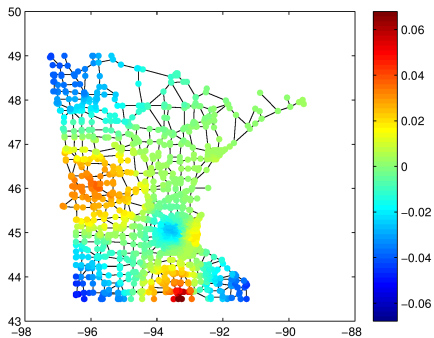
Level $j = 0$, Region $k = 0$, $l = 7$



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.)

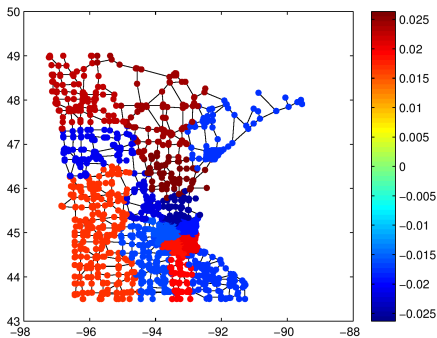
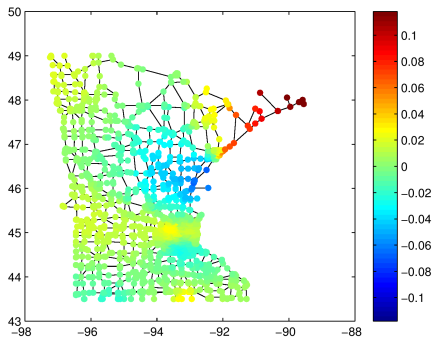
Level $j = 0$, Region $k = 0$, $l = 8$



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.)

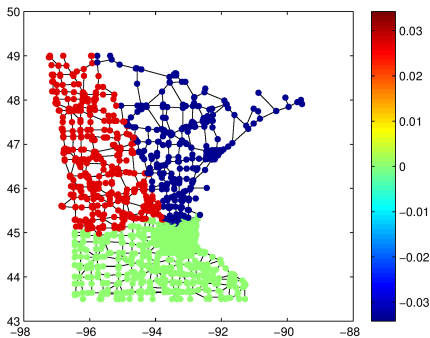
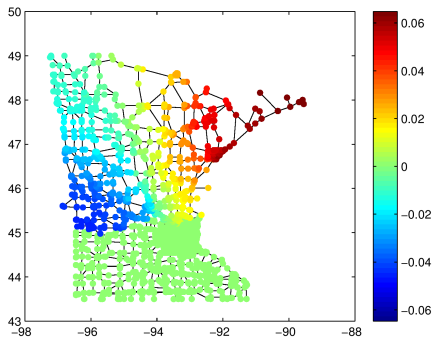
Level $j = 0$, Region $k = 0$, $l = 9$



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.)

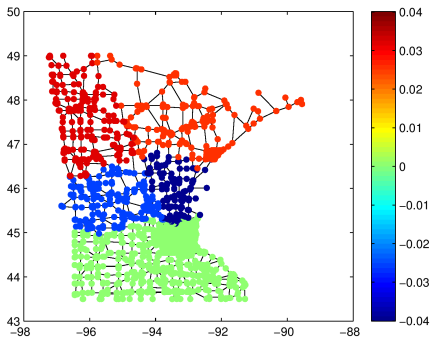
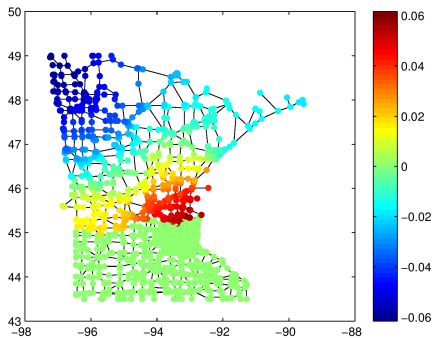
Level $j = 1$, Region $k = 0$, $l = 1$



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.)

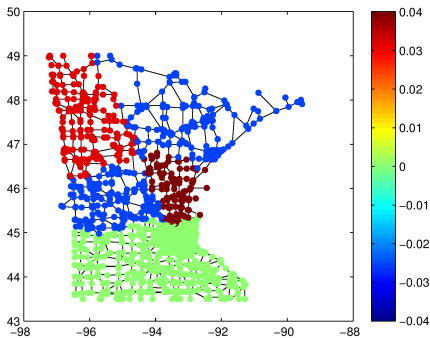
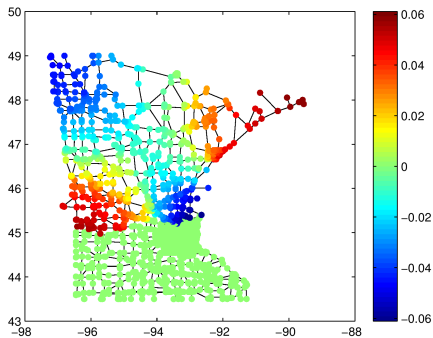
Level $j = 1$, Region $k = 0$, $l = 2$



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.)

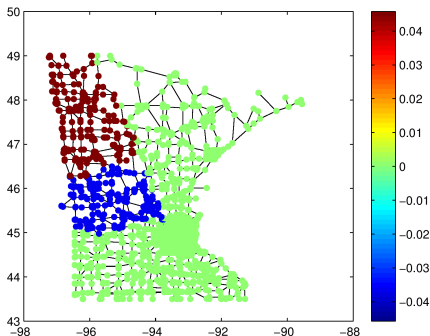
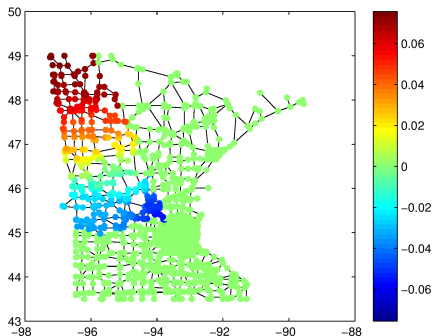
Level $j = 1$, Region $k = 0$, $l = 3$



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.)

