MAT 271: Applied & Computational Harmonic Analysis
Lecture 19: *Multiscale Basis Dictionaries on Graphs and Networks*

Naoki Saito

Department of Mathematics
University of California, Davis

June 7, 2023

- For much more details of this part of lecture, please check my course website on "Harmonic Analysis on Graphs & Networks": `http://www.math.ucdavis.edu/~saito/courses/HarmGraph/` as well as my articles with Jeff Irion at `http://www.math.ucdavis.edu/~saito/publications/`.
- We rely on the so-called graph Laplacians to construct our multiscale basis dictionaries. Some good references on graph Laplacian eigenvalues are:
  - R. B. Bapat: *Graphs and Matrices*, 2nd Ed., Springer, 2014.
  - A. E. Brouwer & W. H. Haemers: *Spectra of Graphs*, Springer, 2012.
  - F. R. K. Chung: *Spectral Graph Theory*, Amer. Math. Soc., 1997.
  - D. Cvetković, P. Rowlinson, & S. Simić: *An Introduction to the Theory of Graph Spectra*, Cambridge Univ. Press, 2010.
  - D. Spielman: "Spectral graph theory," in *Combinatorial Scientific Computing* (O. Schenk, ed.), Chap. 18, pp. 495–524, CRC Press, 2012.
- As for the graph Laplacian *eigenfunctions*, there are not too many books (although there may be many papers); one of the good books is
  - T. Bıyıkoğlu, J. Leydold, & P. F. Stadler, *Laplacian Eigenvectors of Graphs*, Lecture Notes in Mathematics, vol. 1915, Springer, 2007.

Motivations: Why Graphs?

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

Motivations: Why Graphs?

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

Motivations: Why Graphs?

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

Motivations: Why Graphs?

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

Motivations: Why Graphs?

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

Motivations: Why Graphs?

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

Motivations: Why Graphs?

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

Motivations: Why Graphs?

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

Motivations: Why Graphs?

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

Motivations: Why Graphs?

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

Motivations: Why Graphs?

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

An Example of Sensor Networks

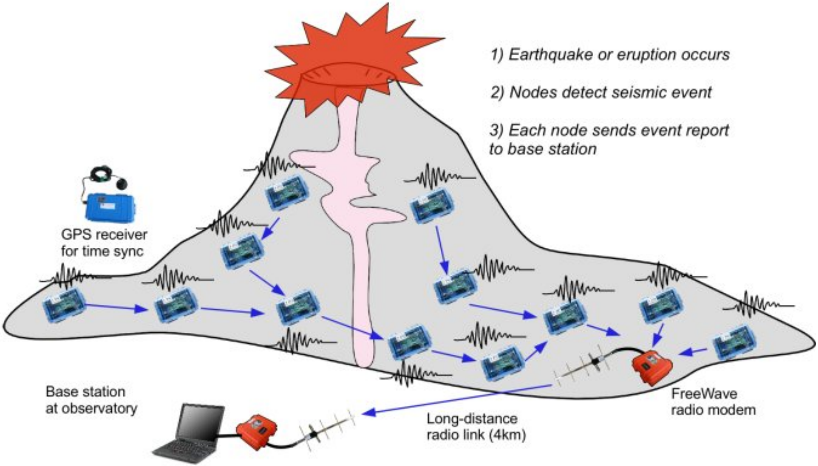


Figure: Volcano monitoring sensor network architecture of Harvard Sensor Networks Lab

An Example of Social Networks

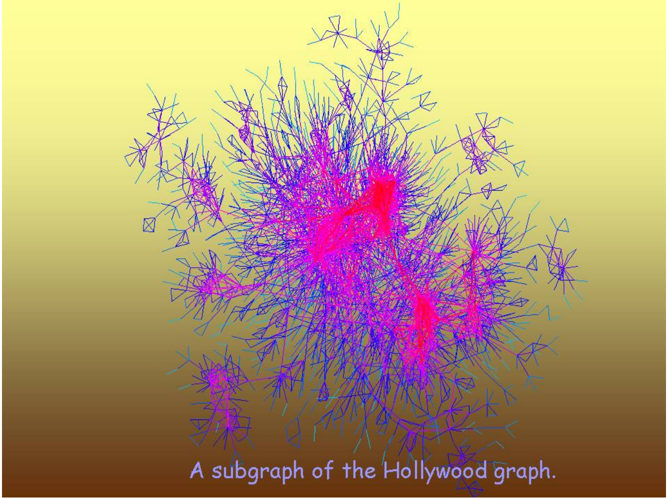


Figure: Through the courtesy of Prof. Fan Chung, UC San Diego

An Example of Biological Networks

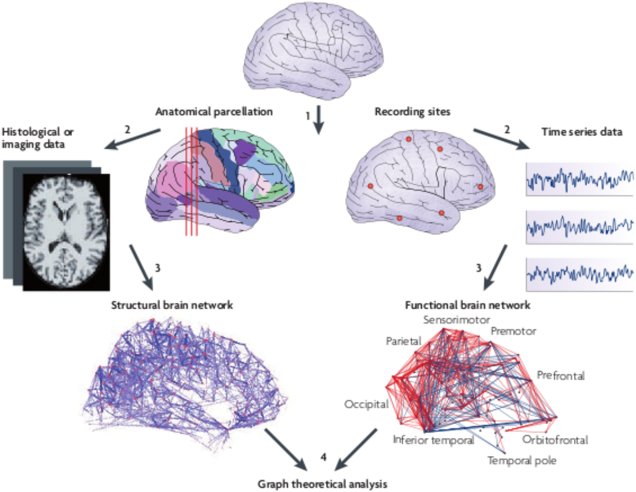
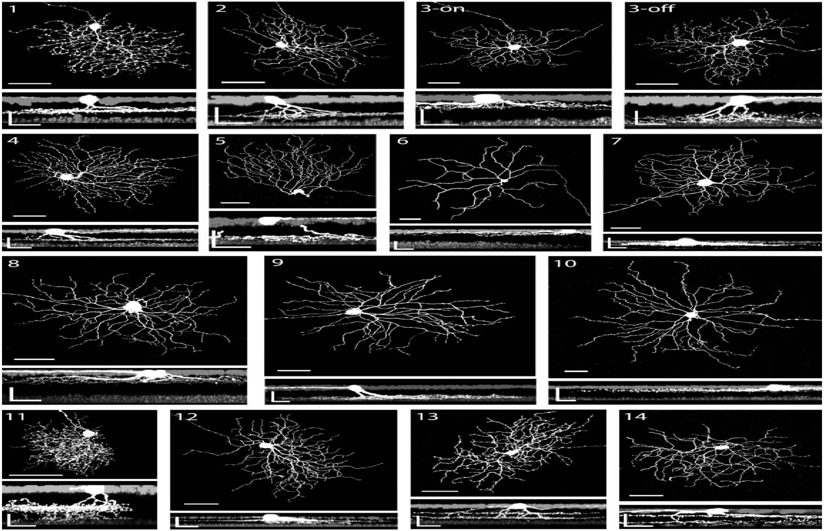
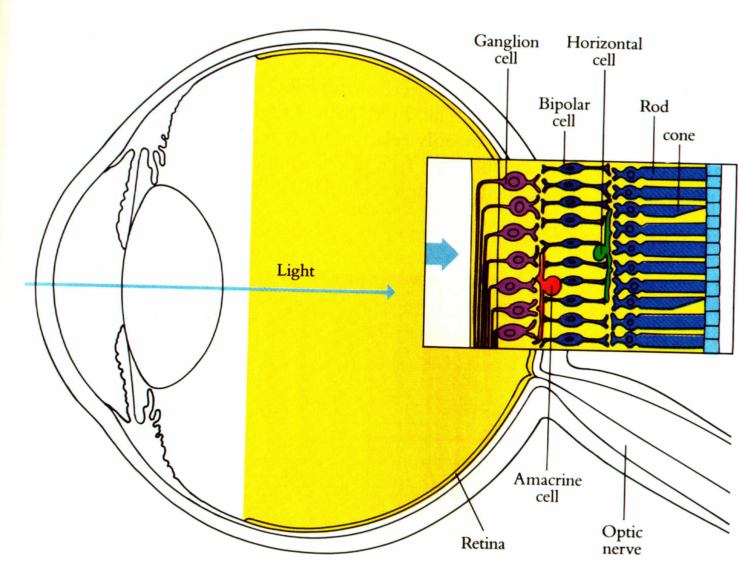


Figure: From E. Bullmore and O. Sporns, *Nature Reviews Neuroscience*, vol. 10, pp.186–198, Mar. 2009.

