On Wavelet and Wavelet Packet Transforms on Graphs and Networks

Naoki Saito & Jeff Irion

Department of Mathematics
University of California, Davis

Organized Session by Activity Group on Wavelet Analysis
JSIAM 2013 Annual Meeting
Fukuoka, Japan
September 11, 2013

Aims & Objectives

Wavelets

- Successful on regular domains
- Extend to irregular domains ⇒ "2nd Generation Wavelets"

For example,

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

Aims & Objectives

Wavelets

- Successful on regular domains
- Extend to irregular domains ⇒ "2nd Generation Wavelets"

For example,

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

Aims & Objectives

Step 1. Develop and implement multiscale transforms for data on graphs and point clouds.

Step 2. Investigate usefulness for:

1. **Approximation/Denoising.**
   - Smoothing crime rate data
2. **Classification.**
   - Twitter spam account classification/detection

Aims & Objectives

Step 1. Develop and implement multiscale transforms for data on graphs and point clouds.

Step 2. Investigate usefulness for:

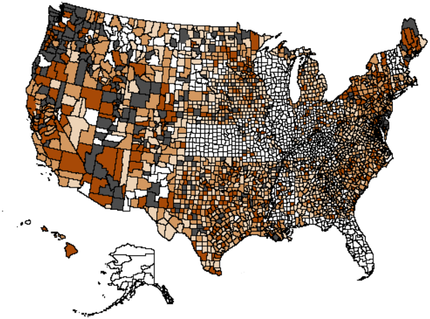
1. **Approximation/Denoising.**
   - Smoothing crime rate data
2. **Classification.**
   - Twitter spam account classification/detection

Aims & Objectives

Step 1. Develop and implement multiscale transforms for data on graphs and point clouds.

Step 2. Investigate usefulness for:

1. **Approximation/Denoising.**
   - Smoothing crime rate data



https://www.ncjrs.gov

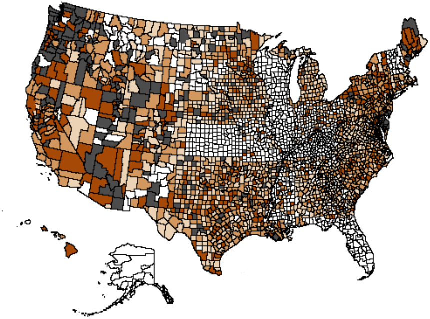
2. Classification.
   - Twitter spam account classification/detection

Aims & Objectives

Step 1. Develop and implement multiscale transforms for data on graphs and point clouds.

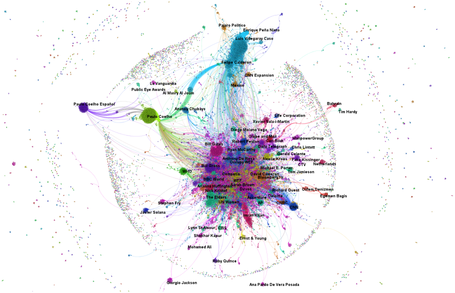
Step 2. Investigate usefulness for:

1. **Approximation/Denoising.**
   - Smoothing crime rate data



https://www.ncjrs.gov

2. **Classification.**
   - Twitter spam account classification/detection



http://beautifuldata.net

Basic Definitions and Notation

- Let G be a graph.
- If G is a connected graph without cycles/loops, then it is called a tree.
- Let $V = V(G) = \{v_1, \ldots, v_N\}$ be a set of vertices representing some data.
- Let $|V(G)| = N$, and let $0 = \lambda_0(G) \le \lambda_1(G) \le \cdots \le \lambda_{N-1}(G)$ be the sorted eigenvalues of $L(G)$.
- Let $E = E(G) = \{e_1, \ldots, e_{N'}\}$ be a 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 $1 \le i, j \le N$. Note that if G is a tree, then $|E(G)| = |V(G)| - 1$.
- Let $d(v_k) = d_{v_k}$ be the degree of the vertex v_k.

Graph Laplacians

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

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

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

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

Why Graph Laplacians?

- Let $f \in L^2(V)$. Then
$$L(G)f(v_i) = d_{v_i} f(v_i) - \sum_{j \neq i} w_{ij} f(v_j),$$
i.e., 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.
- *Spherical harmonics, Bessel functions, and Prolate Spheroidal Wave Functions* are part of the eigenfunctions of the Laplacian for the *spherical, cylindrical, and spheroidal* domains, respectively.
- Hence, the eigenfunction expansion of data measured at the vertices using the eigenfunctions (in fact, eigenvectors) of a graph Laplacian corresponds to *Fourier (or spectral) analysis of the data on that graph*.
- They also play a useful role in understanding a graph (e.g., *the discrete nodal domain theorem* useful for grouping vertices; see Bıyıkoğlu, Leydold, & Stadler, LNM, Springer, 2007)

Why Graph Laplacians? ...

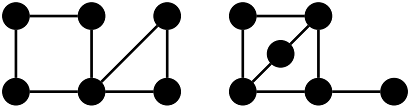
- Furthermore, the eigenvalues of $L(G)$ reflect various intrinsic geometric and topological information about the graph including:
  - connectivity or the number of separated components
  - diameter (the maximum distance over all pairs of vertices)
  - mean distance, ...
  - Fan Chung: *Spectral Graph Theory*, AMS, 1997, says: *"This monograph is an intertwined tale of eigenvalues and their use in unlocking a thousand secrets about graphs.*
- However, eigenvalues of $L(G)$ cannot uniquely determine the graph G.
  $\sim$ Kac (1966): "Can one hear the shape of a drum?"
  $\Rightarrow$ Gordon, Webb, & Wolpert (1992): "One cannot hear the shape of a drum."
- An example of "isospectral" graphs (Tan, 1998; Fujii & Katsuda, 1999):

Why Graph Laplacians? ...

- Furthermore, the eigenvalues of $L(G)$ reflect various intrinsic geometric and topological information about the graph including:
  - connectivity or the number of separated components
  - diameter (the maximum distance over all pairs of vertices)
  - mean distance, ...
  - Fan Chung: *Spectral Graph Theory*, AMS, 1997, says: *"This monograph is an intertwined tale of eigenvalues and their use in unlocking a thousand secrets about graphs.*
- However, eigenvalues of $L(G)$ cannot uniquely determine the graph G.
  $\sim$ Kac (1966): "Can one hear the shape of a drum?"
  $\Rightarrow$ Gordon, Webb, & Wolpert (1992): "One cannot hear the shape of a drum."
- An example of "isospectral" graphs (Tan, 1998; Fujii & Katsuda, 1999):

Why Graph Laplacians? ...

- Furthermore, the eigenvalues of $L(G)$ reflect various intrinsic geometric and topological information about the graph including:
  - connectivity or the number of separated components
  - diameter (the maximum distance over all pairs of vertices)
  - mean distance, ...
  - Fan Chung: *Spectral Graph Theory*, AMS, 1997, says: *"This monograph is an intertwined tale of eigenvalues and their use in unlocking a thousand secrets about graphs.*
- However, eigenvalues of $L(G)$ cannot uniquely determine the graph G.
  $\sim$ Kac (1966): "Can one hear the shape of a drum?"
  $\Rightarrow$ Gordon, Webb, & Wolpert (1992): "One cannot hear the shape of a drum."
- An example of "isospectral" graphs (Tan, 1998; Fujii & Katsuda, 1999):



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).

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

Now we turn our focus to 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).

In order to utilize a hierarchical scheme, we will need to partition the graph. Therefore, we will now review some information about graph partitioning.

Now we turn our focus to 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).

In order to utilize a hierarchical scheme, we will need to partition the graph. Therefore, we will now review some information about graph partitioning.

Graph Partitioning via Spectral Clustering

Goal: split the vertices V into two subsets, X and X^c.

Plan: minimize the RatioCut function[1],

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

where

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

- 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
- Dividing by the *volume* of nodes instead of the number of nodes leads to the popular *Normalized Cut* (NCut) of Shi and Malik[2]

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

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

Graph Partitioning via Spectral Clustering

Goal: split the vertices V into two subsets, X and X^c.

Plan: minimize the RatioCut function[1],

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

where

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

- Dividing by the number of nodes ensures that the partitions are of roughly the same size ⇒ we do not simply cleave a small number of nodes
- Dividing by the *volume* of nodes instead of the number of nodes leads to the popular *Normalized Cut* (NCut) of Shi and Malik[2]

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

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

Graph Partitioning via Spectral Clustering

Goal: split the vertices V into two subsets, X and X^c.

Plan: minimize the RatioCut function[1],

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

where

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

- Dividing by the number of nodes ensures that the partitions are of roughly the same size ⇒ we do not simply cleave a small number of nodes
- Dividing by the *volume* of nodes instead of the number of nodes leads to the popular *Normalized Cut* (NCut) of Shi and Malik[2]

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

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

Graph Partitioning via Spectral Clustering

Let us reformulate the RatioCut minimization problem.

1. Define $\boldsymbol{f} \in \mathbb{R}^N$ 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
$$\min_{X \subset V} \boldsymbol{f}^{\mathsf{T}} L \boldsymbol{f} \qquad \text{s.t.} \qquad \boldsymbol{f} \text{ defined as above}$$

Graph Partitioning via Spectral Clustering

Let us reformulate the RatioCut minimization problem.

1. Define $\boldsymbol{f} \in \mathbb{R}^N$ 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

$$\min_{X \subset V} \boldsymbol{f}^{\top} L \boldsymbol{f} \qquad \text{s.t.} \qquad \boldsymbol{f} \text{ defined as above}$$

Graph Partitioning via Spectral Clustering

Let us reformulate the RatioCut minimization problem.

1. Define $\boldsymbol{f} \in \mathbb{R}^N$ 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

$$\min_{X \subset V} \boldsymbol{f}^{\mathsf{T}} L \boldsymbol{f} \qquad \text{s.t.} \qquad \boldsymbol{f} \text{ defined as above}$$

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

Graph Partitioning via Spectral Clustering

Let us reformulate the RatioCut minimization problem.

1. Define $\boldsymbol{f} \in \mathbb{R}^N$ 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
$$\min_{X \subset V} \boldsymbol{f}^{\mathsf{T}} L \boldsymbol{f}, \qquad \boldsymbol{f} \text{ defined as above}$$

Unfortunately, this problem is NP hard...

Graph Partitioning via Spectral Clustering

Let us reformulate the RatioCut minimization problem.