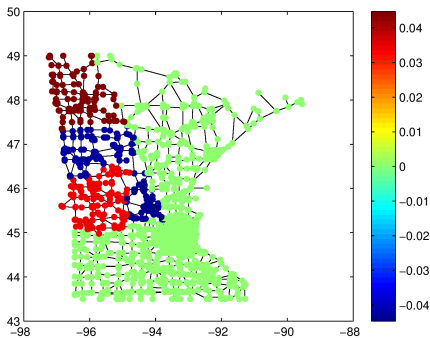
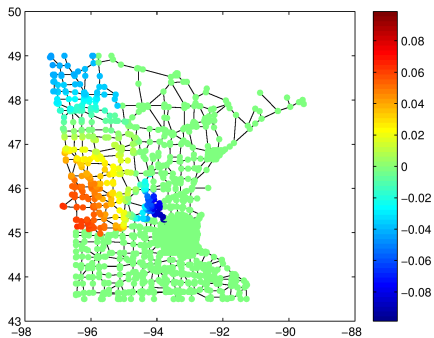
Level $j = 2$, Region $k = 0$, $l = 1$



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.)

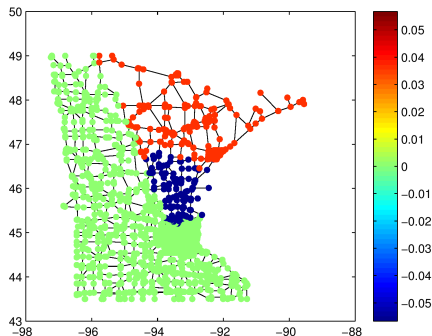
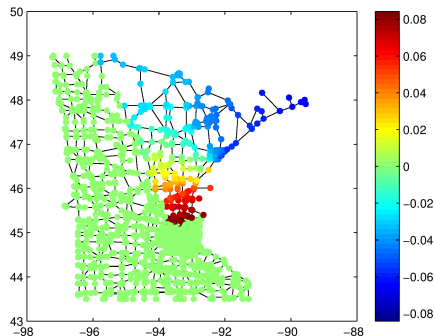
Level $j = 2$, Region $k = 0$, $l = 2$



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.)

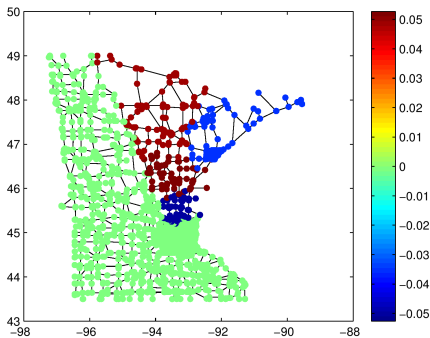
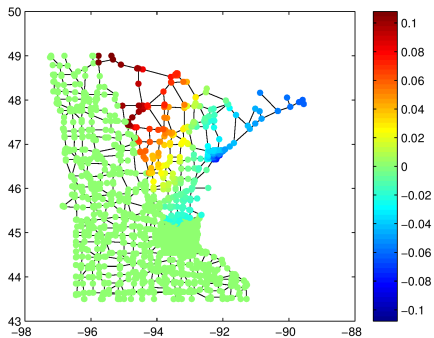
Level $j = 2$, Region $k = 1$, $l = 1$



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.)

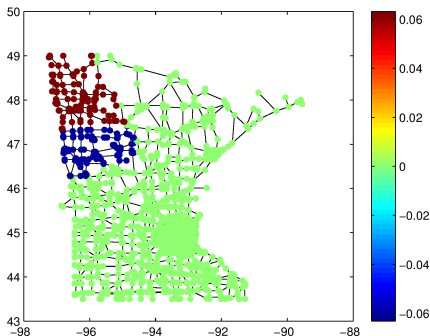
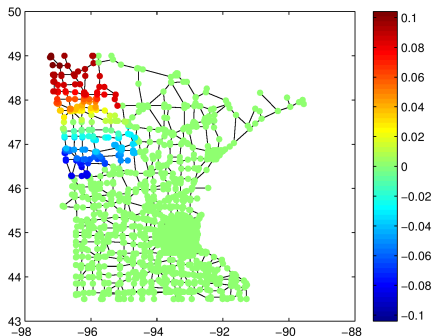
Level $j = 2$, Region $k = 1$, $l = 2$



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.)

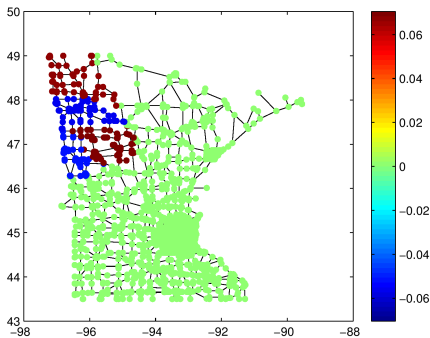
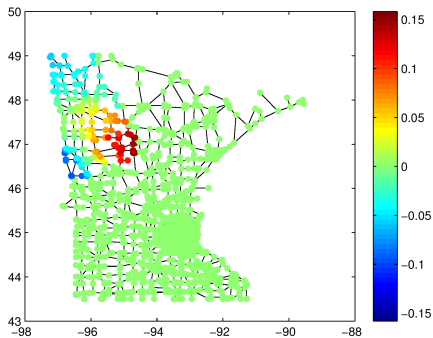
Level $j = 3$, Region $k = 0$, $l = 1$



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.)