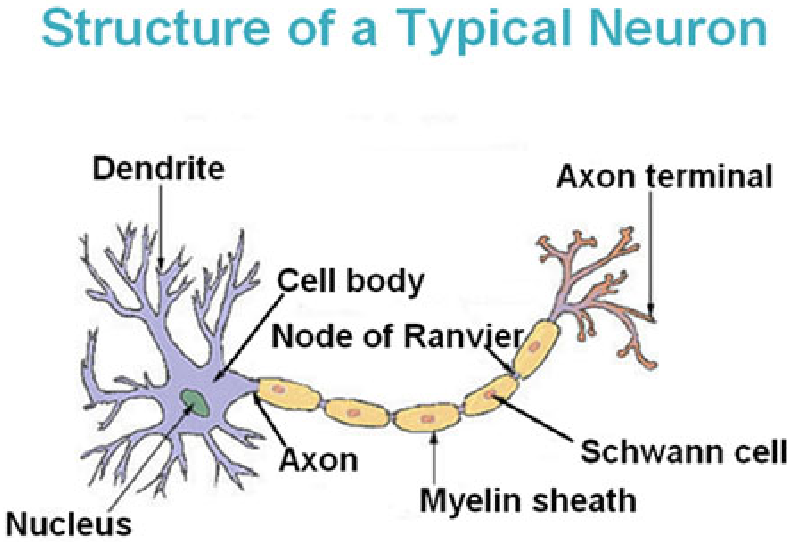
Another Biological Example: Retinal Ganglion Cells



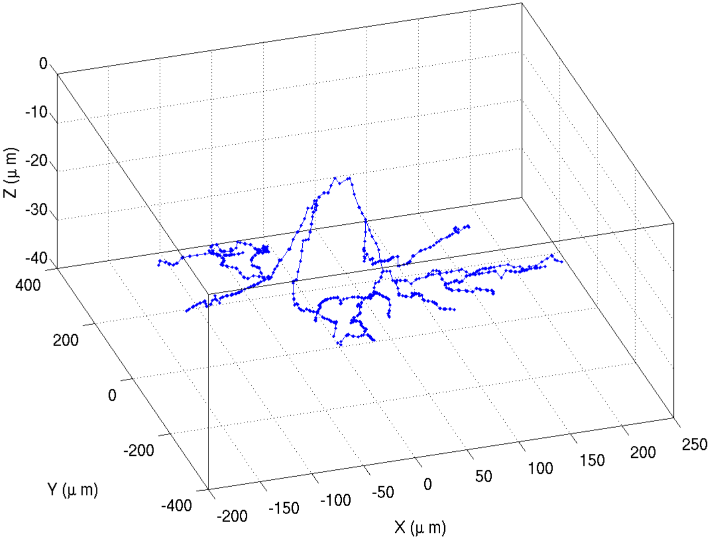
Retinal Ganglion Cells (D. Hubel: *Eye, Brain, & Vision*, '95)



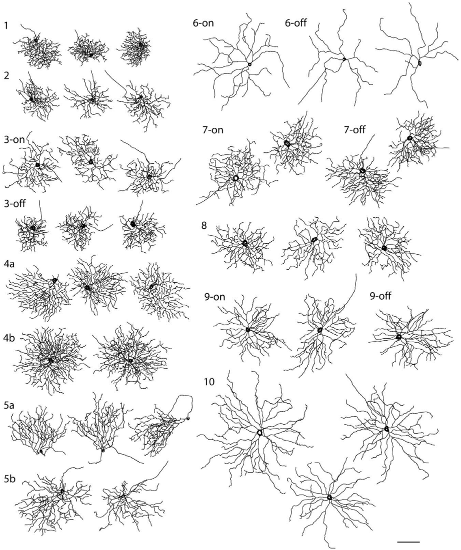
A Typical Neuron (from Wikipedia)



Mouse's RGC as a Graph



Clustering using Features Derived by Neurolucida®



Representing a Regular Image as a Graph

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

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

Representing a Regular Image as a Graph

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

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

Representing a Regular Image as a Graph

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

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

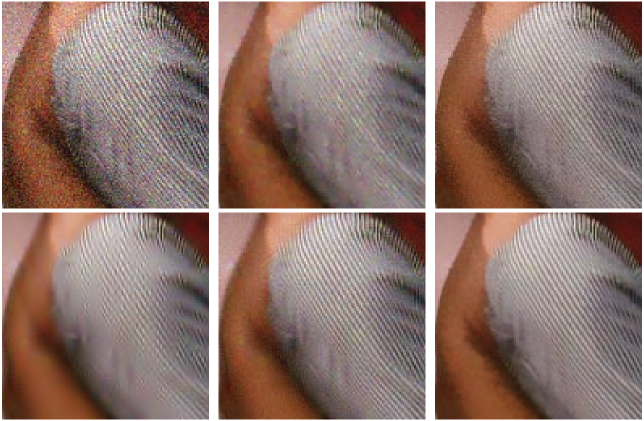
Representing a Regular Image as a Graph

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

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

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

Noisy Image; Total Variation Denoising; Neighborhood Filter



Trans. Inv. Wavelets; Empirical Wiener; Nonlocal Means

Motivations: Multiscale Basis Dictionary on Graphs

Wavelets

- Have been quite successful on regular domains
- Have been extended to irregular domains ⇒ "2nd Generation Wavelets"

For example:

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

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

Common strategy: Develop wavelet-*like* multiscale transforms

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

Motivations: Multiscale Basis Dictionary on Graphs

Wavelets

- Have been quite successful on regular domains
- Have been extended to irregular domains ⇒ "2nd Generation Wavelets"

For example:

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

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

Common strategy: Develop wavelet-*like* multiscale transforms

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

Motivations: Multiscale Basis Dictionary on Graphs

Wavelets

- Have been quite successful on regular domains
- Have been extended to irregular domains ⇒ "2nd Generation Wavelets"

For example:

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

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

Common strategy: Develop wavelet-*like* multiscale transforms

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

Motivations: Multiscale Basis Dictionary on Graphs

Wavelets

- Have been quite successful on regular domains
- Have been extended to irregular domains ⇒ "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

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

Common strategy: Develop wavelet-*like* multiscale transforms

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

Motivations: Multiscale Basis Dictionary on Graphs

Wavelets

- Have been quite successful on regular domains
- Have been extended to irregular domains ⇒ "2nd Generation Wavelets"

For example:

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

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

Common strategy: Develop wavelet-*like* multiscale transforms

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

Goals

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

Goals

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

Goals

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

Definitions and Notation

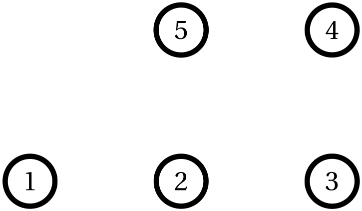
Let G be a graph.

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

Definitions and Notation

Let G be a graph.

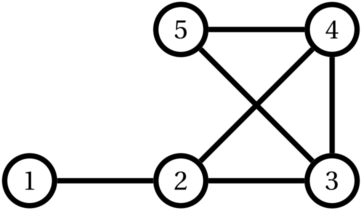
- $V = V(G) = \{v_1, \dots, v_N\}$ is the set of vertices.
- $E = E(G) = \{e_1, \dots, e_{N'}\}$ is the set of edges, where $e_k = (v_i, v_j)$ represents an edge (or line segment) connecting between adjacent vertices v_i, v_j for some $1 \le i, j \le N$.
- $W = W(G) \in \mathbb{R}^{N \times N}$ is the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



Definitions and Notation

Let G be a graph.

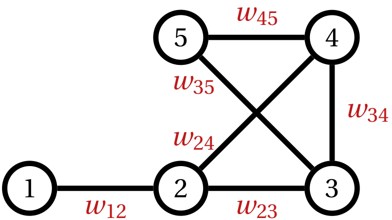
- $V = V(G) = \{v_1, \ldots, v_N\}$ is the set of vertices.
- $E = E(G) = \{e_1, \ldots, e_{N'}\}$ is the set of edges, where $e_k = (v_i, v_j)$ represents an edge (or line segment) connecting between adjacent vertices v_i, v_j for some $1 \le i, j \le N$.
- $W = W(G) \in \mathbb{R}^{N \times N}$ is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



Definitions and Notation

Let G be a graph.