1. Define $\boldsymbol{f} \in \mathbb{R}^N$ 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

$$\min_{X \subset V} \boldsymbol{f}^{\mathsf{T}} L \boldsymbol{f}, \qquad \boldsymbol{f} \text{ defined as above}$$

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

Graph Partitioning via Spectral Clustering

A couple things to note about $\boldsymbol{f}$:

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

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

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

$$\|\boldsymbol{f}\|^2 = \sum_{i=1}^{N} f_i^2$$
$$= |X|\frac{|X^c|}{|X|} + |X^c|\frac{|X|}{|X^c|}$$
$$= |X| + |X^c| = N$$

Graph Partitioning via Spectral Clustering

A couple things to note about $\boldsymbol{f}$:

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

$$\sum_{i=1}^{N} f_i = \sum_{v_i \in X} \sqrt{\frac{|X^c|}{|X|}} - \sum_{v_i \in X^c} \sqrt{\frac{|X|}{|X^c|}}$$

$$= |X| \sqrt{\frac{|X^c|}{|X|}} - |X^c| \sqrt{\frac{|X|}{|X^c|}} = 0$$

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

$$\|\boldsymbol{f}\|^2 = \sum_{i=1}^{N} f_i^2$$

$$= |X| \frac{|X^c|}{|X|} + |X^c| \frac{|X|}{|X^c|}$$

$$= |X| + |X^c| = N$$

Graph Partitioning via Spectral Clustering

A couple things to note about $\boldsymbol{f}$:

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

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

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

$$\|\boldsymbol{f}\|^2 = \sum_{i=1}^{N} f_i^2$$
$$= |X|\frac{|X^c|}{|X|} + |X^c|\frac{|X|}{|X^c|}$$
$$= |X| + |X^c| = N$$

Graph Partitioning via Spectral Clustering

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

$$\min_{X \subset V} \boldsymbol{f}^{\mathsf{T}} L \boldsymbol{f} \qquad \text{s.t.} \qquad \boldsymbol{f} \perp \mathbf{1},\ \|\boldsymbol{f}\| = \sqrt{N}$$

- By the Rayleigh-Ritz Theorem, the solution is given by $\boldsymbol{\phi}_1$ (scaled as necessary), where $\boldsymbol{\phi}_1$ is the eigenvector corresponding to the second smallest eigenvalue of L.
- $\boldsymbol{\phi}_1$ 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, $L_{\mathrm{rw}} := I - D^{-1}W$, over the usual Laplacian matrix L, which leads to the *NCut* and the generalized eigenvalue problem: $L\boldsymbol{\phi} = \lambda D \boldsymbol{\phi}$.

[1] U. 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 $\boldsymbol{f}$ and simply require that (i) $\boldsymbol{f} \perp \mathbf{1}$ and (ii) $\|\boldsymbol{f}\| = \sqrt{N}$, then we get the relaxed minimization problem[1]:

$$\min_{X \subset V} \boldsymbol{f}^{\mathsf{T}} L \boldsymbol{f} \qquad \text{s.t.} \qquad \boldsymbol{f} \perp \mathbf{1},\ \|\boldsymbol{f}\| = \sqrt{N}$$

- By the Rayleigh-Ritz Theorem, the solution is given by $\boldsymbol{\phi}_1$ (scaled as necessary), where $\boldsymbol{\phi}_1$ is the eigenvector corresponding to the second smallest eigenvalue of L.
- $\boldsymbol{\phi}_1$ 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, $L_{\mathrm{rw}} := I - D^{-1}W$, over the usual Laplacian matrix L, which leads to the *NCut* and the generalized eigenvalue problem: $L\boldsymbol{\phi} = \lambda D \boldsymbol{\phi}$.

[1] U. 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 $\boldsymbol{f}$ and simply require that (i) $\boldsymbol{f} \perp \mathbf{1}$ and (ii) $\|\boldsymbol{f}\| = \sqrt{N}$, then we get the relaxed minimization problem[1]:

$$\min_{X \subset V} \boldsymbol{f}^{\mathsf{T}} L \boldsymbol{f} \qquad \text{s.t.} \qquad \boldsymbol{f} \perp \mathbf{1},\ \|\boldsymbol{f}\| = \sqrt{N}$$

- By the Rayleigh-Ritz Theorem, the solution is given by $\boldsymbol{\phi}_1$ (scaled as necessary), where $\boldsymbol{\phi}_1$ is the eigenvector corresponding to the second smallest eigenvalue of L.
- $\boldsymbol{\phi}_1$ 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, $L_{\mathrm{rw}} := I - D^{-1}W$, over the usual Laplacian matrix L, which leads to the *NCut* and the generalized eigenvalue problem: $L\boldsymbol{\phi} = \lambda D \boldsymbol{\phi}$.

[1] U. 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 $\boldsymbol{f}$ and simply require that (i) $\boldsymbol{f} \perp \mathbf{1}$ and (ii) $\|\boldsymbol{f}\| = \sqrt{N}$, then we get the relaxed minimization problem[1]:

$$\min_{X \subset V} \boldsymbol{f}^{\mathsf{T}} L \boldsymbol{f} \qquad \text{s.t.} \qquad \boldsymbol{f} \perp \mathbf{1},\ \|\boldsymbol{f}\| = \sqrt{N}$$

- By the Rayleigh-Ritz Theorem, the solution is given by $\boldsymbol{\phi}_1$ (scaled as necessary), where $\boldsymbol{\phi}_1$ is the eigenvector corresponding to the second smallest eigenvalue of L.
- $\boldsymbol{\phi}_1$ 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, $L_{\mathrm{rw}} := I - D^{-1}W$, over the usual Laplacian matrix L, which leads to the *NCut* and the generalized eigenvalue problem: $L\boldsymbol{\phi} = \lambda D \boldsymbol{\phi}$.

[1] U. 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

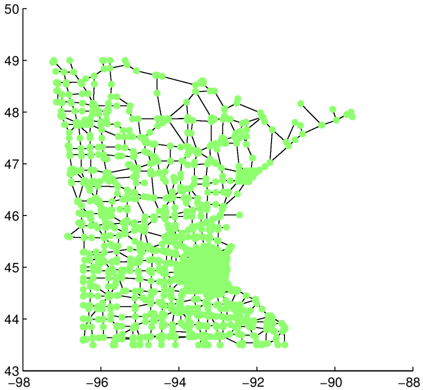


Figure: The MN road network

Example of Graph Partitioning

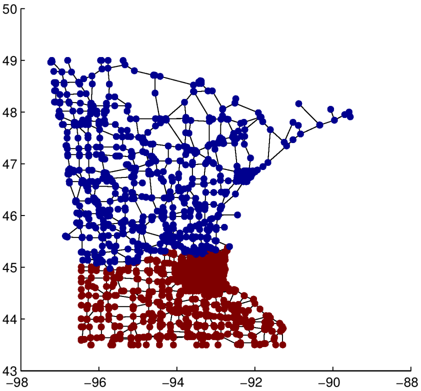


Figure: The MN road network partitioned into two regions via the Fiedler vector

And now, we present our Hierarchical Graph Laplacian Eigen Transform:

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

And now, we present our Hierarchical Graph Laplacian Eigen Transform:

1. Generate an orthonormal basis for the entire graph $\Rightarrow$ Laplacian eigenvectors (Notation is $\phi^j_{k,l}$ with $j=0$)
2. Partition the graph using the Fiedler vector $\phi^j_{k,1}$
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

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

And now, we present our Hierarchical Graph Laplacian Eigen Transform:

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

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

And now, we present our Hierarchical Graph Laplacian Eigen Transform:

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

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

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

And now, we present our Hierarchical Graph Laplacian Eigen Transform:

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

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

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

And now, we present our Hierarchical Graph Laplacian Eigen Transform:

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

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

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

And now, we present our Hierarchical Graph Laplacian Eigen Transform:

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

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

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

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

And now, we present our Hierarchical Graph Laplacian Eigen Transform:

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

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

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

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

And now, we present our Hierarchical Graph Laplacian Eigen Transform:

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

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

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

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

$$\vdots$$

And now, we present our Hierarchical Graph Laplacian Eigen Transform:

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

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

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

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

$$\vdots$$

Observations

- 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

Observations

- 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

Observations

- 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

Observations

- 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

Observations

- 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 ⇒ best-basis algorithm, local discriminant basis algorithm, ...
  - A union of bases on disjoint subsets is obviously orthonormal

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

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

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

Observations

- 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 ⇒ best-basis algorithm, local discriminant basis algorithm, . . .
  - A union of bases on disjoint subsets is obviously orthonormal

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

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

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

Observations

- 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 ⇒ best-basis algorithm, local discriminant basis algorithm, ...
  - A union of bases on disjoint subsets is obviously orthonormal

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

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

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

Observations

- 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

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

$$[\boldsymbol{\phi}^1_{0,0} \quad \boldsymbol{\phi}^1_{0,1} \quad \boldsymbol{\phi}^1_{0,2} \quad \cdots \quad \boldsymbol{\phi}^1_{0,N_0-1}]\ {\color{red}[\boldsymbol{\phi}^1_{1,0} \quad \boldsymbol{\phi}^1_{1,1} \quad \boldsymbol{\phi}^1_{1,2} \quad \cdots \quad \boldsymbol{\phi}^1_{1,N_1-1}]}$$

$${\color{red}[\boldsymbol{\phi}^2_{0,0} \quad \cdots \quad \boldsymbol{\phi}^2_{0,N_0-1}]\ [\boldsymbol{\phi}^2_{1,0} \quad \cdots \quad \boldsymbol{\phi}^2_{1,N_1-1}]}\ [\boldsymbol{\phi}^2_{2,0} \quad \cdots \quad \boldsymbol{\phi}^2_{2,N_2-1}]\ [\boldsymbol{\phi}^2_{3,0} \quad \cdots \quad \boldsymbol{\phi}^2_{3,N_3-1}]$$

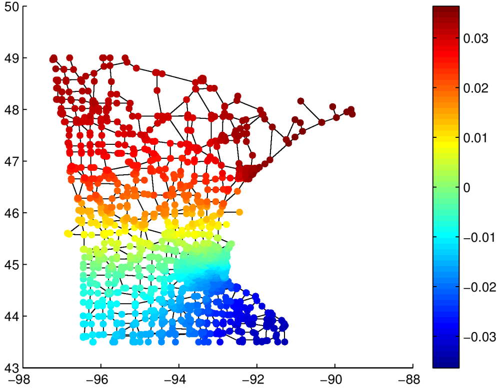
HGLET Basis Vectors on MN

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

HGLET Basis Vectors on MN

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

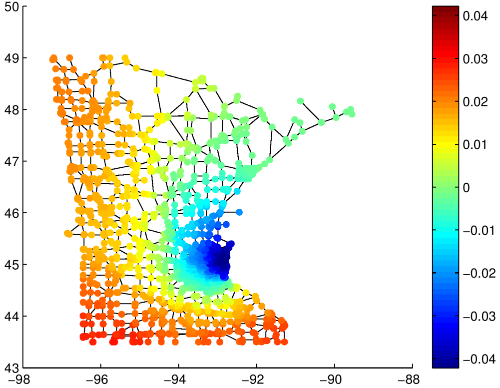
Level $j = 0$, Region $k = 0$, $\phi^{0}_{0,1}$



HGLET Basis Vectors on MN

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

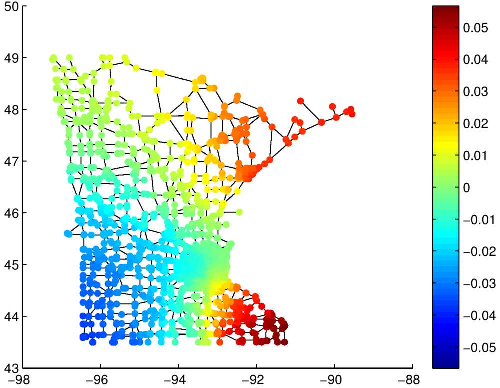
Level $j = 0$, Region $k = 0$, $\boldsymbol{\phi}^{0}_{0,2}$