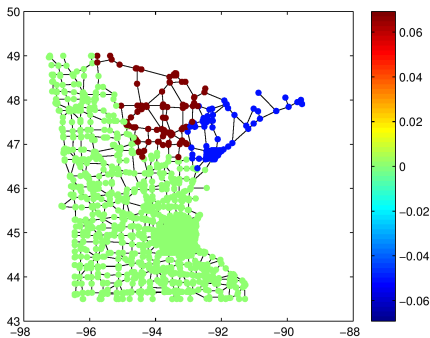
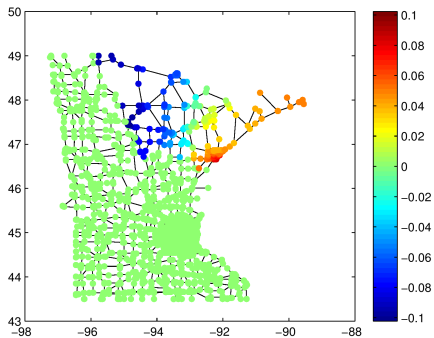
Level $j = 3$, Region $k = 0$, $l = 2$



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.)

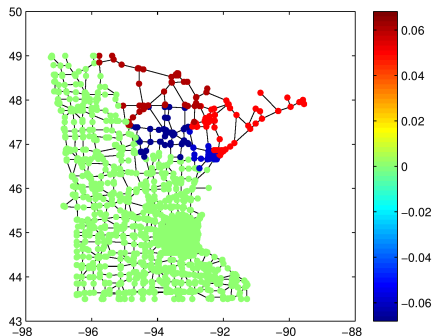
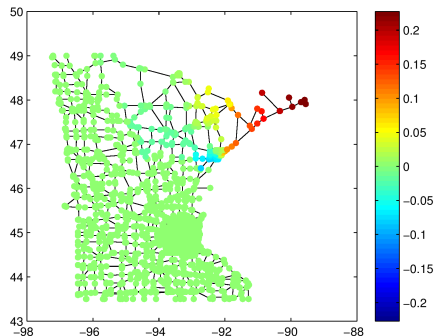
Level $j = 3$, Region $k = 2$, $l = 1$



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



Computational Complexity: GHWT

	Computational Complexity	Run Time for MN^1
HGLET (redundant)	$O(N^3)$	67 sec
GHWT (redundant)	$O(N^2)$	10 sec

¹Computations performed on a personal laptop (4.00 GB RAM, 2.26 GHz), $N = 2640$ and $\text{nnz}(W) = 6604$.

Related Work

The following articles also discussed the Haar-like transform on graphs and trees, but *not the Walsh-Hadamard transform* on them:

- 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.

- 1 Introductory Remarks
- 2 Motivations: Why Graphs?
- 3 Background
 - Basic Graph Theory Terminology
 - Graph Laplacians
 - Graph Partitioning via Spectral Clustering
- 4 Multiscale Basis Dictionaries
 - Hierarchical Graph Laplacian Eigen Transform (HGLET)
 - Generalized Haar-Walsh Transform (GHWT)
- 5 Best-Basis Algorithm for HGLET & GHWT**
- 6 Approximation Experiments
- 7 Summary and Future Work

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.

As before, we require a cost functional \mathcal{J} . For example:

$$\mathcal{J}(\mathbf{x}) = \left(\sum_{i=1}^n |x_i|^p \right)^{1/p} = \text{norm}(\mathbf{x}, p) \quad 0 < p \leq 1$$

- For our approximation experiments in the following pages, we used $p = 0.1$.

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.

As before, we require a cost functional \mathcal{J} . For example:

$$\mathcal{J}(x) = \left(\sum_{i=1}^n |x_i|^p \right)^{1/p} = \text{norm}(x, p) \quad 0 < p \leq 1$$

- For our approximation experiments in the following pages, we used $p = 0.1$.

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.

As before, we require a cost functional \mathcal{J} . For example:

$$\mathcal{J}(\mathbf{x}) = \left(\sum_{i=1}^n |x_i|^p \right)^{1/p} = \text{norm}(\mathbf{x}, p) \quad 0 < p \leq 1$$

- For our approximation experiments in the following pages, we used $p = 0.1$.

$$\begin{bmatrix} \phi_{0,0}^0 & \phi_{0,1}^0 & \phi_{0,2}^0 & \cdots & \phi_{0,N_0^0-1}^0 \\ d_{0,0}^0 & d_{0,1}^0 & d_{0,2}^0 & \cdots & d_{0,N_0^0-1}^0 \end{bmatrix}$$

$$\begin{bmatrix} \phi_{0,0}^1 & \phi_{0,1}^1 & \phi_{0,2}^1 & \cdots & \phi_{0,N_0^1-1}^1 \\ d_{0,0}^1 & d_{0,1}^1 & d_{0,2}^1 & \cdots & d_{0,N_0^1-1}^1 \end{bmatrix} \quad \begin{bmatrix} \phi_{1,0}^1 & \phi_{1,1}^1 & \phi_{1,2}^1 & \cdots & \phi_{1,N_1^1-1}^1 \\ d_{1,0}^1 & d_{1,1}^1 & d_{1,2}^1 & \cdots & d_{1,N_1^1-1}^1 \end{bmatrix}$$

$$\begin{bmatrix} \phi_{0,0}^2 & \phi_{0,1}^2 & \cdots & \phi_{0,N_0^2-1}^2 \\ d_{0,0}^2 & d_{0,1}^2 & \cdots & d_{0,N_0^2-1}^2 \end{bmatrix} \quad \begin{bmatrix} \phi_{1,0}^2 & \phi_{1,1}^2 & \cdots & \phi_{1,N_1^2-1}^2 \\ d_{1,0}^2 & d_{1,1}^2 & \cdots & d_{1,N_1^2-1}^2 \end{bmatrix} \quad \begin{bmatrix} \phi_{2,0}^2 & \phi_{2,1}^2 & \cdots & \phi_{2,N_2^2-1}^2 \\ d_{2,0}^2 & d_{2,1}^2 & \cdots & d_{2,N_2^2-1}^2 \end{bmatrix} \quad \begin{bmatrix} \phi_{3,0}^2 & \phi_{3,1}^2 & \cdots & \phi_{3,N_3^2-1}^2 \\ d_{3,0}^2 & d_{3,1}^2 & \cdots & d_{3,N_3^2-1}^2 \end{bmatrix}$$