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



Definitions and Notation

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

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

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

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

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

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

Definitions and Notation

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

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

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

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

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

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

Our Assumptions

In this lecture, we assume that the graph is

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

The graph may be weighted or unweighted.

Our Assumptions

In this lecture, we assume that the graph is

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

The graph may be weighted or unweighted.

Our Assumptions

In this lecture, we assume that the graph is

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

The graph may be weighted or unweighted.

Our Assumptions

In this lecture, we assume that the graph is

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

The graph may be weighted or unweighted.

Matrices Associated with a Graph

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

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

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

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

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

Matrices Associated with a Graph

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

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

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

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

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

Matrices Associated with a Graph

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

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

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

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

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

Graph Laplacians . . .

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

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

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

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

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

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

Why Graph Laplacian Eigenfunctions?

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

Why Graph Laplacian Eigenfunctions?

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

Why Graph Laplacian Eigenfunctions?

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

Why Graph Laplacian Eigenfunctions?

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

Why Graph Laplacian Eigenfunctions?

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

Why Graph Laplacian Eigenfunctions?

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

Why Graph Laplacian Eigenfunctions?

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

A Simple Yet Important Example: A Path Graph

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

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

- $\lambda_k = 2 - 2\cos(\pi k/N) = 4\sin^2(\pi k/2N),\ k = 0 : N-1.$
- $\phi_k(\ell) = a_{k;N} \cos\left(\pi k(\ell + \frac{1}{2})/N\right),\ k, \ell = 0 : N-1;$ $a_{k;N}$ is a const. s.t. $\|\phi_k\|_2 = 1$.
- In this simple case, λ (eigenvalue) is a monotonic function w.r.t. the frequency, which is the eigenvalue index k. *For a general graph, however, the notion of frequency is not well defined.*

A Simple Yet Important Example: A Path Graph

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

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

- $\lambda_k = 2 - 2\cos(\pi k/N) = 4\sin^2(\pi k/2N), \; k = 0 : N-1.$
- $\boldsymbol{\phi}_k(\ell) = a_{k;N} \cos\left(\pi k(\ell + \frac{1}{2})/N\right), \; k, \ell = 0 : N-1;$ $a_{k;N}$ is a const. s.t. $\|\boldsymbol{\phi}_k\|_2 = 1$.
- In this simple case, λ (eigenvalue) is a monotonic function w.r.t. the frequency, which is the eigenvalue index k. *For a general graph, however, the notion of frequency is not well defined.*

A Brief Review of Graph Laplacian Eigenpairs

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

A Brief Review of Graph Laplacian Eigenpairs

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

A Brief Review of Graph Laplacian Eigenpairs

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

A Brief Review of Graph Laplacian Eigenpairs

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

A Brief Review of Graph Laplacian Eigenpairs

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

A Brief Review of Graph Laplacian Eigenpairs

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

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

Plan: use the signs of the entries in $\boldsymbol{\phi}_1$, which is known as the Fiedler vector

Why? Using $\boldsymbol{\phi}_1$ to generate X and X^c yields an approximate minimizer of the RatioCut function[1,2]:

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

where

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

- 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

[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] We could also use the signs of $\boldsymbol{\phi}_1$ for L_{rw} (equivalently, L_{sym}), which yield an approximate minimizer of the popular Normalized Cut function: J. Shi & J. Malik: "Normalized cuts and image segmentation", *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, no. 8, pp. 888–905, 2000.

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

Plan: use the signs of the entries in $\boldsymbol{\phi}_1$, which is known as the Fiedler vector

Why? Using $\boldsymbol{\phi}_1$ to generate X and X^c yields an approximate minimizer of the RatioCut function[1,2]:

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

where

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

- 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

[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] We could also use the signs of $\boldsymbol{\phi}_1$ for L_{rw} (equivalently, L_{sym}), which yield an approximate minimizer of the popular Normalized Cut function: J. Shi & J. Malik: "Normalized cuts and image segmentation", *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, no. 8, pp. 888–905, 2000.

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

Plan: use the signs of the entries in $\boldsymbol{\phi}_1$, which is known as the Fiedler vector

Why? Using $\boldsymbol{\phi}_1$ to generate X and X^c yields an approximate minimizer of the RatioCut function[1,2]:

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

where

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

- 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

[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] We could also use the signs of $\boldsymbol{\phi}_1$ for L_{rw} (equivalently, L_{sym}), which yield an approximate minimizer of the popular Normalized Cut function: J. Shi & J. Malik: "Normalized cuts and image segmentation", *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, no. 8, pp. 888–905, 2000.

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

Plan: use the signs of the entries in $\boldsymbol{\phi}_1$, which is known as the Fiedler vector

Why? Using $\boldsymbol{\phi}_1$ to generate X and X^c yields an approximate minimizer of the RatioCut function[1,2]:

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

where

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

- 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

[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] We could also use the signs of $\boldsymbol{\phi}_1$ for L_{rw} (equivalently, L_{sym}), which yield an approximate minimizer of the popular Normalized Cut function: J. Shi & J. Malik: "Normalized cuts and image segmentation", *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, no. 8, pp. 888–905, 2000.

Graph Partitioning via Spectral Clustering

Let us reformulate the RatioCut minimization problem.

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

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

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

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

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

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

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

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

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

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

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

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]:

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

- 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]:

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

- 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]:

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

- 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]:

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

- 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

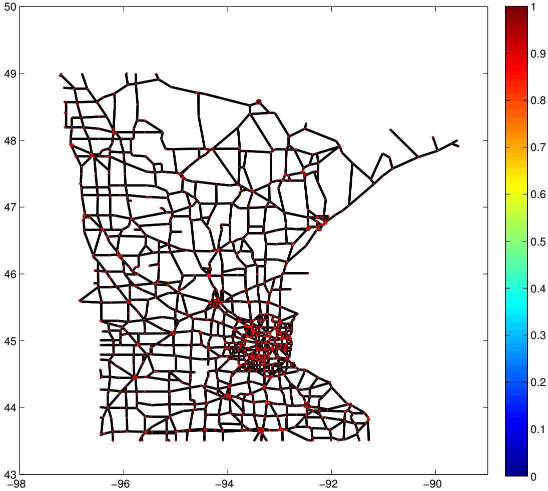


Figure: The MN road network

Example of Graph Partitioning

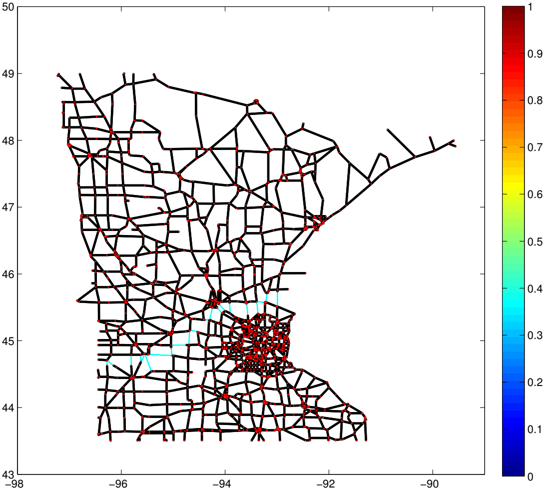
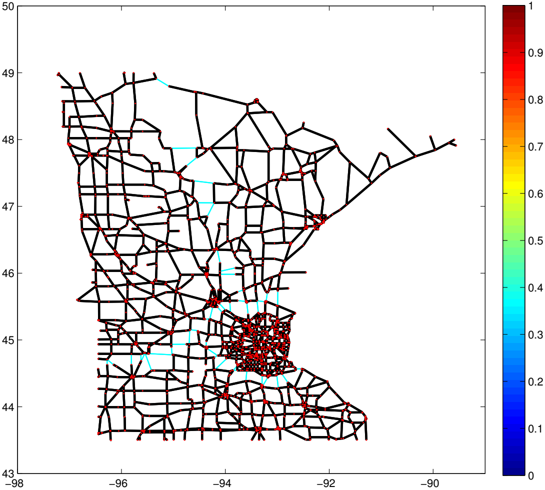


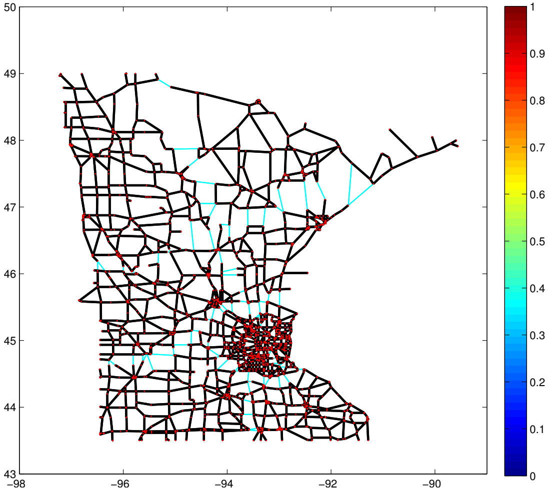
Figure: The MN road network partitioned via the Fiedler vector of L_{rw}

One Can Do This Recursively!