HGLET Basis Vectors on MN

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

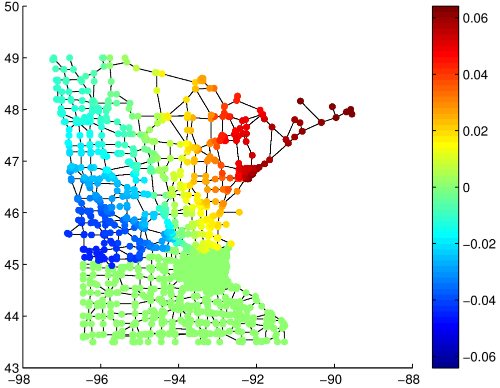
Level $j = 0$, Region $k = 0$, $\boldsymbol{\phi}_{0,3}^{0}$



HGLET Basis Vectors on MN

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

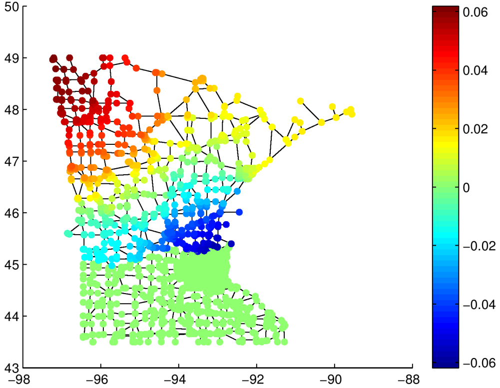
Level $j = 1$, Region $k = 0$, $\boldsymbol{\phi}^1_{0,1}$



HGLET Basis Vectors on MN

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

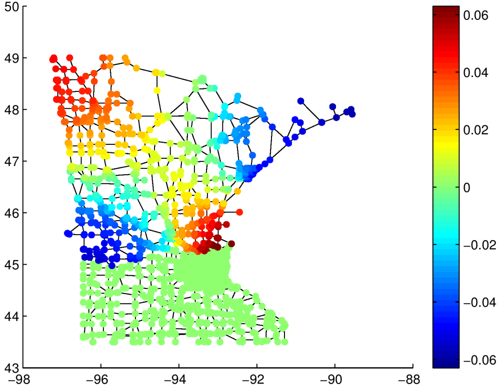
Level $j = 1$, Region $k = 0$, $\boldsymbol{\phi}^1_{0,2}$



HGLET Basis Vectors on MN

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

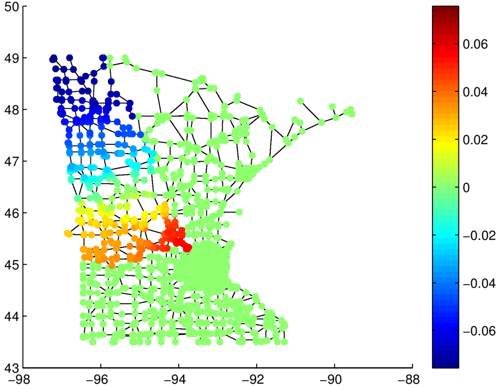
Level $j = 1$, Region $k = 0$, $\boldsymbol{\phi}^1_{0,3}$



HGLET Basis Vectors on MN

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

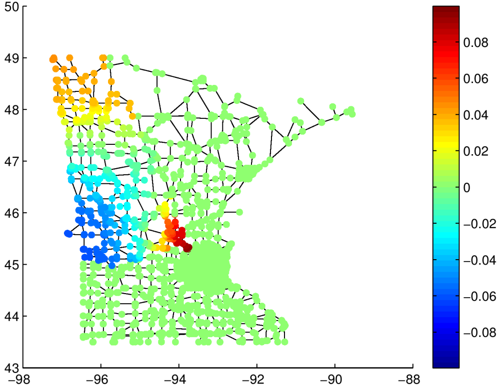
Level $j = 2$, Region $k = 0$, $\phi^2_{0,1}$



HGLET Basis Vectors on MN

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

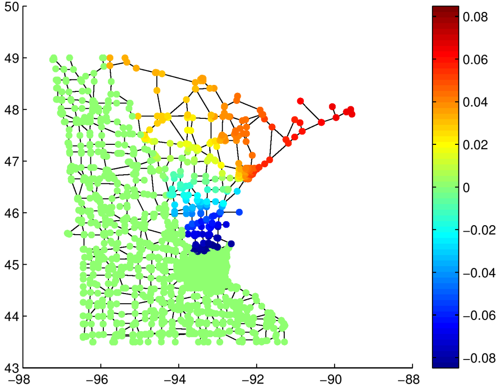
Level $j = 2$, Region $k = 0$, $\boldsymbol{\phi}^2_{0,2}$



HGLET Basis Vectors on MN

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

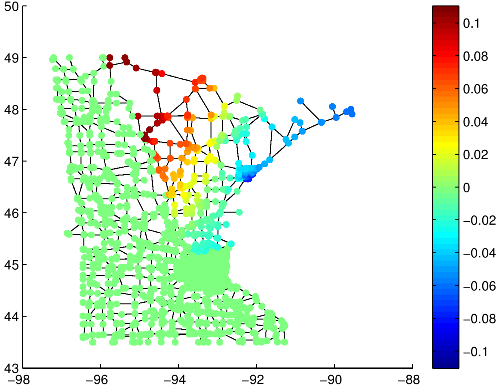
Level $j = 2$, Region $k = 1$, $\boldsymbol{\phi}^2_{1,1}$



HGLET Basis Vectors on MN

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

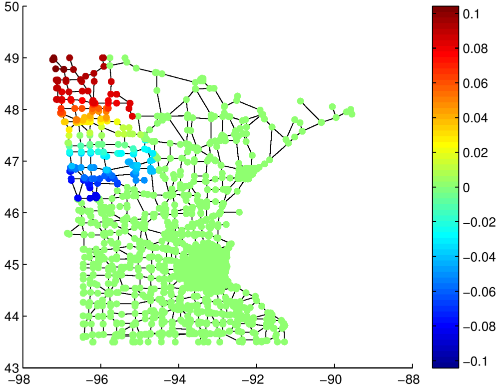
Level $j = 2$, Region $k = 1$, $\boldsymbol{\phi}^2_{1,2}$



HGLET Basis Vectors on MN

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

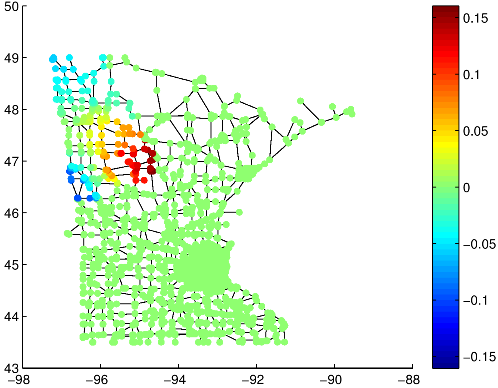
Level $j = 3$, Region $k = 0$, $\phi^3_{0,1}$



HGLET Basis Vectors on MN

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

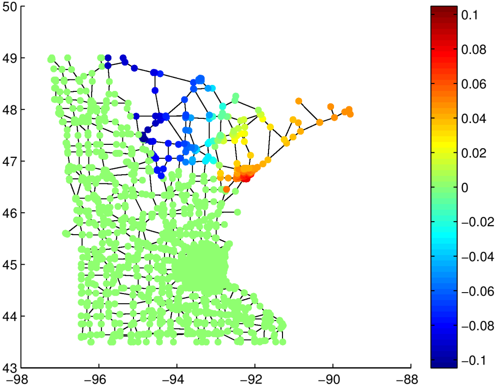
Level $j=3$, Region $k=0$, $\boldsymbol{\phi}^{3}_{0,2}$



HGLET Basis Vectors on MN

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

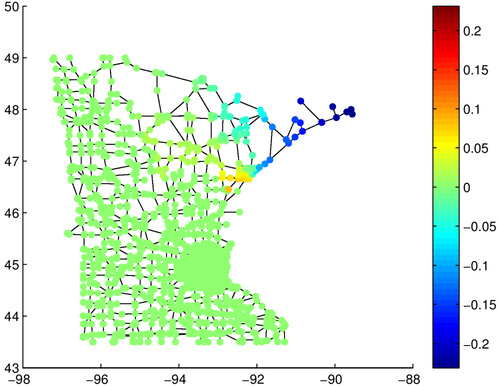
Level $j = 3$, Region $k = 1$, $\boldsymbol{\phi}^3_{1,1}$



HGLET Basis Vectors on MN

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

Level $j = 3$, Region $k = 1$, $\boldsymbol{\phi}^3_{1,2}$



Computational Complexity: HGLET

	Computational Complexity	Run Time for MN[1]
HGLET (redundant)	$O(N^3)$	83 sec

[1] Computations performed on a personal laptop (4.00 GB RAM, 2.26 GHz), $N = 2640$ and `nnz(W)` $= 6604$.

Now we present a Haar-like modification of our scheme:

1. Starting with the entire graph (i.e., level $j = 0$), compute the Fiedler vector $\boldsymbol{\phi}_1$ ($\boldsymbol{\phi}_0$ is trivially known, and we denote it by $\varphi_{0,0}$). Convert $\boldsymbol{\phi}_1$ to a Haar-like vector:[1]

$$\psi_{0,0}(i) := \begin{cases} 1 & \text{if } \boldsymbol{\phi}_1(i) \geq 0 \\ -\frac{\#\text{ nonnegative}}{\#\text{ negative}} & \text{if } \boldsymbol{\phi}_1(i) < 0 \end{cases}$$

   and then ℓ^2-normalize it
2. Partition the graph ⇒ Fiedler vector
3. Compute the Fiedler vector for each partition and convert it to a Haar-like vector on its respective partition[1] ⇒ $\psi_{j,k}$
4. Repeat...

This yields an orthonormal basis: $\varphi_{0,0} \cup \{\psi_{j,k}\}_{0 \leq j < J,\ k}$

[1] As with the HGLET, we could generate a full orthonormal basis by converting all the Laplacian eigenvectors into piecewise-constant orthonormal vectors according to their sign, similar to the *Walsh-Hadamard transform*.

Now we present a Haar-like modification of our scheme:

1. Starting with the entire graph (i.e., level $j = 0$), compute the Fiedler vector $\boldsymbol{\phi}_1$ ($\boldsymbol{\phi}_0$ is trivially known, and we denote it by $\varphi_{0,0}$). Convert $\boldsymbol{\phi}_1$ to a Haar-like vector:[1]

$$\psi_{0,0}(i) := \begin{cases} 1 & \text{if } \boldsymbol{\phi}_1(i) \geq 0 \\ -\frac{\#\text{ nonnegative}}{\#\text{ negative}} & \text{if } \boldsymbol{\phi}_1(i) < 0 \end{cases}$$

and then ℓ^2-normalize it

2. Partition the graph ⇒ Fiedler vector
3. Compute the Fiedler vector for each partition and convert it to a Haar-like vector on its respective partition[1] ⇒ $\psi_{j,k}$
4. Repeat...

This yields an orthonormal basis: $\varphi_{0,0} \cup \{\psi_{j,k}\}_{0 \leq j < J,\ k}$

[1] As with the HGLET, we could generate a full orthonormal basis by converting all the Laplacian eigenvectors into piecewise-constant orthonormal vectors according to their sign, similar to the *Walsh-Hadamard transform*.