$$\begin{bmatrix} \phi_{0,0}^0 & \phi_{0,1}^0 & \phi_{0,2}^0 & \cdots & \phi_{0,N_0^0-1}^0 \\ d_{0,0}^0 & d_{0,1}^0 & d_{0,2}^0 & \cdots & d_{0,N_0^0-1}^0 \end{bmatrix}$$

$$\begin{bmatrix} \phi_{0,0}^1 & \phi_{0,1}^1 & \phi_{0,2}^1 & \cdots & \phi_{0,N_0^1-1}^1 \\ d_{0,0}^1 & d_{0,1}^1 & d_{0,2}^1 & \cdots & d_{0,N_0^1-1}^1 \end{bmatrix} \quad \begin{bmatrix} \phi_{1,0}^1 & \phi_{1,1}^1 & \phi_{1,2}^1 & \cdots & \phi_{1,N_1^1-1}^1 \\ d_{1,0}^1 & d_{1,1}^1 & d_{1,2}^1 & \cdots & d_{1,N_1^1-1}^1 \end{bmatrix}$$

$$\begin{bmatrix} \phi_{0,0}^2 & \phi_{0,1}^2 & \cdots & \phi_{0,N_0^2-1}^2 \\ d_{0,0}^2 & d_{0,1}^2 & \cdots & d_{0,N_0^2-1}^2 \end{bmatrix} \quad \begin{bmatrix} \phi_{1,0}^2 & \phi_{1,1}^2 & \cdots & \phi_{1,N_1^2-1}^2 \\ d_{1,0}^2 & d_{1,1}^2 & \cdots & d_{1,N_1^2-1}^2 \end{bmatrix} \quad \begin{bmatrix} \phi_{2,0}^2 & \phi_{2,1}^2 & \cdots & \phi_{2,N_2^2-1}^2 \\ d_{2,0}^2 & d_{2,1}^2 & \cdots & d_{2,N_2^2-1}^2 \end{bmatrix} \quad \begin{bmatrix} \phi_{3,0}^2 & \phi_{3,1}^2 & \cdots & \phi_{3,N_3^2-1}^2 \\ d_{3,0}^2 & d_{3,1}^2 & \cdots & d_{3,N_3^2-1}^2 \end{bmatrix}$$

$$\begin{bmatrix} \phi_{0,0}^0 & \phi_{0,1}^0 & \phi_{0,2}^0 & \cdots & \phi_{0,N_0^0-1}^0 \\ d_{0,0}^0 & d_{0,1}^0 & d_{0,2}^0 & \cdots & d_{0,N_0^0-1}^0 \end{bmatrix}$$

$$\begin{bmatrix} \phi_{0,0}^1 & \phi_{0,1}^1 & \phi_{0,2}^1 & \cdots & \phi_{0,N_0^1-1}^1 \\ d_{0,0}^1 & d_{0,1}^1 & d_{0,2}^1 & \cdots & d_{0,N_0^1-1}^1 \end{bmatrix} \quad \begin{bmatrix} \phi_{1,0}^1 & \phi_{1,1}^1 & \phi_{1,2}^1 & \cdots & \phi_{1,N_1^1-1}^1 \\ d_{1,0}^1 & d_{1,1}^1 & d_{1,2}^1 & \cdots & d_{1,N_1^1-1}^1 \end{bmatrix}$$

$$\begin{bmatrix} \phi_{0,0}^2 & \phi_{0,1}^2 & \cdots & \phi_{0,N_0^2-1}^2 \\ d_{0,0}^2 & d_{0,1}^2 & \cdots & d_{0,N_0^2-1}^2 \end{bmatrix} \quad \begin{bmatrix} \phi_{1,0}^2 & \phi_{1,1}^2 & \cdots & \phi_{1,N_1^2-1}^2 \\ d_{1,0}^2 & d_{1,1}^2 & \cdots & d_{1,N_1^2-1}^2 \end{bmatrix} \quad \begin{bmatrix} \phi_{2,0}^2 & \phi_{2,1}^2 & \cdots & \phi_{2,N_2^2-1}^2 \\ d_{2,0}^2 & d_{2,1}^2 & \cdots & d_{2,N_2^2-1}^2 \end{bmatrix} \quad \begin{bmatrix} \phi_{3,0}^2 & \phi_{3,1}^2 & \cdots & \phi_{3,N_3^2-1}^2 \\ d_{3,0}^2 & d_{3,1}^2 & \cdots & d_{3,N_3^2-1}^2 \end{bmatrix}$$

$$\begin{bmatrix} \phi_{0,0}^0 & \phi_{0,1}^0 & \phi_{0,2}^0 & \cdots & \phi_{0,N_0-1}^0 \\ d_{0,0}^0 & d_{0,1}^0 & d_{0,2}^0 & \cdots & d_{0,N_0-1}^0 \end{bmatrix}$$

$$\begin{bmatrix} \phi_{0,0}^1 & \phi_{0,1}^1 & \phi_{0,2}^1 & \cdots & \phi_{0,N_0-1}^1 \\ d_{0,0}^1 & d_{0,1}^1 & d_{0,2}^1 & \cdots & d_{0,N_0-1}^1 \end{bmatrix} \quad \begin{bmatrix} \phi_{1,0}^1 & \phi_{1,1}^1 & \phi_{1,2}^1 & \cdots & \phi_{1,N_1-1}^1 \\ d_{1,0}^1 & d_{1,1}^1 & d_{1,2}^1 & \cdots & d_{1,N_1-1}^1 \end{bmatrix}$$

$$\begin{bmatrix} \phi_{2,0}^2 & \phi_{2,1}^2 & \cdots & \phi_{2,N_2-1}^2 \\ d_{2,0}^2 & d_{2,1}^2 & \cdots & d_{2,N_2-1}^2 \end{bmatrix} \quad \begin{bmatrix} \phi_{3,0}^2 & \phi_{3,1}^2 & \cdots & \phi_{3,N_3-1}^2 \\ d_{3,0}^2 & d_{3,1}^2 & \cdots & d_{3,N_3-1}^2 \end{bmatrix}$$