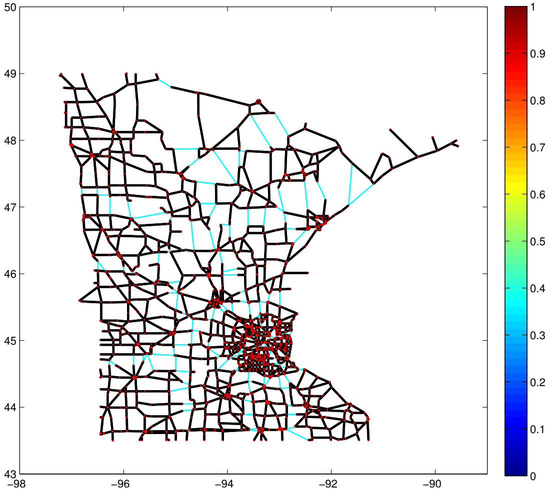
The MN road network recursively partitioned via the Fiedler vectors of L_{rw}'s of subgraphs: $j = 2$

One Can Do This Recursively!



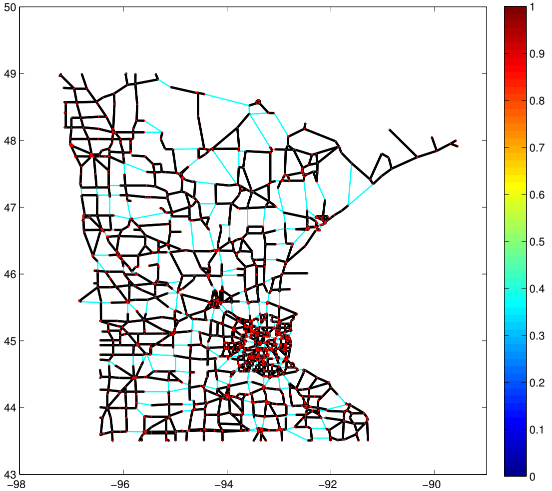
$j = 3$

One Can Do This Recursively!



$j = 4$

One Can Do This Recursively!



$j = 5$

Our transforms involve 2 main steps:

1. Recursively partition the graph

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

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

Our transforms involve 2 main steps:

1. Recursively partition the graph

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

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

Hierarchical Graph Laplacian Eigen Transform (HGLET)

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

The algorithm proceeds as follows...

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

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

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

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

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

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

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

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

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

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

1. Generate an orthonormal basis for the entire graph ⇒ 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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\vdots$$

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

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

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

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

$$\vdots$$

Remarks

- For an unweighted path graph, this 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

Remarks

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

Remarks

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

Remarks

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

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

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

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

Remarks

- For an unweighted path graph, this 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

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

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

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

Remarks

- For an unweighted path graph, this 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

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

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

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

Remarks

- For an unweighted path graph, this 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

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

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

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

Remarks

- For an unweighted path graph, this 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

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

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

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

Remarks

- For an unweighted path graph, this 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

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

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

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

Related Work

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

1. A. D. Szlam, M. Maggioni, R. R. Coifman, and J. C. Bremer, Jr., "Diffusion-driven multiscale analysis on manifolds and graphs: top-down and bottom-up constructions," in *Wavelets XI* (M. Papadakis et al. eds.), *Proc. SPIE 5914*, Paper # 59141D, 2005.

Related Work

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

1. A. D. Szlam, M. Maggioni, R. R. Coifman, and J. C. Bremer, Jr., "Diffusion-driven multiscale analysis on manifolds and graphs: top-down and bottom-up constructions," in *Wavelets XI* (M. Papadakis et al. eds.), *Proc. SPIE 5914*, Paper # 59141D, 2005.

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

Computational Complexity: HGLET

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

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

Generalized Haar-Walsh Transform (GHWT)

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

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

The algorithm proceeds as follows...

Generalized Haar-Walsh Transform (GHWT)

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

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

The algorithm proceeds as follows...

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

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

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

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

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

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

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

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

$$\vdots$$

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

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

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

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

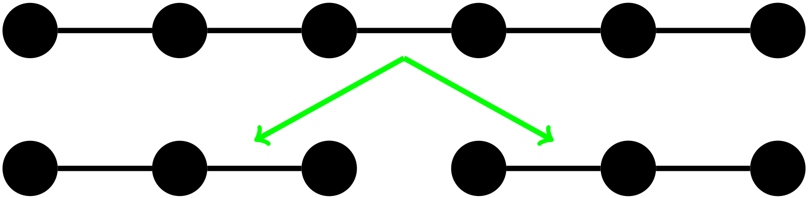
$$\vdots$$

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

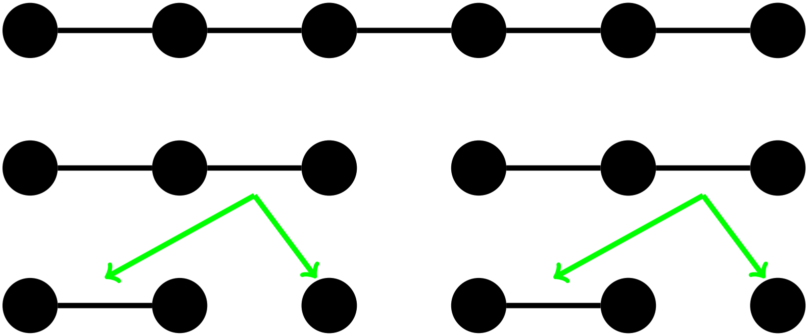
$$\left[\ \boldsymbol{\psi}^{j_{\max}}_{0,0}\ \right]\ \left[\ \boldsymbol{\psi}^{j_{\max}}_{1,0}\ \right]\ \left[\ \boldsymbol{\psi}^{j_{\max}}_{2,0}\ \right]\ \left[\ \boldsymbol{\psi}^{j_{\max}}_{3,0}\ \right]\ \cdots\ \left[\ \boldsymbol{\psi}^{j_{\max}}_{K^{j_{\max}}-2,0}\ \right]\ \left[\ \boldsymbol{\psi}^{j_{\max}}_{K^{j_{\max}}-1,0}\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