Now we present a Haar-like modification of our scheme:

1. Starting with the entire graph (i.e., level $j = 0$), compute the Fiedler vector $\boldsymbol{\phi}_1$ ($\boldsymbol{\phi}_0$ is trivially known, and we denote it by $\varphi_{0,0}$). Convert $\boldsymbol{\phi}_1$ to a Haar-like vector:[1]

$$\psi_{0,0}(i) := \begin{cases} 1 & \text{if } \boldsymbol{\phi}_1(i) \geq 0 \\ -\frac{\#\ \text{nonnegative}}{\#\ \text{negative}} & \text{if } \boldsymbol{\phi}_1(i) < 0 \end{cases}$$

   and then ℓ^2-normalize it

2. Partition the graph ⇒ Fiedler vector
3. Compute the Fiedler vector for each partition and convert it to a Haar-like vector on its respective partition[1] ⇒ $\psi_{j,k}$
4. Repeat...

This yields an orthonormal basis: $\varphi_{0,0} \cup \{\psi_{j,k}\}_{0 \leq j < J,\ k}$

[1] As with the HGLET, we could generate a full orthonormal basis by converting all the Laplacian eigenvectors into piecewise-constant orthonormal vectors according to their sign, similar to the *Walsh-Hadamard transform*.

Now we present a Haar-like modification of our scheme:

1. Starting with the entire graph (i.e., level $j = 0$), compute the Fiedler vector $\boldsymbol{\phi}_1$ ($\boldsymbol{\phi}_0$ is trivially known, and we denote it by $\varphi_{0,0}$). Convert $\boldsymbol{\phi}_1$ to a Haar-like vector:[1]

$$\psi_{0,0}(i) := \begin{cases} 1 & \text{if } \boldsymbol{\phi}_1(i) \geq 0 \\ -\frac{\#\text{ nonnegative}}{\#\text{ negative}} & \text{if } \boldsymbol{\phi}_1(i) < 0 \end{cases}$$

   and then ℓ^2-normalize it

2. Partition the graph $\Rightarrow$ Fiedler vector
3. Compute the Fiedler vector for each partition and convert it to a Haar-like vector on its respective partition[1] $\Rightarrow \psi_{j,k}$
4. Repeat...

This yields an orthonormal basis: $\varphi_{0,0} \cup \{\psi_{j,k}\}_{0 \leq j < J,\ k}$

[1] As with the HGLET, we could generate a full orthonormal basis by converting all the Laplacian eigenvectors into piecewise-constant orthonormal vectors according to their sign, similar to the *Walsh-Hadamard transform*.

Now we present a Haar-like modification of our scheme:

1. Starting with the entire graph (i.e., level $j = 0$), compute the Fiedler vector $\boldsymbol{\phi}_1$ ($\boldsymbol{\phi}_0$ is trivially known, and we denote it by $\varphi_{0,0}$). Convert $\boldsymbol{\phi}_1$ to a Haar-like vector:[1]

$$\psi_{0,0}(i) := \begin{cases} 1 & \text{if } \boldsymbol{\phi}_1(i) \geq 0 \\ -\frac{\#\ \text{nonnegative}}{\#\ \text{negative}} & \text{if } \boldsymbol{\phi}_1(i) < 0 \end{cases}$$

   and then ℓ^2-normalize it

2. Partition the graph ⇒ Fiedler vector
3. Compute the Fiedler vector for each partition and convert it to a Haar-like vector on its respective partition[1] ⇒ $\psi_{j,k}$
4. Repeat...

This yields an orthonormal basis: $\varphi_{0,0} \cup \{\psi_{j,k}\}_{0 \leq j < J,\ k}$

[1] As with the HGLET, we could generate a full orthonormal basis by converting all the Laplacian eigenvectors into piecewise-constant orthonormal vectors according to their sign, similar to the *Walsh-Hadamard transform*.

Now we present a Haar-like modification of our scheme:

1. Starting with the entire graph (i.e., level $j = 0$), compute the Fiedler vector $\boldsymbol{\phi}_1$ ($\boldsymbol{\phi}_0$ is trivially known, and we denote it by $\varphi_{0,0}$). Convert $\boldsymbol{\phi}_1$ to a Haar-like vector:[1]

$$\psi_{0,0}(i) := \begin{cases} 1 & \text{if } \boldsymbol{\phi}_1(i) \ge 0 \\ -\frac{\#\text{ nonnegative}}{\#\text{ negative}} & \text{if } \boldsymbol{\phi}_1(i) < 0 \end{cases}$$

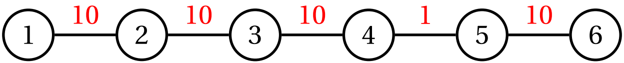
   and then ℓ^2-normalize it

2. Partition the graph ⇒ Fiedler vector
3. Compute the Fiedler vector for each partition and convert it to a Haar-like vector on its respective partition[1] ⇒ $\psi_{j,k}$
4. Repeat...

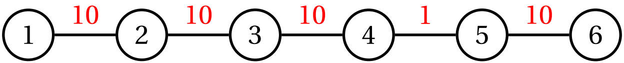
This yields an orthonormal basis: $\varphi_{0,0} \cup \{\psi_{j,k}\}_{0 \le j < J,\ k}$

[1] As with the HGLET, we could generate a full orthonormal basis by converting all the Laplacian eigenvectors into piecewise-constant orthonormal vectors according to their sign, similar to the *Walsh-Hadamard transform*.

HGLET Haar-like Basis Example

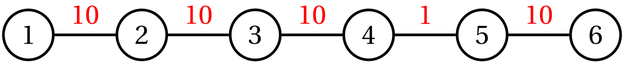


HGLET Haar-like Basis Example



$$\left[\ \varphi_{0,0}\quad \psi_{0,0} \qquad\qquad\qquad\qquad\qquad\qquad \right]$$

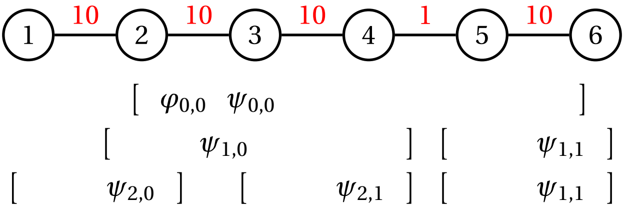
HGLET Haar-like Basis Example



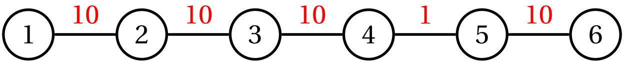
$[\ \varphi_{0,0} \quad \psi_{0,0} \qquad\qquad\qquad\qquad]$

$[\qquad \psi_{1,0} \qquad\qquad] \ [\qquad \psi_{1,1} \ ]$

HGLET Haar-like Basis Example



HGLET Haar-like Basis Example



$[\ \varphi_{0,0}\ \ \psi_{0,0}\ ]$

$[\ \psi_{1,0}\ ]\ [\ \psi_{1,1}\ ]$

$[\ \psi_{2,0}\ ]\ [\ \psi_{2,1}\ ]\ [\ \psi_{1,1}\ ]$

Thus, we generate a matrix whose columns (after ℓ^2-normalization) form an orthonormal basis:

$\varphi_{0,0}$	$\psi_{0,0}$	$\psi_{1,0}$	$\psi_{1,1}$	$\psi_{2,0}$	$\psi_{2,1}$
1	1	1	0	1	0
1	1	1	0	-1	0
1	1	-1	0	0	1
1	1	-1	0	0	-1
1	-2	0	1	0	0
1	-2	0	-1	0	0

Computational Complexity: Haar-like HGLET

	Computational Complexity	Run Time for MN[1]
HGLET (redundant)	$O(N^3)$	83 sec
Haar-like HGLET	$O(N \log N)$	5 sec

[1] Computations performed on a personal laptop (4.00 GB RAM, 2.26 GHz), $N = 2640$ and `nnz(W)` $= 6604$.

We have also developed and implemented a modification that is similar to the Haar-like HGLET, but yields a smoother set of orthonormal basis functions. We call this the Orthonormalized Hierarchical Fiedler Transform (OHFT).

1. Starting with the entire graph (i.e., level $j = 0$), compute the Fiedler vector ϕ_1 and denote it as $\psi_{0,0}$ (ϕ_0 is trivially known, and we denote it by $\varphi_{0,0}$)[1]
2. Partition the graph $\Rightarrow$ Fiedler vector
3. Compute the Fiedler vector for each partition and orthonormalize it against all $\psi_{j,k}$'s computed thus far (it is already orthogonal to $\varphi_{0,0}$)[1] $\Rightarrow \psi_{j,k}$
4. Repeat...

This yields an orthonormal basis: $\varphi_{0,0} \cup \{\psi_{j,k}\}_{0 \le j < J,\ k}$

[1] As with the HGLET, we could generate a full orthonormal basis. However, this would require computing all of the eigenvectors, and so we do not perform this step. But we point this out to show consistency with the HGLET.

We have also developed and implemented a modification that is similar to the Haar-like HGLET, but yields a smoother set of orthonormal basis functions. We call this the Orthonormalized Hierarchical Fiedler Transform (OHFT).

1. Starting with the entire graph (i.e., level $j = 0$), compute the Fiedler vector $\boldsymbol{\phi}_1$ and denote it as $\psi_{0,0}$ ($\boldsymbol{\phi}_0$ is trivially known, and we denote it by $\varphi_{0,0}$)[1]
2. Partition the graph $\Rightarrow$ Fiedler vector
3. Compute the Fiedler vector for each partition and orthonormalize it against all $\psi_{j,k}$'s computed thus far (it is already orthogonal to $\varphi_{0,0}$)[1] $\Rightarrow \psi_{j,k}$
4. Repeat...

This yields an orthonormal basis: $\varphi_{0,0} \cup \{\psi_{j,k}\}_{0 \le j < J,\ k}$

[1] As with the HGLET, we could generate a full orthonormal basis. However, this would require computing all of the eigenvectors, and so we do not perform this step. But we point this out to show consistency with the HGLET.

We have also developed and implemented a modification that is similar to the Haar-like HGLET, but yields a smoother set of orthonormal basis functions. We call this the Orthonormalized Hierarchical Fiedler Transform (OHFT).

1. Starting with the entire graph (i.e., level $j = 0$), compute the Fiedler vector $\boldsymbol{\phi}_1$ and denote it as $\psi_{0,0}$ ($\boldsymbol{\phi}_0$ is trivially known, and we denote it by $\varphi_{0,0}$)[1]
2. Partition the graph $\Rightarrow$ Fiedler vector
3. Compute the Fiedler vector for each partition and orthonormalize it against all $\psi_{j,k}$'s computed thus far (it is already orthogonal to $\varphi_{0,0}$)[1] $\Rightarrow \psi_{j,k}$
4. Repeat...

This yields an orthonormal basis: $\varphi_{0,0} \cup \{\psi_{j,k}\}_{0 \le j < J,\ k}$

[1] As with the HGLET, we could generate a full orthonormal basis. However, this would require computing all of the eigenvectors, and so we do not perform this step. But we point this out to show consistency with the HGLET.

We have also developed and implemented a modification that is similar to the Haar-like HGLET, but yields a smoother set of orthonormal basis functions. We call this the Orthonormalized Hierarchical Fiedler Transform (OHFT).