$$\begin{bmatrix} \phi_{0,0}^0 & \phi_{0,1}^0 & \phi_{0,2}^0 & \cdots & \phi_{0,N_0^0-1}^0 \\ d_{0,0}^0 & d_{0,1}^0 & d_{0,2}^0 & \cdots & d_{0,N_0^0-1}^0 \end{bmatrix}$$

$$\begin{bmatrix} \phi_{0,0}^1 & \phi_{0,1}^1 & \phi_{0,2}^1 & \cdots & \phi_{0,N_0^1-1}^1 \\ d_{0,0}^1 & d_{0,1}^1 & d_{0,2}^1 & \cdots & d_{0,N_0^1-1}^1 \end{bmatrix} \quad \begin{bmatrix} \phi_{1,0}^1 & \phi_{1,1}^1 & \phi_{1,2}^1 & \cdots & \phi_{1,N_1^1-1}^1 \\ d_{1,0}^1 & d_{1,1}^1 & d_{1,2}^1 & \cdots & d_{1,N_1^1-1}^1 \end{bmatrix}$$

$$\begin{bmatrix} \phi_{2,0}^2 & \phi_{2,1}^2 & \cdots & \phi_{2,N_2^2-1}^2 \\ d_{2,0}^2 & d_{2,1}^2 & \cdots & d_{2,N_2^2-1}^2 \end{bmatrix} \quad \begin{bmatrix} \phi_{3,0}^2 & \phi_{3,1}^2 & \cdots & \phi_{3,N_3^2-1}^2 \\ d_{3,0}^2 & d_{3,1}^2 & \cdots & d_{3,N_3^2-1}^2 \end{bmatrix}$$

$$\begin{bmatrix} \phi_{0,0}^0 & \phi_{0,1}^0 & \phi_{0,2}^0 & \cdots & \phi_{0,N_0^0-1}^0 \\ d_{0,0}^0 & d_{0,1}^0 & d_{0,2}^0 & \cdots & d_{0,N_0^0-1}^0 \end{bmatrix}$$

$$\begin{bmatrix} \phi_{0,0}^1 & \phi_{0,1}^1 & \phi_{0,2}^1 & \cdots & \phi_{0,N_0^1-1}^1 \\ d_{0,0}^1 & d_{0,1}^1 & d_{0,2}^1 & \cdots & d_{0,N_0^1-1}^1 \end{bmatrix}$$

$$\begin{bmatrix} \phi_{2,0}^2 & \phi_{2,1}^2 & \cdots & \phi_{2,N_2^2-1}^2 \\ d_{2,0}^2 & d_{2,1}^2 & \cdots & d_{2,N_2^2-1}^2 \end{bmatrix} \quad \begin{bmatrix} \phi_{3,0}^2 & \phi_{3,1}^2 & \cdots & \phi_{3,N_3^2-1}^2 \\ d_{3,0}^2 & d_{3,1}^2 & \cdots & d_{3,N_3^2-1}^2 \end{bmatrix}$$

$$\begin{bmatrix} \phi_{0,0}^0 & \phi_{0,1}^0 & \phi_{0,2}^0 & \cdots & \phi_{0,N_0^0-1}^0 \\ d_{0,0}^0 & d_{0,1}^0 & d_{0,2}^0 & \cdots & d_{0,N_0^0-1}^0 \end{bmatrix}$$

$$\begin{bmatrix} \phi_{0,0}^1 & \phi_{0,1}^1 & \phi_{0,2}^1 & \cdots & \phi_{0,N_0^1-1}^1 \\ d_{0,0}^1 & d_{0,1}^1 & d_{0,2}^1 & \cdots & d_{0,N_0^1-1}^1 \end{bmatrix}$$

$$\begin{bmatrix} \phi_{2,0}^2 & \phi_{2,1}^2 & \cdots & \phi_{2,N_2^2-1}^2 \\ d_{2,0}^2 & d_{2,1}^2 & \cdots & d_{2,N_2^2-1}^2 \end{bmatrix} \quad \begin{bmatrix} \phi_{3,0}^2 & \phi_{3,1}^2 & \cdots & \phi_{3,N_3^2-1}^2 \\ d_{3,0}^2 & d_{3,1}^2 & \cdots & d_{3,N_3^2-1}^2 \end{bmatrix}$$

$$\begin{bmatrix} \phi_{0,0}^1 & \phi_{0,1}^1 & \phi_{0,2}^1 & \cdots & \phi_{0,N_0^1-1}^1 \\ d_{0,0}^1 & d_{0,1}^1 & d_{0,2}^1 & \cdots & d_{0,N_0^1-1}^1 \end{bmatrix}$$

$$\begin{bmatrix} \phi_{2,0}^2 & \phi_{2,1}^2 & \cdots & \phi_{2,N_2^2-1}^2 \\ d_{2,0}^2 & d_{2,1}^2 & \cdots & d_{2,N_2^2-1}^2 \end{bmatrix} \quad \begin{bmatrix} \phi_{3,0}^2 & \phi_{3,1}^2 & \cdots & \phi_{3,N_3^2-1}^2 \\ d_{3,0}^2 & d_{3,1}^2 & \cdots & d_{3,N_3^2-1}^2 \end{bmatrix}$$

$$\begin{bmatrix} \phi_{0,0}^1 & \phi_{0,1}^1 & \phi_{0,2}^1 & \cdots & \phi_{0,N_0^1-1}^1 \\ d_{0,0}^1 & d_{0,1}^1 & d_{0,2}^1 & \cdots & d_{0,N_0^1-1}^1 \end{bmatrix}$$

$$\begin{bmatrix} \phi_{2,0}^2 & \phi_{2,1}^2 & \cdots & \phi_{2,N_2^2-1}^2 \\ d_{2,0}^2 & d_{2,1}^2 & \cdots & d_{2,N_2^2-1}^2 \end{bmatrix} \quad \begin{bmatrix} \phi_{3,0}^2 & \phi_{3,1}^2 & \cdots & \phi_{3,N_3^2-1}^2 \\ d_{3,0}^2 & d_{3,1}^2 & \cdots & d_{3,N_3^2-1}^2 \end{bmatrix}$$