GHWT on P_6

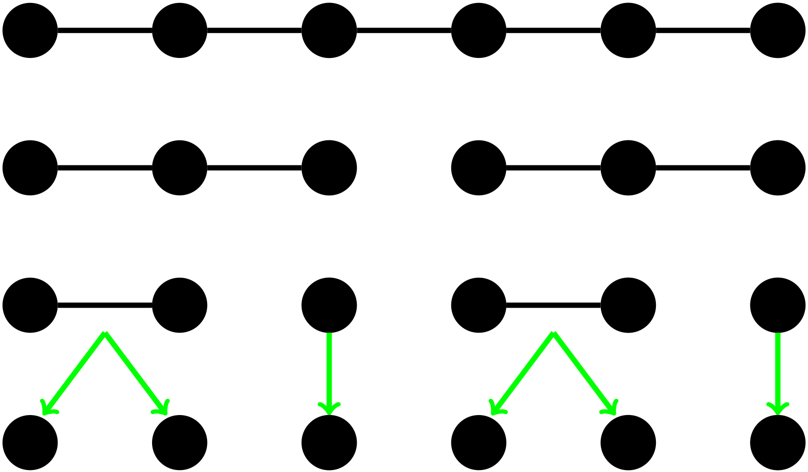
GHWT on P_6



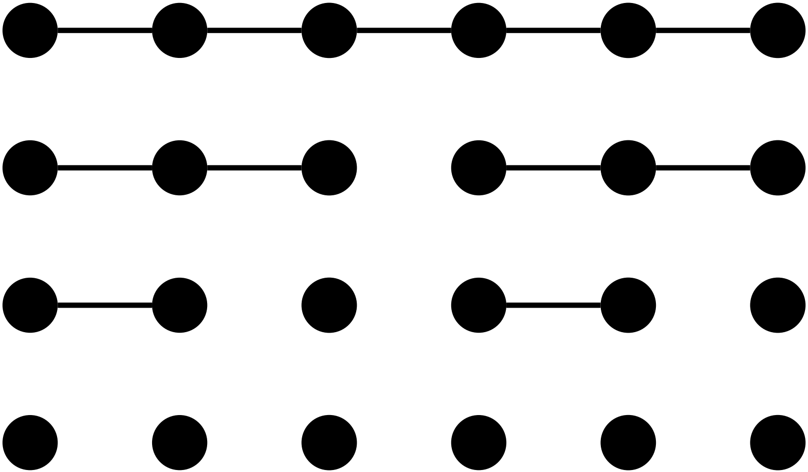
GHWT on P_6



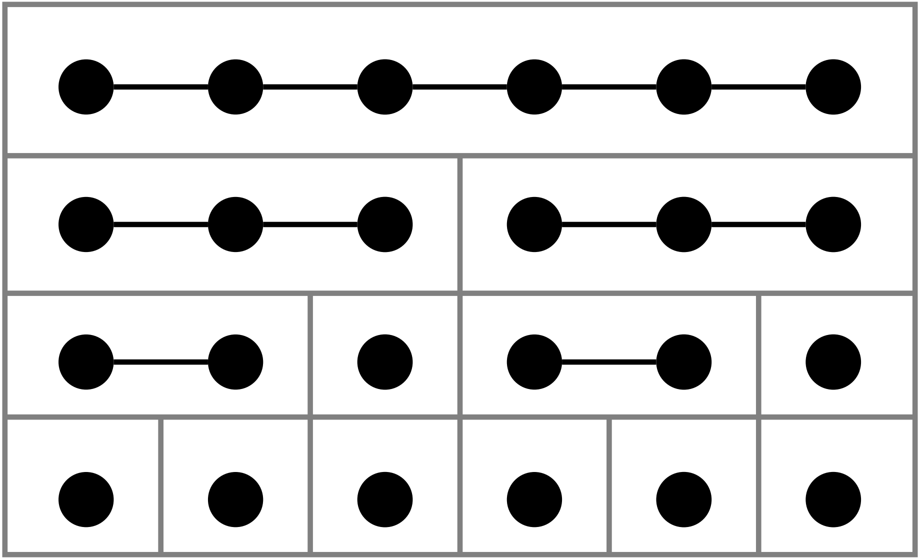
GHWT on P_6



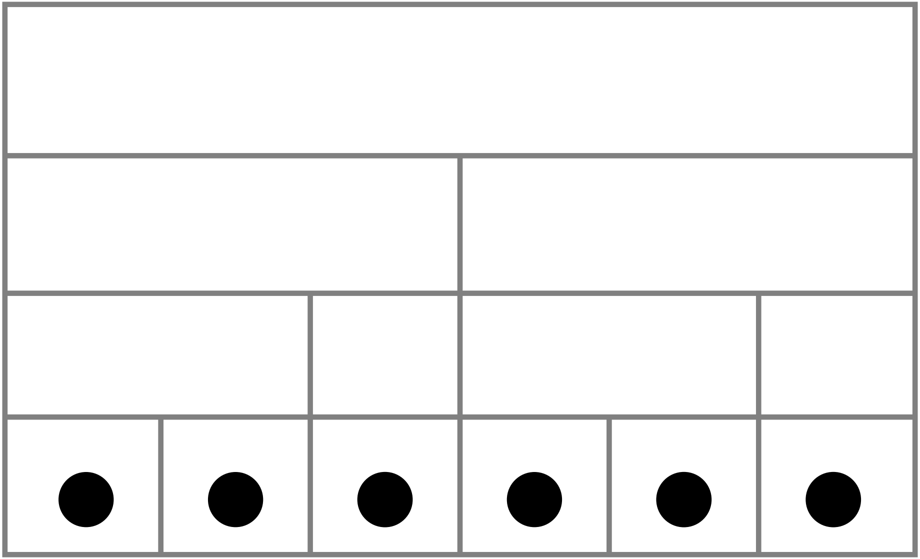
GHWT on P_6



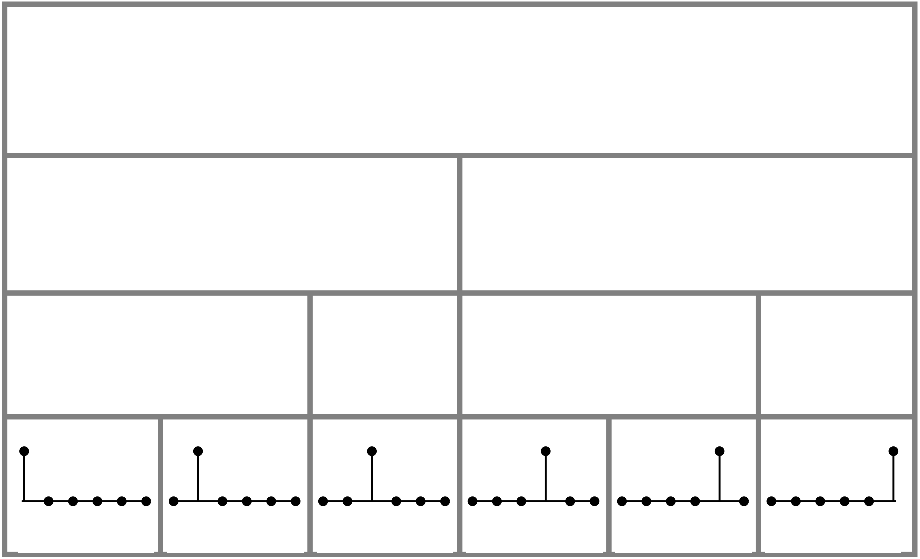
GHWT on P_6