1. Starting with the entire graph (i.e., level $j=0$), compute the Fiedler vector $\boldsymbol{\phi}_1$ and denote it as $\psi_{0,0}$ ($\boldsymbol{\phi}_0$ is trivially known, and we denote it by $\varphi_{0,0}$)[1]
2. Partition the graph $\Rightarrow$ Fiedler vector
3. Compute the Fiedler vector for each partition and orthonormalize it against all $\psi_{j,k}$'s computed thus far (it is already orthogonal to $\varphi_{0,0}$)[1] $\Rightarrow \psi_{j,k}$
4. Repeat...

This yields an orthonormal basis: $\varphi_{0,0} \cup \{\psi_{j,k}\}_{0\le j<J,\ k}$

[1] As with the HGLET, we could generate a full orthonormal basis. However, this would require computing all of the eigenvectors, and so we do not perform this step. But we point this out to show consistency with the HGLET.

We have also developed and implemented a modification that is similar to the Haar-like HGLET, but yields a smoother set of orthonormal basis functions. We call this the Orthonormalized Hierarchical Fiedler Transform (OHFT).

1. Starting with the entire graph (i.e., level $j=0$), compute the Fiedler vector $\boldsymbol{\phi}_1$ and denote it as $\psi_{0,0}$ ($\boldsymbol{\phi}_0$ is trivially known, and we denote it by $\varphi_{0,0}$)[1]
2. Partition the graph $\Rightarrow$ Fiedler vector
3. Compute the Fiedler vector for each partition and orthonormalize it against all $\psi_{j,k}$'s computed thus far (it is already orthogonal to $\varphi_{0,0}$)[1] $\Rightarrow$ $\psi_{j,k}$
4. Repeat...

This yields an orthonormal basis: $\varphi_{0,0} \cup \{\psi_{j,k}\}_{0 \le j < J,\ k}$

[1] As with the HGLET, we could generate a full orthonormal basis. However, this would require computing all of the eigenvectors, and so we do not perform this step. But we point this out to show consistency with the HGLET.

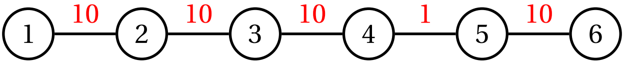
We have also developed and implemented a modification that is similar to the Haar-like HGLET, but yields a smoother set of orthonormal basis functions. We call this the Orthonormalized Hierarchical Fiedler Transform (OHFT).

1. Starting with the entire graph (i.e., level $j = 0$), compute the Fiedler vector $\boldsymbol{\phi}_1$ and denote it as $\psi_{0,0}$ ($\boldsymbol{\phi}_0$ is trivially known, and we denote it by $\varphi_{0,0}$)[1]
2. Partition the graph $\Rightarrow$ Fiedler vector
3. Compute the Fiedler vector for each partition and orthonormalize it against all $\psi_{j,k}$'s computed thus far (it is already orthogonal to $\varphi_{0,0}$)[1] $\Rightarrow \psi_{j,k}$
4. Repeat...

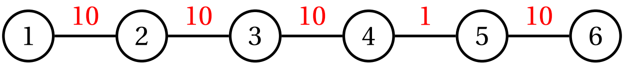
This yields an orthonormal basis: $\varphi_{0,0} \cup \{\psi_{j,k}\}_{0 \le j < J,\ k}$

[1] As with the HGLET, we could generate a full orthonormal basis. However, this would require computing all of the eigenvectors, and so we do not perform this step. But we point this out to show consistency with the HGLET.

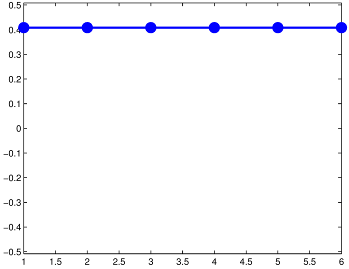
Haar-like HGLET vs. OHFT



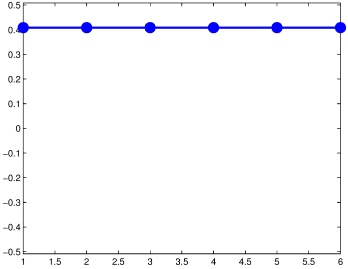
Haar-like HGLET vs. OHFT



$\varphi_{0,0}$ is the same in both cases: a global constant vector.

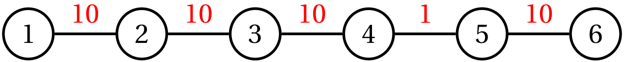


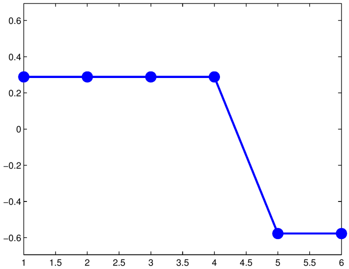
Haar-like $\varphi_{0,0}$



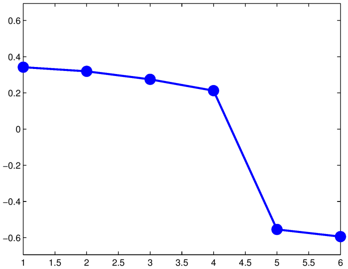
OHFT $\varphi_{0,0}$

Haar-like HGLET vs. OHFT



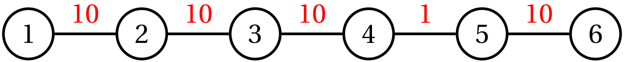


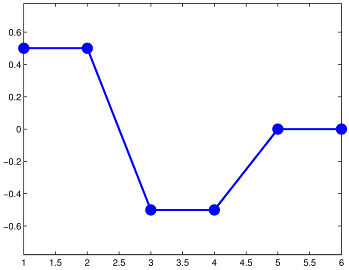
Haar-like $\psi_{0,0}$



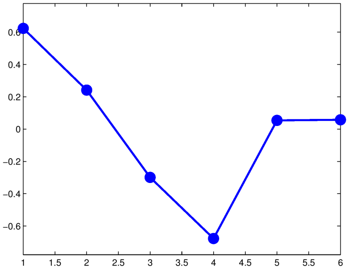
OHFT $\psi_{0,0}$

Haar-like HGLET vs. OHFT



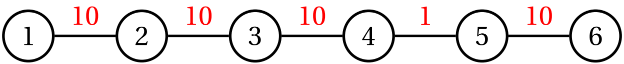


Haar-like $\psi_{1,0}$

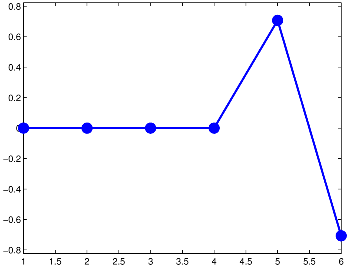


OHFT $\psi_{1,0}$

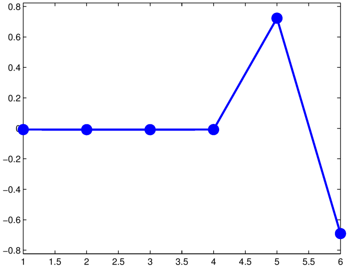
Haar-like HGLET vs. OHFT



(These vectors look the same, but they are not.)

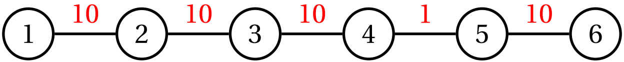


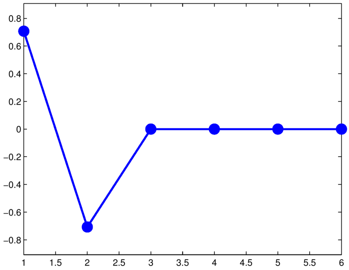
Haar-like $\psi_{1,1}$



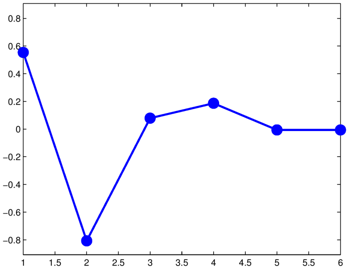
OHFT $\psi_{1,1}$

Haar-like HGLET vs. OHFT



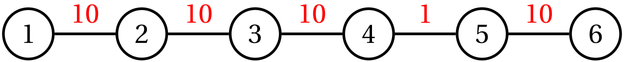


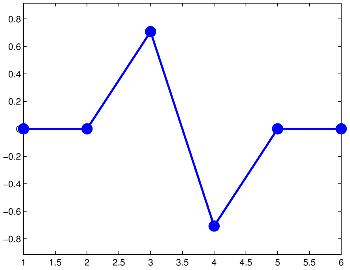
Haar-like $\psi_{2,0}$



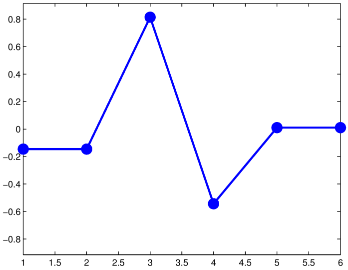
OHFT $\psi_{2,0}$

Haar-like HGLET vs. OHFT





Haar-like $\psi_{2,1}$



OHFT $\psi_{2,1}$

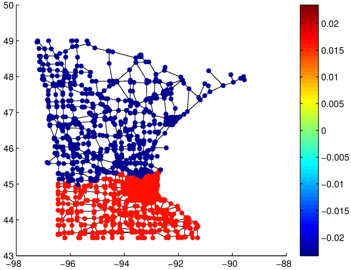
Haar-like HGLET vs. OHFT

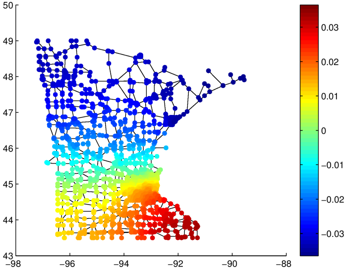
Now we compare the basis functions they generate on the MN road network.