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

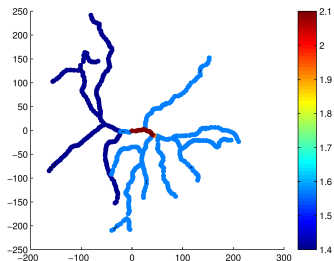
$$\begin{bmatrix} \phi_{0,0}^1 & \phi_{0,1}^1 & \phi_{0,2}^1 & \cdots & \phi_{0,N_0^1-1}^1 \\ d_{0,0}^1 & d_{0,1}^1 & d_{0,2}^1 & \cdots & d_{0,N_0^1-1}^1 \end{bmatrix}$$

$$\begin{bmatrix} \phi_{2,0}^2 & \phi_{2,1}^2 & \cdots & \phi_{2,N_2^2-1}^2 \\ d_{2,0}^2 & d_{2,1}^2 & \cdots & d_{2,N_2^2-1}^2 \end{bmatrix} \quad \begin{bmatrix} \phi_{3,0}^2 & \phi_{3,1}^2 & \cdots & \phi_{3,N_3^2-1}^2 \\ d_{3,0}^2 & d_{3,1}^2 & \cdots & d_{3,N_3^2-1}^2 \end{bmatrix}$$

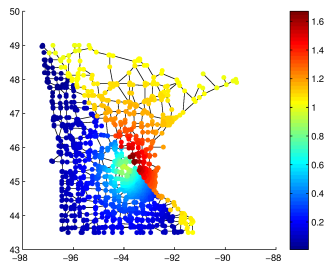
According to cost functional \mathcal{J} , this is the best basis for approximation.

- With the GHWT bases, we run the best-basis algorithm on both the default (coarse-to-fine) dictionary and the reorganized (fine-to-coarse) dictionary and then compare the cost of the 2 bases to determine the best-basis.

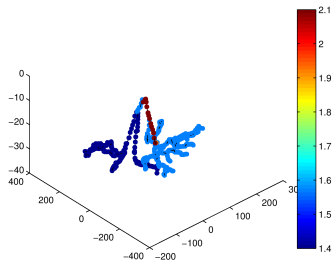
- 1 Introductory Remarks
- 2 Motivations: Why Graphs?
- 3 Background
 - Basic Graph Theory Terminology
 - Graph Laplacians
 - Graph Partitioning via Spectral Clustering
- 4 Multiscale Basis Dictionaries
 - Hierarchical Graph Laplacian Eigen Transform (HGLET)
 - Generalized Haar-Walsh Transform (GHWT)
- 5 Best-Basis Algorithm for HGLET & GHWT
- 6 Approximation Experiments**
- 7 Summary and Future Work



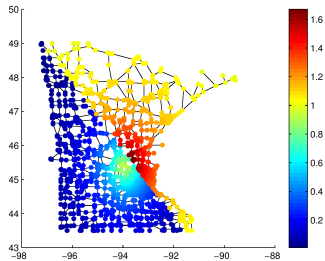
(a) Thickness data on a dendritic tree



(b) A mutilated Gaussian on the MN road network

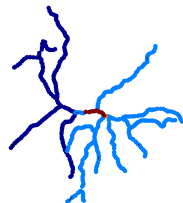
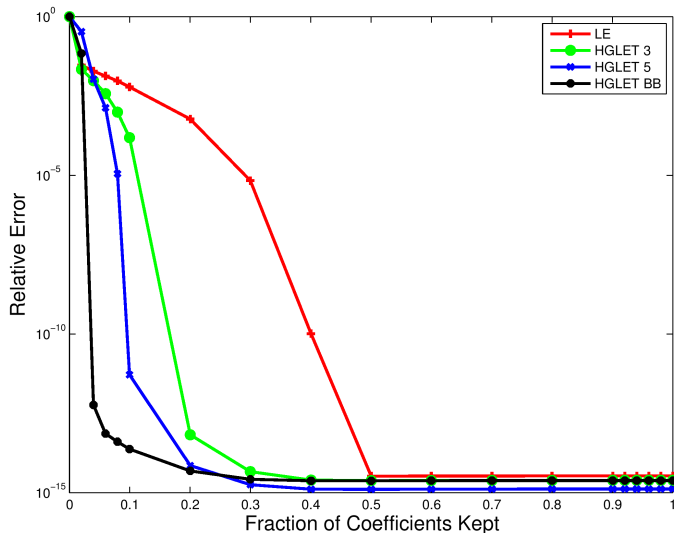


(a) Thickness data on a dendritic tree

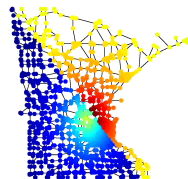
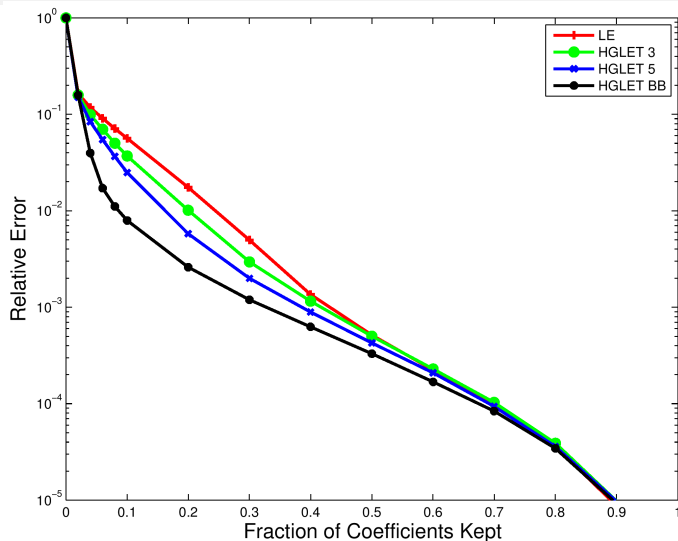