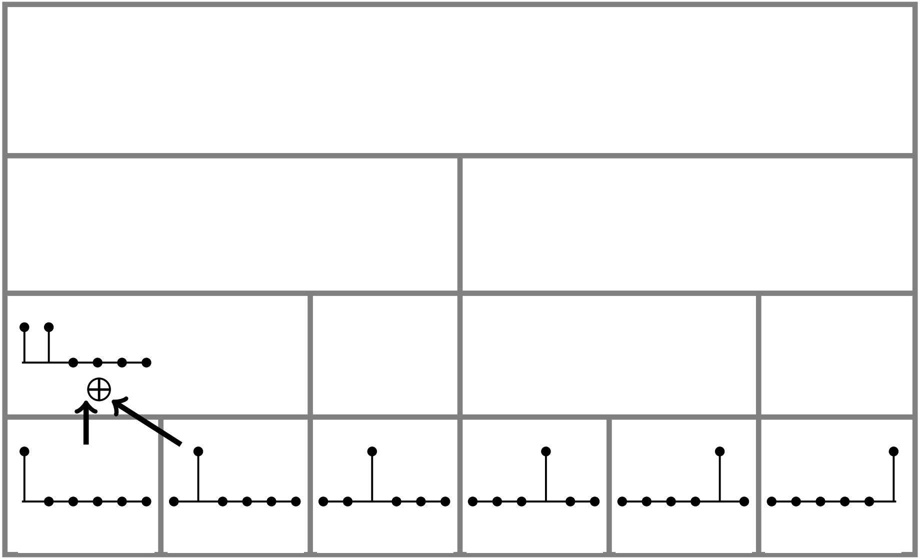
GHWT on P_6



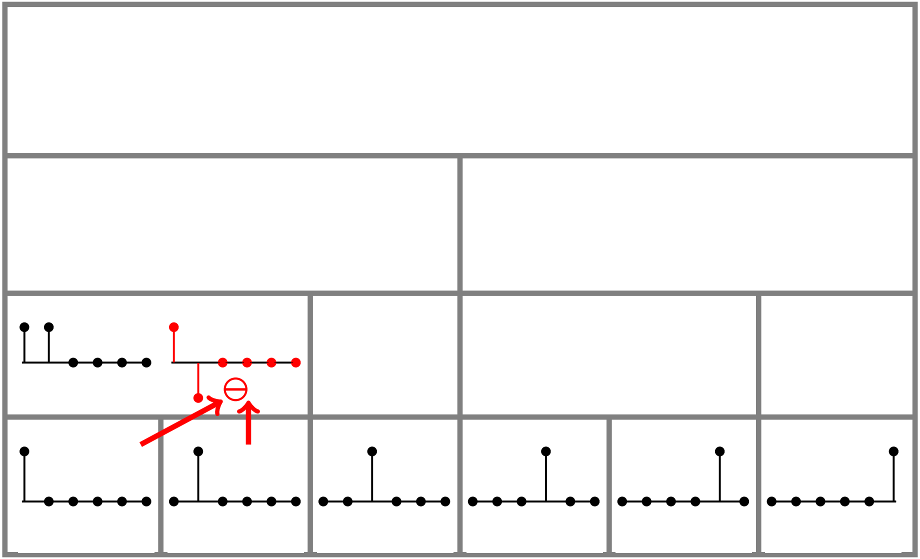
GHWT on P_6



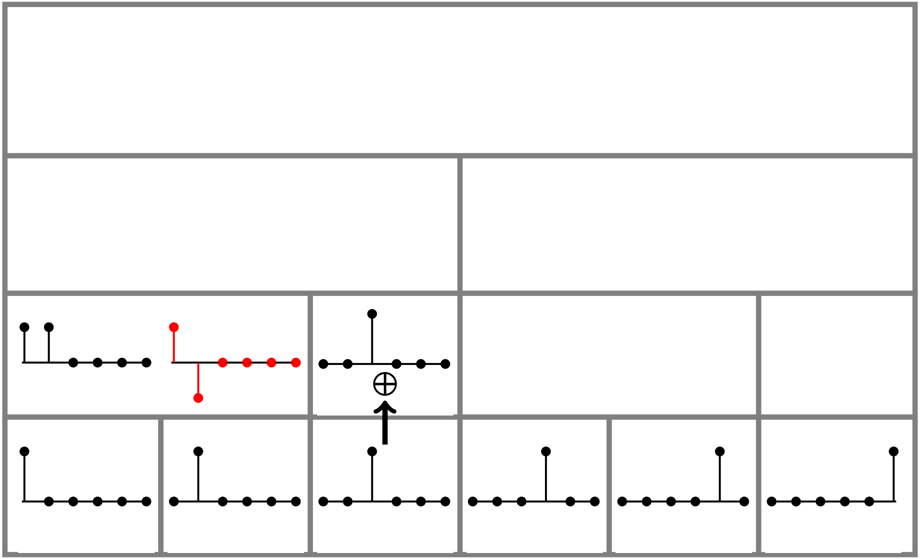
GHWT on P_6



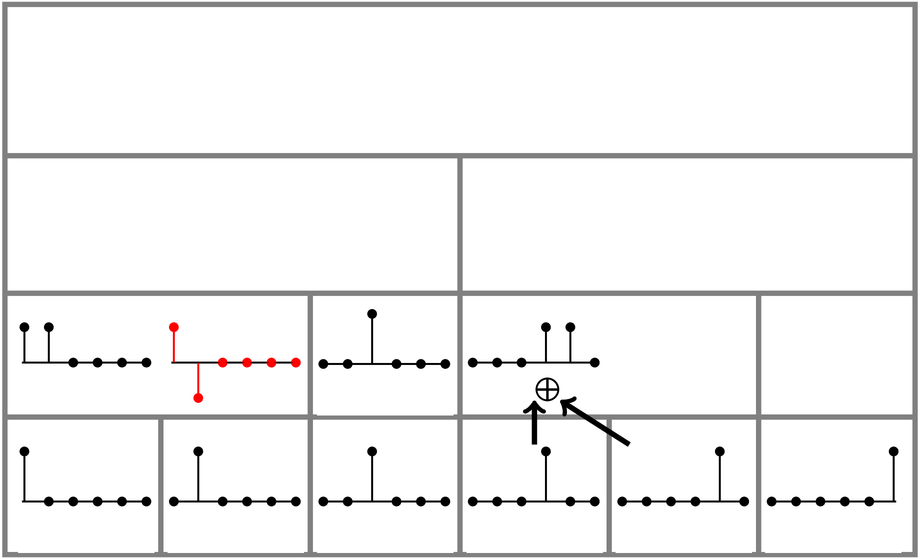
GHWT on P_6



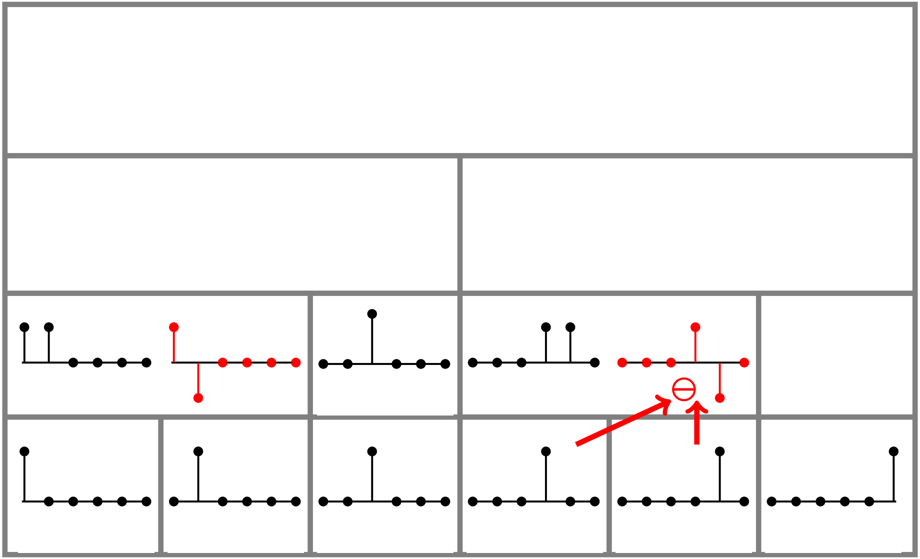
GHWT on P_6



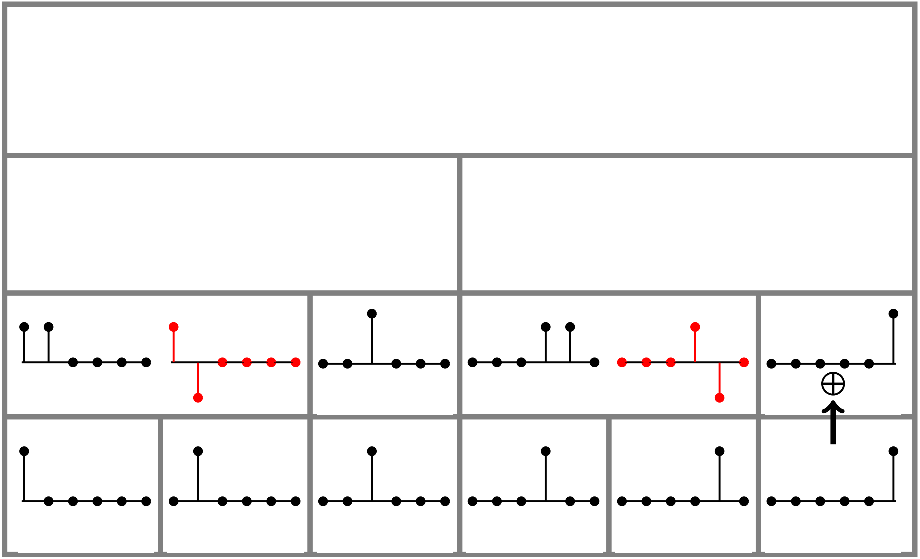