Haar-like HGLET vs. OHFT

Now we compare the basis functions they generate on the MN road network.

$$\psi_{0,0}$$

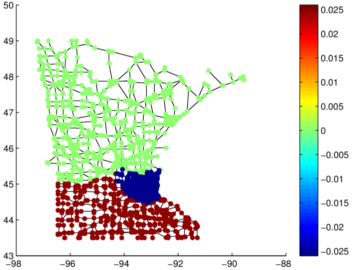


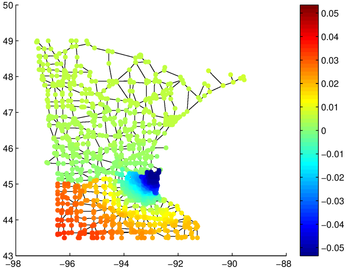


Haar-like HGLET vs. OHFT

Now we compare the basis functions they generate on the MN road network.

$$\psi_{1,0}$$

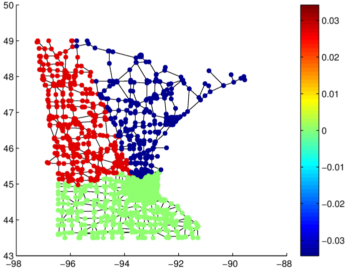


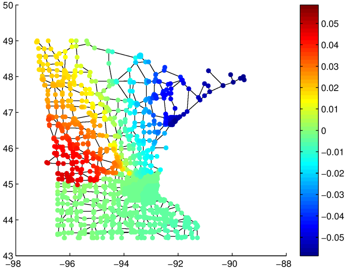


Haar-like HGLET vs. OHFT

Now we compare the basis functions they generate on the MN road network.

$$\psi_{1,1}$$

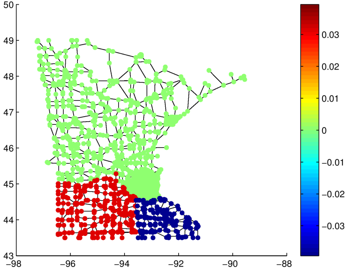


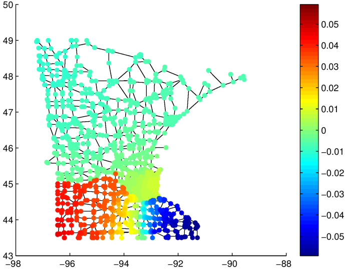


Haar-like HGLET vs. OHFT

Now we compare the basis functions they generate on the MN road network.

$$\psi_{2,0}$$

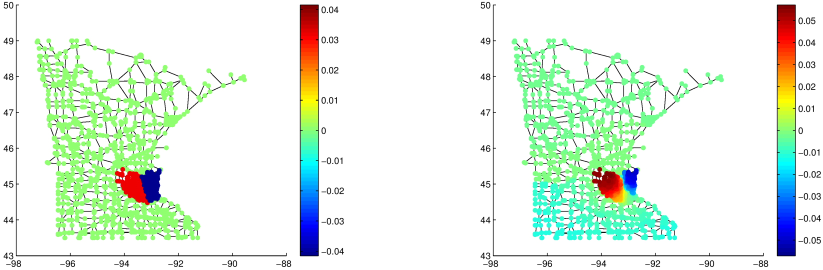




Haar-like HGLET vs. OHFT

Now we compare the basis functions they generate on the MN road network.

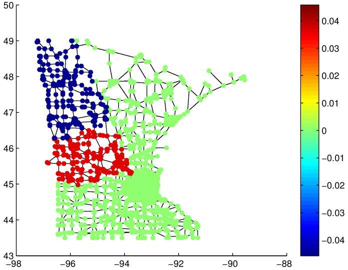
$$\psi_{2,1}$$

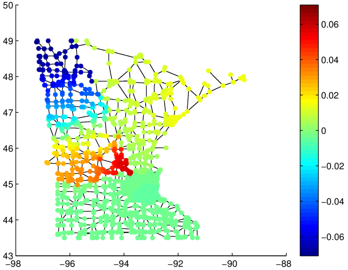


Haar-like HGLET vs. OHFT

Now we compare the basis functions they generate on the MN road network.

$\psi_{2,2}$

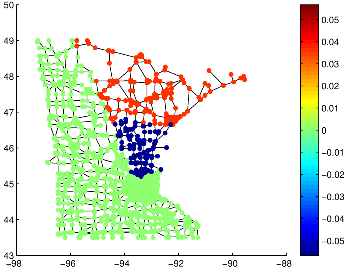


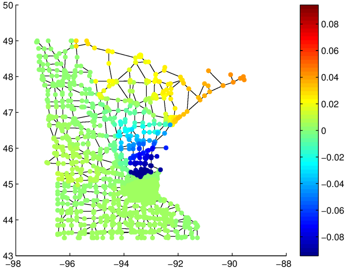


Haar-like HGLET vs. OHFT

Now we compare the basis functions they generate on the MN road network.

$$\psi_{2,3}$$



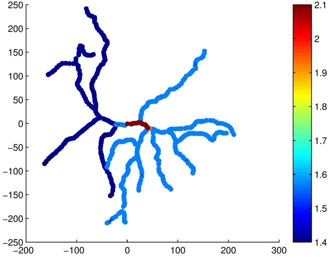


Computational Complexity: OHFT

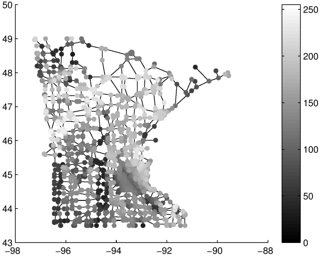
	Computational Complexity	Run Time for MN[1]
HGLET (redundant)	$O(N^3)$	83 sec
Haar-like HGLET	$O(N \log N)$	5 sec
OHFT	$O(N^3)$	8 sec

[1] Computations performed on a personal laptop (4.00 GB RAM, 2.26 GHz), $N = 2640$ and `nnz(W)` $= 6604$.

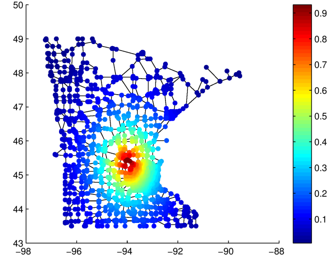
We have performed some preliminary approximation experiments on the following datasets...



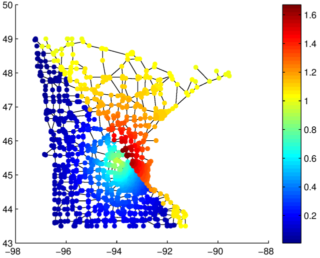
(a) Thickness data on dendritic tree #100



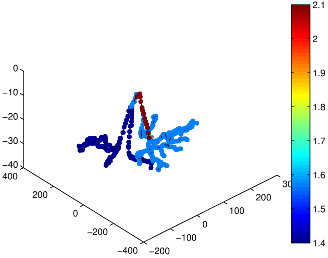
(b) The pixels of the Barbara image mapped to the MN road network



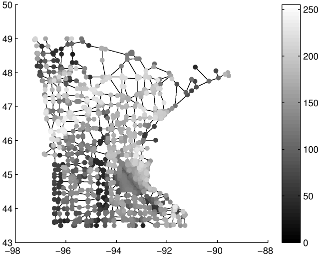
(c) A Gaussian on the MN road network



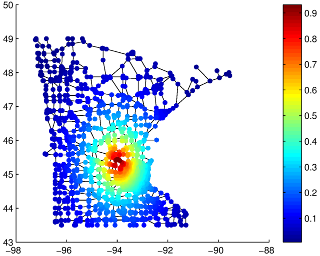
(d) A mutilated Gaussian on the MN road network



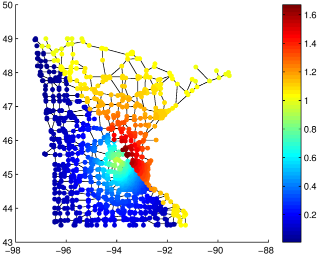
(a) Thickness data on dendritic tree #100



(b) The pixels of the Barbara image mapped to the MN road network



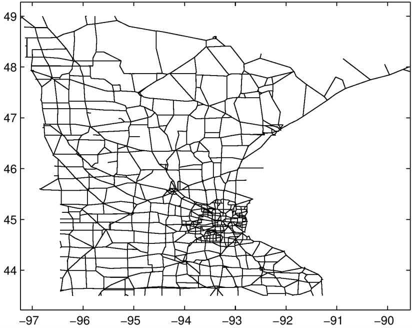
(c) A Gaussian on the MN road network



(d) A mutilated Gaussian on the MN road network

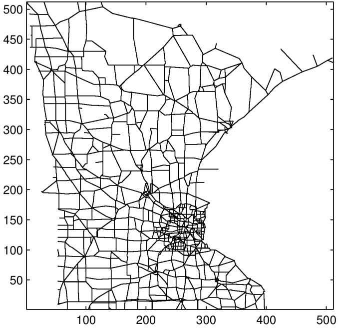
Explanation of Barbara on MN Road Network

The Barbara image (512×512) and the MN road network (2640 nodes)



Explanation of Barbara on MN Road Network

1. Stretch the MN road network so that it is on a $[1,512] \times [1,512]$ grid

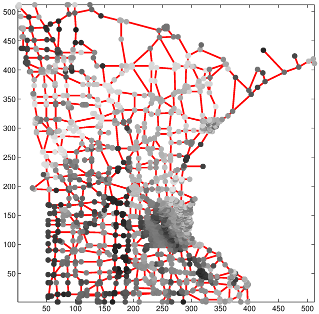


Explanation of Barbara on MN Road Network

2. Superimpose the stretched MN road network onto Barbara

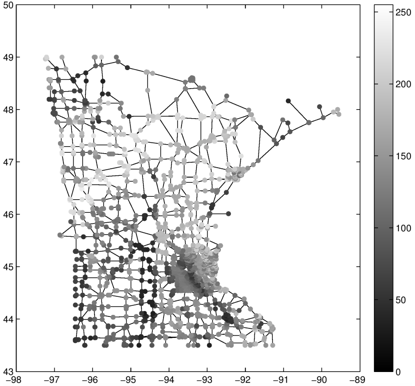
Explanation of Barbara on MN Road Network