(b) A mutilated Gaussian on the MN road network

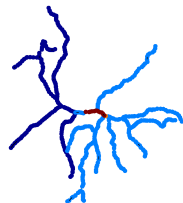
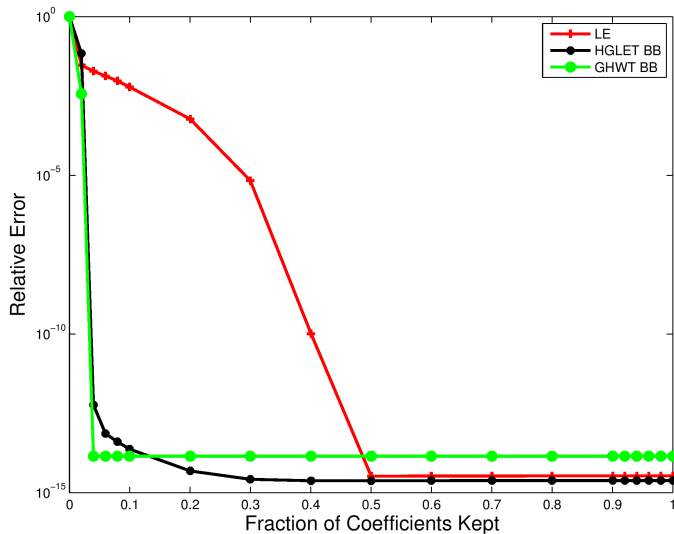
HGLET on Dendrite (weights = inv. Euclidean dist.)



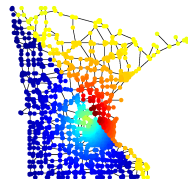
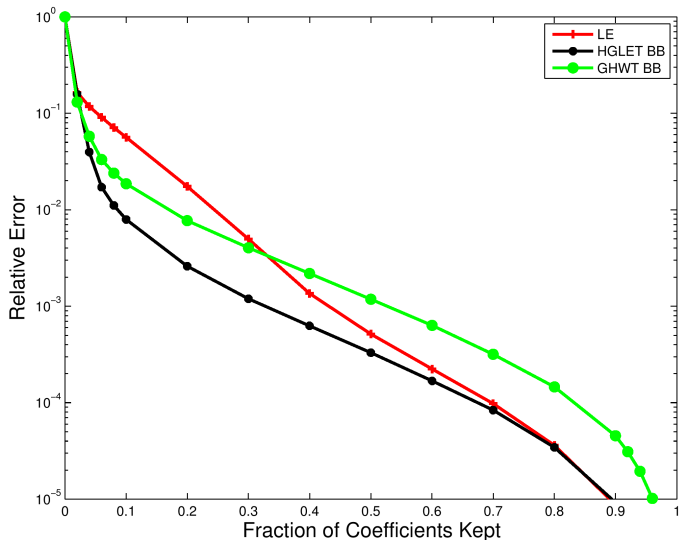
HGLET on MN Mutilated Gaussian (weights = inv. Euclidean dist.)



GHWT vs. HGLET on Dendrite



GHWT vs. HGLET on MN Mutilated Gaussian



Discussion of Approximation Results

- From the HGLET plots, we see that HGLET best-basis > HGLET Level 5 > HGLET Level 3 > Laplacian eigenvectors (HGLET Level 0)
- The HGLET best-basis performs the best on the MN Mutilated Gaussian dataset while the GHWT best-basis outperformed the others on the Dendrite dataset
- These performances make a strong case for using localized basis vectors on *multiple scales*
- Also, these indicate that the *smoothness* of the basis vectors matters depending on the smoothness inherent in data

Discussion of Approximation Results

- From the HGLET plots, we see that HGLET best-basis > HGLET Level 5 > HGLET Level 3 > Laplacian eigenvectors (HGLET Level 0)
- The HGLET best-basis performs the best on the MN Mutilated Gaussian dataset while the GHWT best-basis outperformed the others on the Dendrite dataset
- These performances make a strong case for using localized basis vectors on *multiple scales*
- Also, these indicate that the *smoothness* of the basis vectors matters depending on the smoothness inherent in data

Discussion of Approximation Results

- From the HGLET plots, we see that HGLET best-basis > HGLET Level 5 > HGLET Level 3 > Laplacian eigenvectors (HGLET Level 0)
- The HGLET best-basis performs the best on the MN Mutilated Gaussian dataset while the GHWT best-basis outperformed the others on the Dendrite dataset
- These performances make a strong case for using localized basis vectors on *multiple scales*
- Also, these indicate that the *smoothness* of the basis vectors matters depending on the smoothness inherent in data

Discussion of Approximation Results

- From the HGLET plots, we see that HGLET best-basis > HGLET Level 5 > HGLET Level 3 > Laplacian eigenvectors (HGLET Level 0)
- The HGLET best-basis performs the best on the MN Mutilated Gaussian dataset while the GHWT best-basis outperformed the others on the Dendrite dataset
- These performances make a strong case for using localized basis vectors on *multiple scales*
- Also, these indicate that the *smoothness* of the basis vectors matters depending on the smoothness inherent in data

- 1 Introductory Remarks
- 2 Motivations: Why Graphs?
- 3 Background
 - Basic Graph Theory Terminology
 - Graph Laplacians
 - Graph Partitioning via Spectral Clustering
- 4 Multiscale Basis Dictionaries
 - Hierarchical Graph Laplacian Eigen Transform (HGLET)
 - Generalized Haar-Walsh Transform (GHWT)
- 5 Best-Basis Algorithm for HGLET & GHWT
- 6 Approximation Experiments
- 7 Summary and Future Work**

Summary

- We developed **multiscale basis dictionaries** on graphs and networks: HGLET and GHWT. We also developed a corresponding best-basis algorithm.
- The HGLET is a direct generalization of *Hierarchical Block Discrete Cosine Transforms* originally developed for regularly-sampled signals and images.
- The GHWT is a generalization of the *Haar Transform* and the *Walsh-Hadamard Transform*.
- Both of these transforms allow us to choose an orthonormal basis most suitable for the task at hand, e.g., approximation, classification, regression, . . .
- They may also be useful for regularly-sampled signals, e.g., can deal with signals of non-dyadic length; adaptive segmentation, . . .
- Developing a *true* generalization of smoother wavelet and wavelet packet transforms is more challenging due to the difficulty of the notion of the *frequency domain* of a given graph.