GHWT on P_6



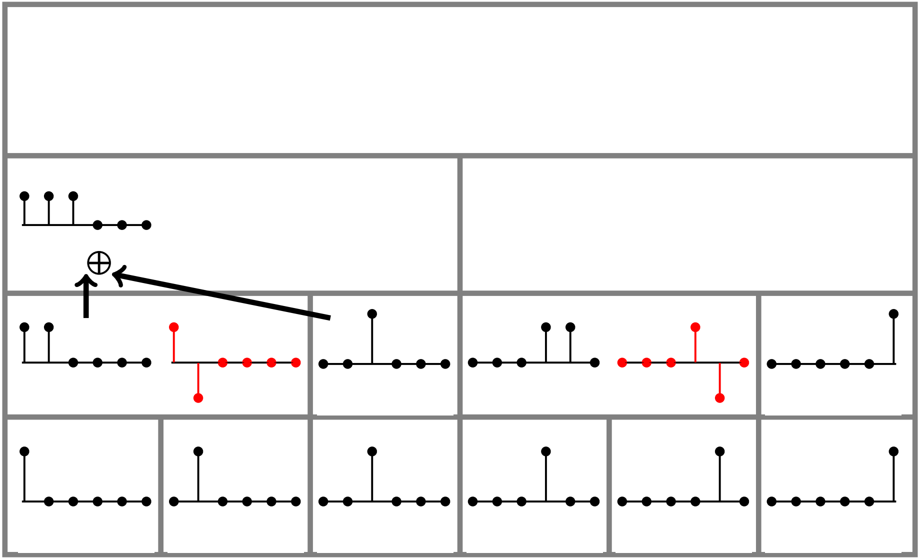
GHWT on P_6



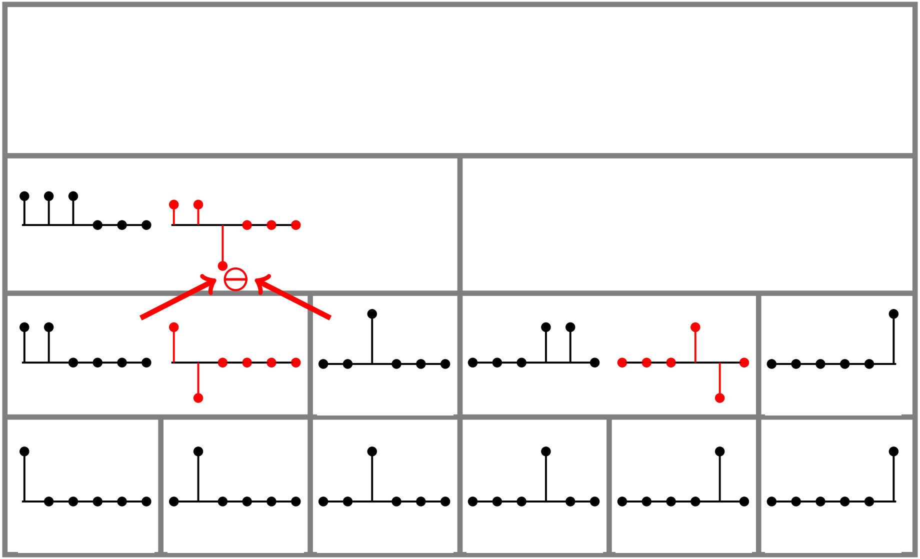
GHWT on P_6



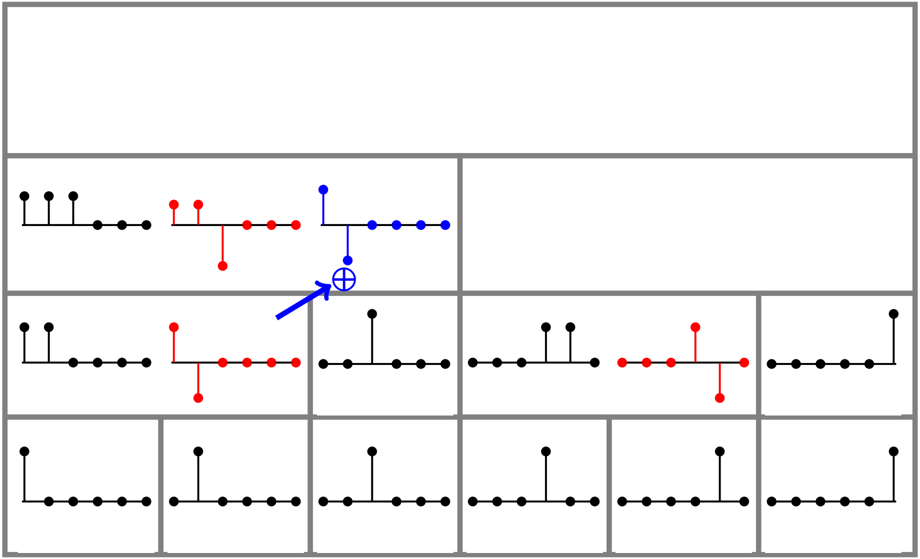
GHWT on P_6



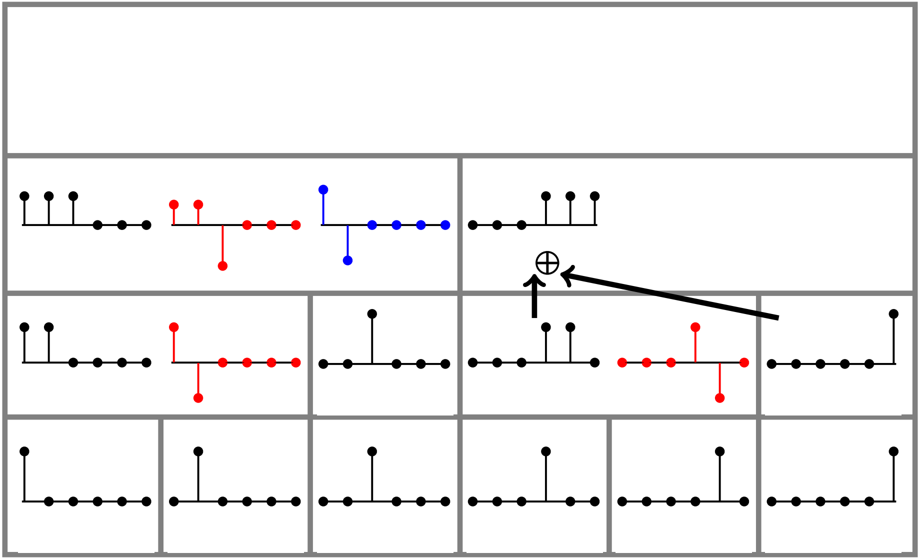
GHWT on P_6



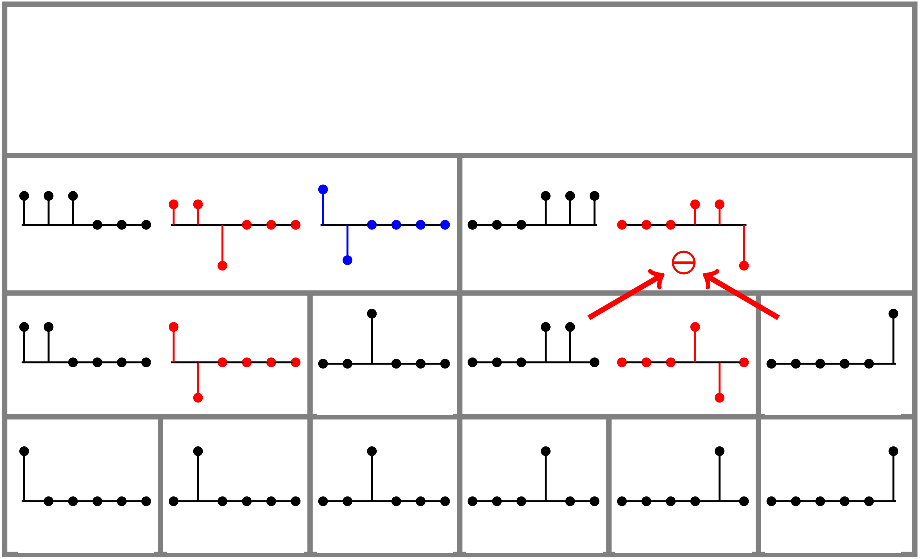