3. Set each node value to be the nearest pixel value

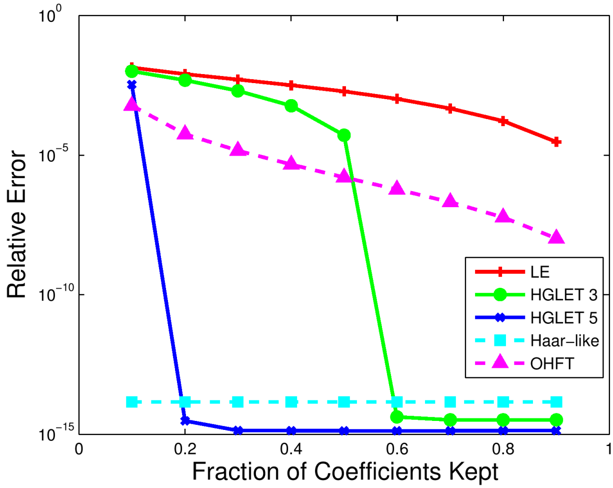


Explanation of Barbara on MN Road Network

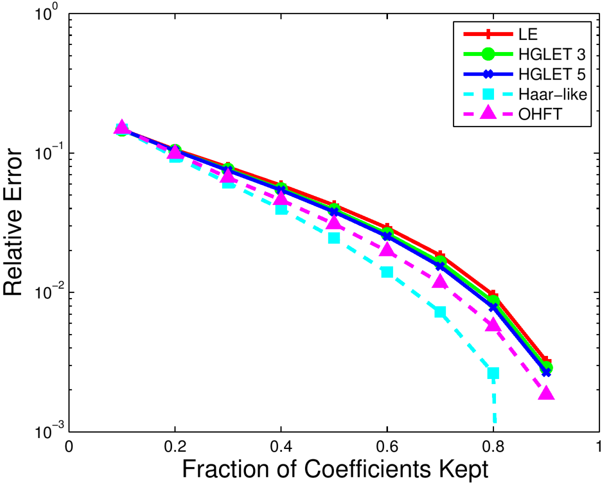
Barbara on the original MN road network



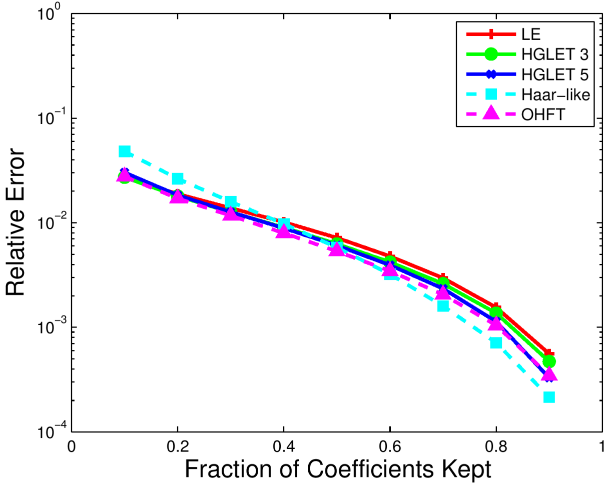
Approximation Results for Dendrite #100



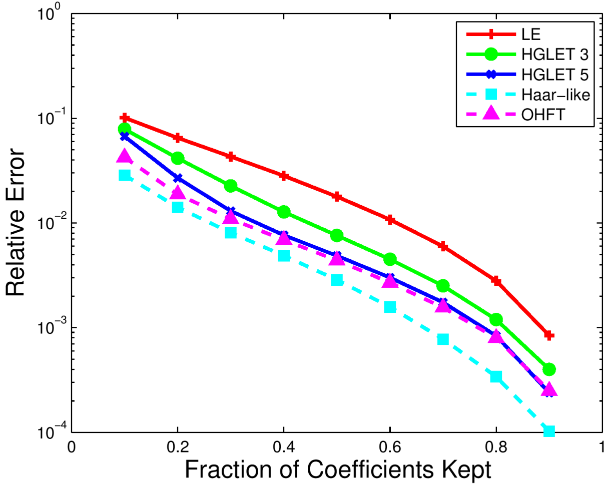
Approximation Results for MN Barbara



Approximation Results for MN Gaussian



Approximation Results for MN Mutilated Gaussian



Discussion of Approximation Results

- Overall, the Haar-like HGLET variation was the best performer, followed by the OHFT. This makes a strong case for using *localized basis functions on multiple scales*.
- Level 5 of the HGLET outperforms Level 3. Both outperform Laplacian eigenvectors (i.e., HGLET Level 0). Again, this demonstrates the merit of using localized basis vectors. Future work will investigate the advantages of using a basis comprised of HGLET vectors from multiple levels.
- Haar-like HGLET vs. OHFT
  - The basis vectors for both are derived from the same Fiedler vectors ⇒ convert to a Haar-like vector vs. orthonormalize against pre-existing basis vectors
  - The OHFT offers a compromise between the localization of the Haar-like HGLET and the smoothness of the HGLET (including Laplacian eigenvectors)
    - This explains why the Haar-like HGLET performs better for the dendrite #100 data (piecewise constant), while the OHFT performs better for $<50\%$ coefficients kept on the MN Gaussian data (smooth)

Discussion of Approximation Results

- Overall, the Haar-like HGLET variation was the best performer, followed by the OHFT. This makes a strong case for using *localized basis functions on multiple scales*.
- Level 5 of the HGLET outperforms Level 3. Both outperform Laplacian eigenvectors (i.e., HGLET Level 0). Again, this demonstrates the merit of using localized basis vectors. Future work will investigate the advantages of using a basis comprised of HGLET vectors from multiple levels.
- Haar-like HGLET vs. OHFT
  - The basis vectors for both are derived from the same Fiedler vectors $\Rightarrow$ convert to a Haar-like vector vs. orthonormalize against pre-existing basis vectors
  - The OHFT offers a compromise between the localization of the Haar-like HGLET and the smoothness of the HGLET (including Laplacian eigenvectors)
    - This explains why the Haar-like HGLET performs better for the dendrite #100 data (piecewise constant), while the OHFT performs better for <50% coefficients kept on the MN Gaussian data (smooth)

Discussion of Approximation Results

- Overall, the Haar-like HGLET variation was the best performer, followed by the OHFT. This makes a strong case for using *localized basis functions on multiple scales*.
- Level 5 of the HGLET outperforms Level 3. Both outperform Laplacian eigenvectors (i.e., HGLET Level 0). Again, this demonstrates the merit of using localized basis vectors. Future work will investigate the advantages of using a basis comprised of HGLET vectors from multiple levels.
- Haar-like HGLET vs. OHFT
  - The basis vectors for both are derived from the same Fiedler vectors ⇒ convert to a Haar-like vector vs. orthonormalize against pre-existing basis vectors
  - The OHFT offers a compromise between the localization of the Haar-like HGLET and the smoothness of the HGLET (including Laplacian eigenvectors)
    - This explains why the Haar-like HGLET performs better for the dendrite #100 data (piecewise constant), while the OHFT performs better for $<50\%$ coefficients kept on the MN Gaussian data (smooth)

Discussion of Approximation Results

- Overall, the Haar-like HGLET variation was the best performer, followed by the OHFT. This makes a strong case for using *localized basis functions on multiple scales*.
- Level 5 of the HGLET outperforms Level 3. Both outperform Laplacian eigenvectors (i.e., HGLET Level 0). Again, this demonstrates the merit of using localized basis vectors. Future work will investigate the advantages of using a basis comprised of HGLET vectors from multiple levels.
- Haar-like HGLET vs. OHFT
  - The basis vectors for both are derived from the same Fiedler vectors $\Rightarrow$ convert to a Haar-like vector vs. orthonormalize against pre-existing basis vectors
  - The OHFT offers a compromise between the localization of the Haar-like HGLET and the smoothness of the HGLET (including Laplacian eigenvectors)
    - This explains why the Haar-like HGLET performs better for the dendrite #100 data (piecewise constant), while the OHFT performs better for $< 50\%$ coefficients kept on the MN Gaussian data (smooth)

Bonus: Simultaneous Signal Segmentation & Compression

- As a bonus, we can apply the HGLET for simultaneously segmenting and compressing a given *nonstationary regularly-sampled signal*.
- Our proposed procedure is:
  1. Form a graph of a given signal by associating each vertex (i.e., the signal sample location) with a set of signal amplitude at that vertex and those of its *local neighbors* (e.g., 3 or 5 points around it);
  2. Compute the graph Laplacian matrix and *the Fiedler vector*;
  3. Segment the signal based on the polarity of the Fiedler vector;
  4. In each segment, apply the *standard* DCT;
  5. Store the compressed coefficients and the segment location info.
- Of course, one can use more sophisticated *feature vectors* instead of the local samples at each vertex; also can use a few more eigenvectors for the segmentation above.

Bonus: Simultaneous Signal Segmentation & Compression

- As a bonus, we can apply the HGLET for simultaneously segmenting and compressing a given *nonstationary regularly-sampled signal*.
- Our proposed procedure is:
  1. Form a graph of a given signal by associating each vertex (i.e., the signal sample location) with a set of signal amplitude at that vertex and those of its *local neighbors* (e.g., 3 or 5 points around it);
  2. Compute the graph Laplacian matrix and *the Fiedler vector*;
  3. Segment the signal based on the polarity of the Fiedler vector;
  4. In each segment, apply the *standard* DCT;
  5. Store the compressed coefficients and the segment location info.
- Of course, one can use more sophisticated *feature vectors* instead of the local samples at each vertex; also can use a few more eigenvectors for the segmentation above.

Bonus: Simultaneous Signal Segmentation & Compression

- As a bonus, we can apply the HGLET for simultaneously segmenting and compressing a given *nonstationary regularly-sampled signal*.
- Our proposed procedure is:
  1. Form a graph of a given signal by associating each vertex (i.e., the signal sample location) with a set of signal amplitude at that vertex and those of its *local neighbors* (e.g., 3 or 5 points around it);
  2. Compute the graph Laplacian matrix and *the Fiedler vector*;
  3. Segment the signal based on the polarity of the Fiedler vector;
  4. In each segment, apply the *standard* DCT;
  5. Store the compressed coefficients and the segment location info.
- Of course, one can use more sophisticated *feature vectors* instead of the local samples at each vertex; also can use a few more eigenvectors for the segmentation above.

Bonus: Simultaneous Signal Segmentation & Compression

- As a bonus, we can apply the HGLET for simultaneously segmenting and compressing a given *nonstationary regularly-sampled signal*.
- Our proposed procedure is:
  1. Form a graph of a given signal by associating each vertex (i.e., the signal sample location) with a set of signal amplitude at that vertex and those of its *local neighbors* (e.g., 3 or 5 points around it);
  2. Compute the graph Laplacian matrix and *the Fiedler vector*;
  3. Segment the signal based on the polarity of the Fiedler vector;
  4. In each segment, apply the *standard* DCT;
  5. Store the compressed coefficients and the segment location info.
- Of course, one can use more sophisticated *feature vectors* instead of the local samples at each vertex; also can use a few more eigenvectors for the segmentation above.

Bonus: Simultaneous Signal Segmentation & Compression

- As a bonus, we can apply the HGLET for simultaneously segmenting and compressing a given *nonstationary regularly-sampled signal*.
- Our proposed procedure is:
  1. Form a graph of a given signal by associating each vertex (i.e., the signal sample location) with a set of signal amplitude at that vertex and those of its *local neighbors* (e.g., 3 or 5 points around it);
  2. Compute the graph Laplacian matrix and *the Fiedler vector*;
  3. Segment the signal based on the polarity of the Fiedler vector;
  4. In each segment, apply the *standard* DCT;
  5. Store the compressed coefficients and the segment location info.
- Of course, one can use more sophisticated *feature vectors* instead of the local samples at each vertex; also can use a few more eigenvectors for the segmentation above.

Bonus: Simultaneous Signal Segmentation & Compression

- As a bonus, we can apply the HGLET for simultaneously segmenting and compressing a given *nonstationary regularly-sampled signal*.
- Our proposed procedure is:
  1. Form a graph of a given signal by associating each vertex (i.e., the signal sample location) with a set of signal amplitude at that vertex and those of its *local neighbors* (e.g., 3 or 5 points around it);
  2. Compute the graph Laplacian matrix and *the Fiedler vector*;
  3. Segment the signal based on the polarity of the Fiedler vector;
  4. In each segment, apply the *standard* DCT;
  5. Store the compressed coefficients and the segment location info.
- Of course, one can use more sophisticated *feature vectors* instead of the local samples at each vertex; also can use a few more eigenvectors for the segmentation above.

Bonus: Simultaneous Signal Segmentation & Compression

- As a bonus, we can apply the HGLET for simultaneously segmenting and compressing a given *nonstationary regularly-sampled signal*.
- Our proposed procedure is:
  1. Form a graph of a given signal by associating each vertex (i.e., the signal sample location) with a set of signal amplitude at that vertex and those of its *local neighbors* (e.g., 3 or 5 points around it);
  2. Compute the graph Laplacian matrix and *the Fiedler vector*;
  3. Segment the signal based on the polarity of the Fiedler vector;
  4. In each segment, apply the *standard* DCT;
  5. Store the compressed coefficients and the segment location info.
- Of course, one can use more sophisticated *feature vectors* instead of the local samples at each vertex; also can use a few more eigenvectors for the segmentation above.

Bonus: Simultaneous Signal Segmentation & Compression

- As a bonus, we can apply the HGLET for simultaneously segmenting and compressing a given *nonstationary regularly-sampled signal*.
- Our proposed procedure is:
  1. Form a graph of a given signal by associating each vertex (i.e., the signal sample location) with a set of signal amplitude at that vertex and those of its *local neighbors* (e.g., 3 or 5 points around it);
  2. Compute the graph Laplacian matrix and *the Fiedler vector*;
  3. Segment the signal based on the polarity of the Fiedler vector;
  4. In each segment, apply the *standard* DCT;
  5. Store the compressed coefficients and the segment location info.
- Of course, one can use more sophisticated *feature vectors* instead of the local samples at each vertex; also can use a few more eigenvectors for the segmentation above.