Summary

- We developed **multiscale basis dictionaries** on graphs and networks: HGLET and GHWT. We also developed a corresponding best-basis algorithm.
- The HGLET is a direct generalization of *Hierarchical Block Discrete Cosine Transforms* originally developed for regularly-sampled signals and images.
- The GHWT is a generalization of the *Haar Transform* and the *Walsh-Hadamard Transform*.
- Both of these transforms allow us to choose an orthonormal basis most suitable for the task at hand, e.g., approximation, classification, regression, . . .
- They may also be useful for regularly-sampled signals, e.g., can deal with signals of non-dyadic length; adaptive segmentation, . . .
- Developing a *true* generalization of smoother wavelet and wavelet packet transforms is more challenging due to the difficulty of the notion of the *frequency domain* of a given graph.

Summary

- We developed **multiscale basis dictionaries** on graphs and networks: HGLET and GHWT. We also developed a corresponding best-basis algorithm.
- The HGLET is a direct generalization of *Hierarchical Block Discrete Cosine Transforms* originally developed for regularly-sampled signals and images.
- The GHWT is a generalization of the *Haar Transform* and the *Walsh-Hadamard Transform*.
- Both of these transforms allow us to choose an orthonormal basis most suitable for the task at hand, e.g., approximation, classification, regression, . . .
- They may also be useful for regularly-sampled signals, e.g., can deal with signals of non-dyadic length; adaptive segmentation, . . .
- Developing a *true* generalization of smoother wavelet and wavelet packet transforms is more challenging due to the difficulty of the notion of the *frequency domain* of a given graph.

Summary

- We developed **multiscale basis dictionaries** on graphs and networks: HGLET and GHWT. We also developed a corresponding best-basis algorithm.
- The HGLET is a direct generalization of *Hierarchical Block Discrete Cosine Transforms* originally developed for regularly-sampled signals and images.
- The GHWT is a generalization of the *Haar Transform* and the *Walsh-Hadamard Transform*.
- Both of these transforms allow us to choose an orthonormal basis most suitable for the task at hand, e.g., approximation, classification, regression, . . .
- They may also be useful for regularly-sampled signals, e.g., can deal with signals of non-dyadic length; adaptive segmentation, . . .
- Developing a *true* generalization of smoother wavelet and wavelet packet transforms is more challenging due to the difficulty of the notion of the *frequency domain* of a given graph.

Summary

- We developed **multiscale basis dictionaries** on graphs and networks: HGLET and GHWT. We also developed a corresponding best-basis algorithm.
- The HGLET is a direct generalization of *Hierarchical Block Discrete Cosine Transforms* originally developed for regularly-sampled signals and images.
- The GHWT is a generalization of the *Haar Transform* and the *Walsh-Hadamard Transform*.
- Both of these transforms allow us to choose an orthonormal basis most suitable for the task at hand, e.g., approximation, classification, regression, . . .
- They may also be useful for regularly-sampled signals, e.g., can deal with signals of non-dyadic length; adaptive segmentation, . . .
- Developing a *true* generalization of smoother wavelet and wavelet packet transforms is more challenging due to the difficulty of the notion of the *frequency domain* of a given graph.

Summary

- We developed **multiscale basis dictionaries** on graphs and networks: HGLET and GHWT. We also developed a corresponding best-basis algorithm.
- The HGLET is a direct generalization of *Hierarchical Block Discrete Cosine Transforms* originally developed for regularly-sampled signals and images.
- The GHWT is a generalization of the *Haar Transform* and the *Walsh-Hadamard Transform*.
- Both of these transforms allow us to choose an orthonormal basis most suitable for the task at hand, e.g., approximation, classification, regression, . . .
- They may also be useful for regularly-sampled signals, e.g., can deal with signals of non-dyadic length; adaptive segmentation, . . .
- Developing a *true* generalization of smoother wavelet and wavelet packet transforms is more challenging due to the difficulty of the notion of the *frequency domain* of a given graph.

Future Work

- Perform classification experiments and compare the results using HGLET and GHWT.
- Explore other methods for graph partitioning:
 - Allow for splitting of a region into an arbitrary number of subregions;
 - Consider a bottom-up clustering method, rather than a top-down partitioning method;
 - Incorporate the *diffuse interface model* and the minimization of the *Ginsburg-Landau functional* proposed by Bertozzi and Flenner (2012).

Future Work

- Perform classification experiments and compare the results using HGLET and GHWT.
- Explore other methods for graph partitioning:
 - Allow for splitting of a region into an arbitrary number of subregions;
 - Consider a bottom-up clustering method, rather than a top-down partitioning method;
 - Incorporate the *diffuse interface model* and the minimization of the *Ginsburg-Landau functional* proposed by Bertozzi and Flenner (2012).

References & Acknowledgments

- <http://www.math.ucdavis.edu/~saito/courses/HarmGraph/> contains my course slides and useful information on “Harmonic Analysis on Graphs and Networks”
- Also visit <http://www.math.ucdavis.edu/~saito/publications/> for various related publications including:
 - 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,” *Japan SIAM Letters*, vol. 1, pp. 13–16, 2009.
 - Y. Nakatsukasa, N. Saito, & E. Woei: “Mysteries around graph Laplacian eigenvalue 4,” *Linear Algebra & Its Applications*, vol. 438, no. 8, pp. 3231–3246, 2013.
 - J. Irion & N. Saito: “Hierarchical graph Laplacian eigen transforms,” *Japan SIAM Letters*, to appear, 2014.
 - J. Irion & N. Saito: “The generalized Haar-Walsh transform,” submitted for publication, 2014.

This research was partially supported by the grants received from the Office of Naval Research and the National Defense Science and Engineering Graduate Fellowship.