Preliminary Result

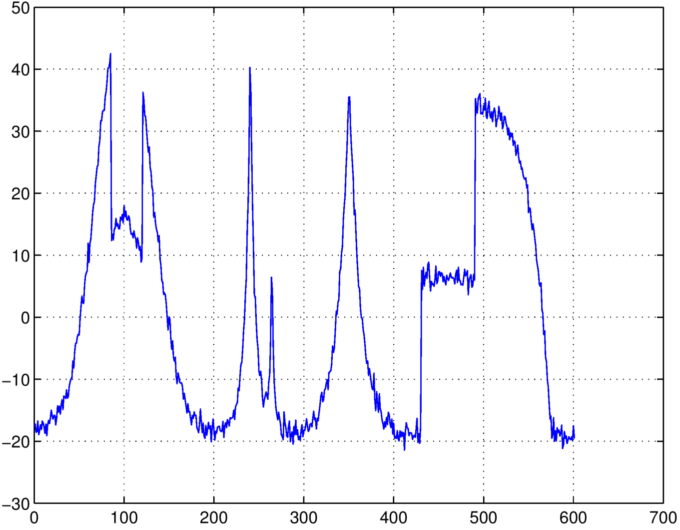


Figure: Noisy 'Piece-Regular' Signal from `WaveLab`

Preliminary Result

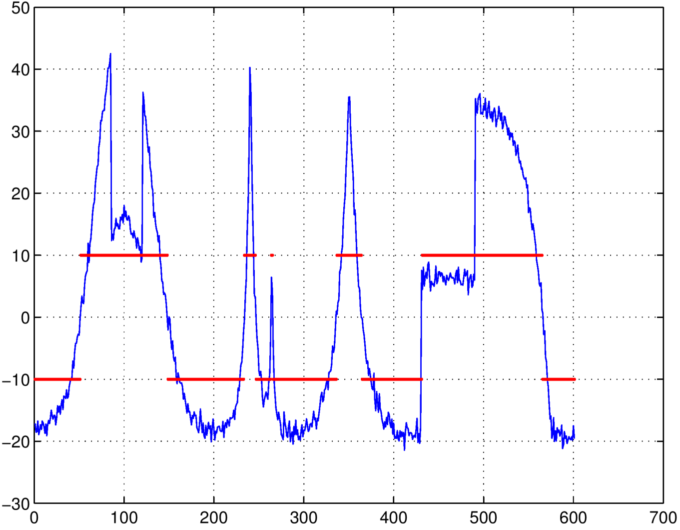


Figure: Segmentation intervals using the Fiedler vector

Preliminary Result

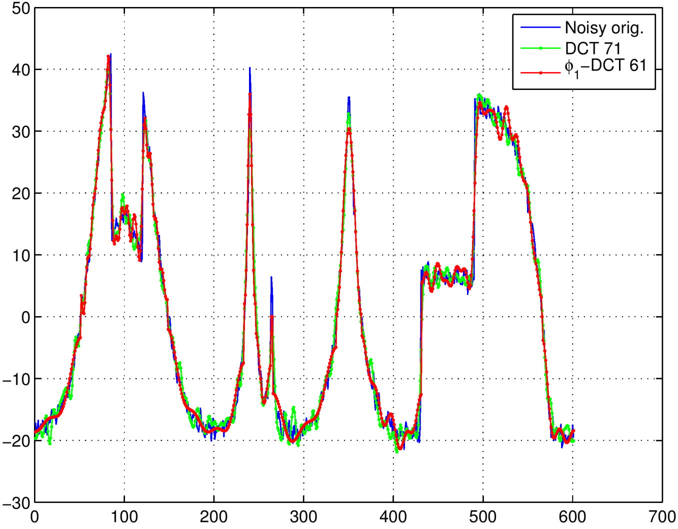


Figure: Approximation comparison

Preliminary Result

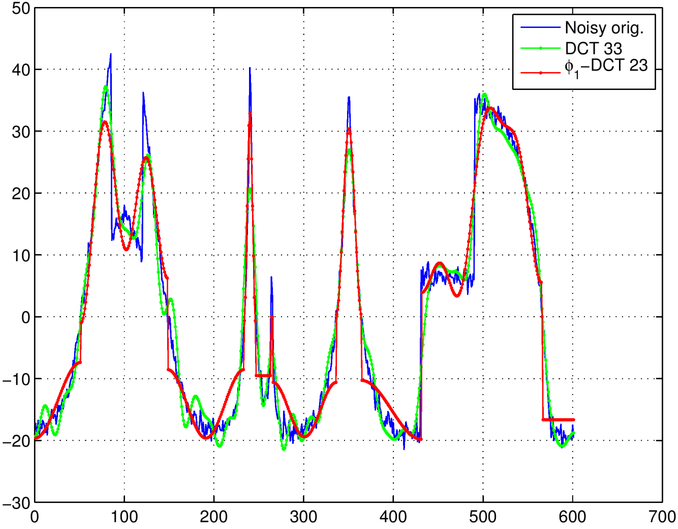


Figure: More concise approximations

Preliminary Result

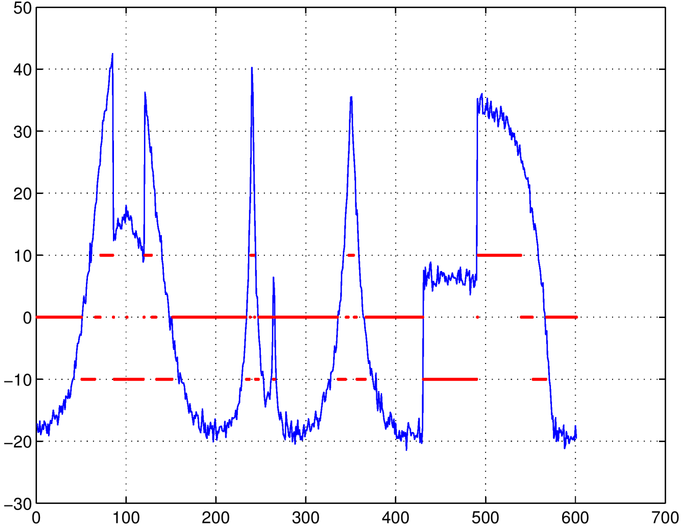


Figure: Segmentation using $\boldsymbol{\phi}_2$

Summary

- We developed a set of multiscale transforms on graphs and networks: HGLET; Haar-like HGLET; OHFT.
- They are direct generalizations of *Hierarchical Block Discrete Cosine Transforms* originally developed for regularly-sampled signals and images.
- They allow us to choose an orthonormal basis most suitable for one's task at hand, e.g., approximation, classification, regression, ...
- They may also be useful for regularly-sampled signals.
- Developing a *true* generalization of 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 a set of multiscale transforms on graphs and networks: HGLET; Haar-like HGLET; OHFT.
- They are direct generalizations of *Hierarchical Block Discrete Cosine Transforms* originally developed for regularly-sampled signals and images.
- They allow us to choose an orthonormal basis most suitable for one's task at hand, e.g., approximation, classification, regression, . . .
- They may also be useful for regularly-sampled signals.
- Developing a *true* generalization of 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 a set of multiscale transforms on graphs and networks: HGLET; Haar-like HGLET; OHFT.
- They are direct generalizations of *Hierarchical Block Discrete Cosine Transforms* originally developed for regularly-sampled signals and images.
- They allow us to choose an orthonormal basis most suitable for one's task at hand, e.g., approximation, classification, regression, . . .
- They may also be useful for regularly-sampled signals.
- Developing a *true* generalization of 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 a set of multiscale transforms on graphs and networks: HGLET; Haar-like HGLET; OHFT.
- They are direct generalizations of *Hierarchical Block Discrete Cosine Transforms* originally developed for regularly-sampled signals and images.
- They allow us to choose an orthonormal basis most suitable for one's task at hand, e.g., approximation, classification, regression, . . .
- They may also be useful for regularly-sampled signals.
- Developing a *true* generalization of 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 a set of multiscale transforms on graphs and networks: HGLET; Haar-like HGLET; OHFT.
- They are direct generalizations of *Hierarchical Block Discrete Cosine Transforms* originally developed for regularly-sampled signals and images.
- They allow us to choose an orthonormal basis most suitable for one's task at hand, e.g., approximation, classification, regression, ...
- They may also be useful for regularly-sampled signals.
- Developing a *true* generalization of 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

- Implement basis selection algorithms to be used in conjunction with the HGLET
  - **Approximation/Denoising** $\Rightarrow$ the best-basis algorithm of Coifman and Wickerhauser (1992)
  - **Classification** $\Rightarrow$ the local discriminant basis algorithms of Saito, Coifman, Geshwind, Warner, Marchand (1995, 2002, 2013)
- Perform classification experiments and compare the results using each of the 3 schemes presented herein
- 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

Future Work

- Implement basis selection algorithms to be used in conjunction with the HGLET
  - **Approximation/Denoising** ⇒ the best-basis algorithm of Coifman and Wickerhauser (1992)
  - **Classification** ⇒ the local discriminant basis algorithms of Saito, Coifman, Geshwind, Warner, Marchand (1995, 2002, 2013)
- Perform classification experiments and compare the results using each of the 3 schemes presented herein
- 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

Future Work

- Implement basis selection algorithms to be used in conjunction with the HGLET
  - **Approximation/Denoising** $\Rightarrow$ the best-basis algorithm of Coifman and Wickerhauser (1992)
  - **Classification** $\Rightarrow$ the local discriminant basis algorithms of Saito, Coifman, Geshwind, Warner, Marchand (1995, 2002, 2013)
- Perform classification experiments and compare the results using each of the 3 schemes presented herein
- 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

Future Work

- Implement basis selection algorithms to be used in conjunction with the HGLET
    - **Approximation/Denoising** ⇒ the best-basis algorithm of Coifman and Wickerhauser (1992)
    - **Classification** ⇒ the local discriminant basis algorithms of Saito, Coifman, Geshwind, Warner, Marchand (1995, 2002, 2013)
- Perform classification experiments and compare the results using each of the 3 schemes presented herein
- 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

References

- http://www.math.ucdavis.edu/~saito/courses/HarmGraph/ contains my course slides and useful information on "Harmonic Analysis on Graphs and Networks"
- http://www.math.ucdavis.edu/~saito/confs/ICIAM11/ contains talk slides of the minisymposium on Harmonic Analysis on Graphs and Networks, ICIAM 2011, Zürich (Organizers: NS, Mauro Maggioni)
- 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.
  - N. Saito & E. Woei: "On the phase transition phenomenon of graph Laplacian eigenfunctions on trees," *RIMS Kôkyûroku*, vol. 1743, pp. 77–90, 2011.
  - Y. Nakatsukasa, N. Saito, & E. Woei: "Mysteries around graph Laplacian eigenvalue 4," *Linear Algebra & Its Applications*, vol. 438, no. 8, pp. 3231–3246, 2013.

Acknowledgment

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

Thank you very much for your attention!

Any Questions?