GHWT on P_6



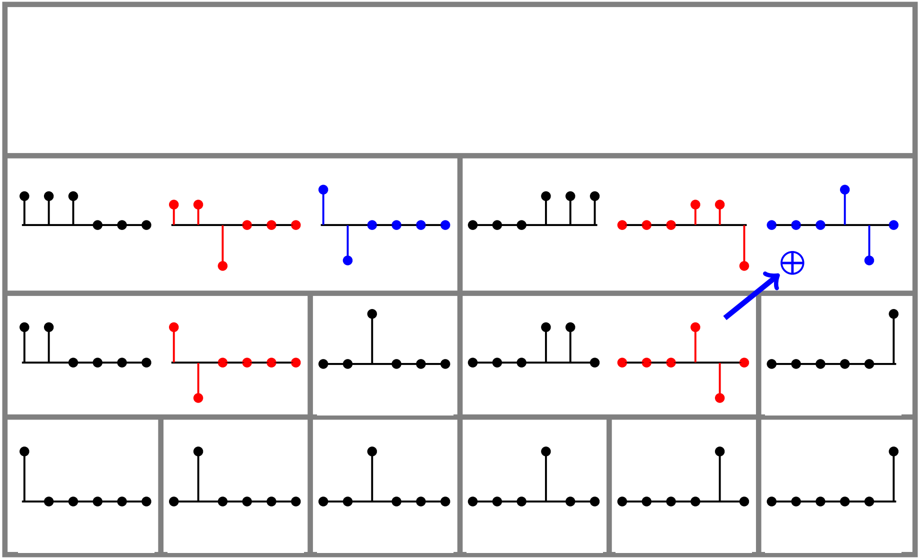
GHWT on P_6



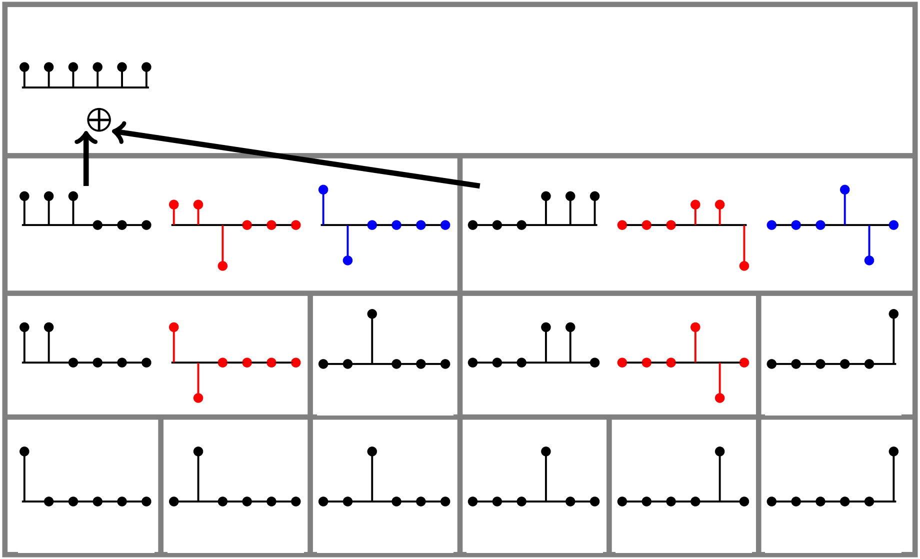
GHWT on P_6



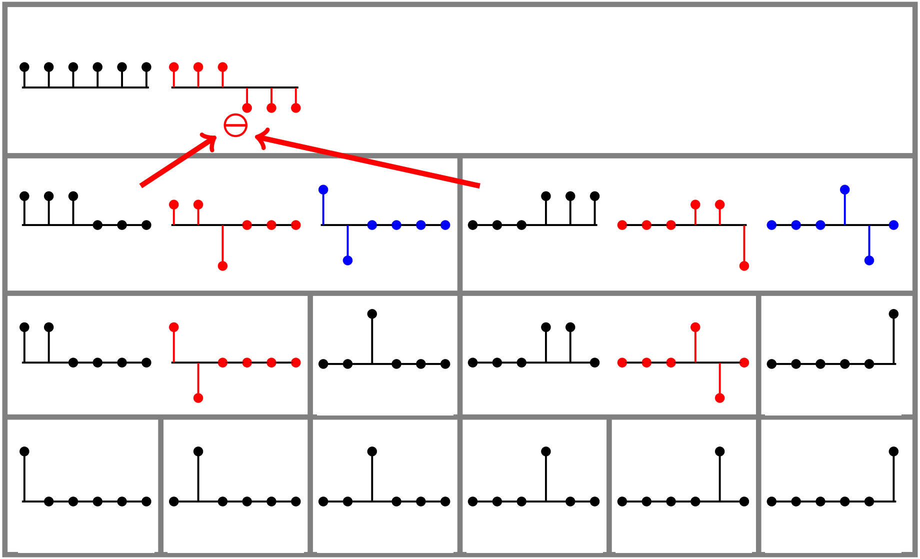
GHWT on P_6



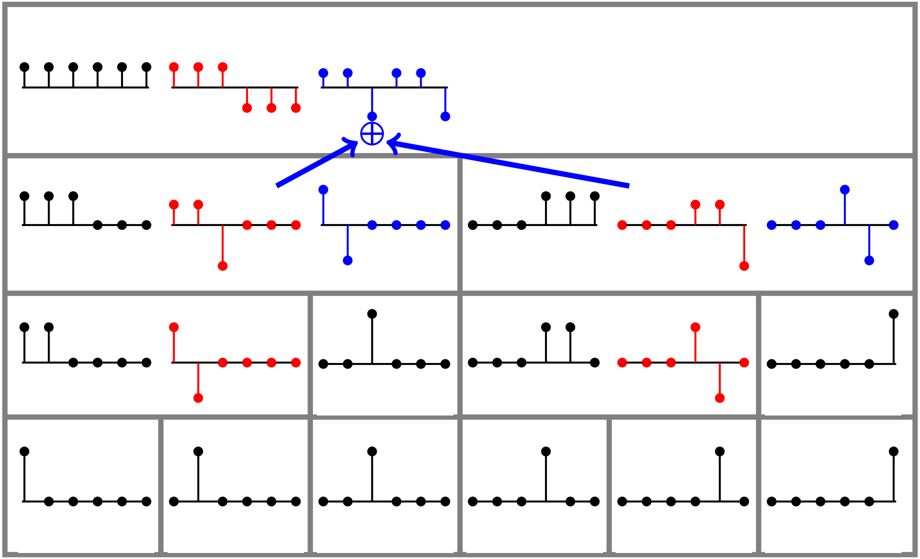
GHWT on P_6



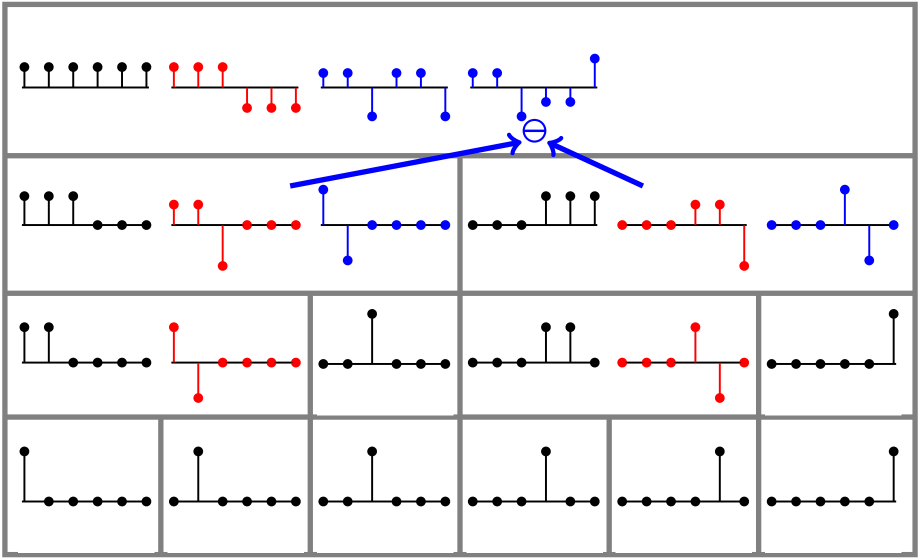
GHWT on P_6



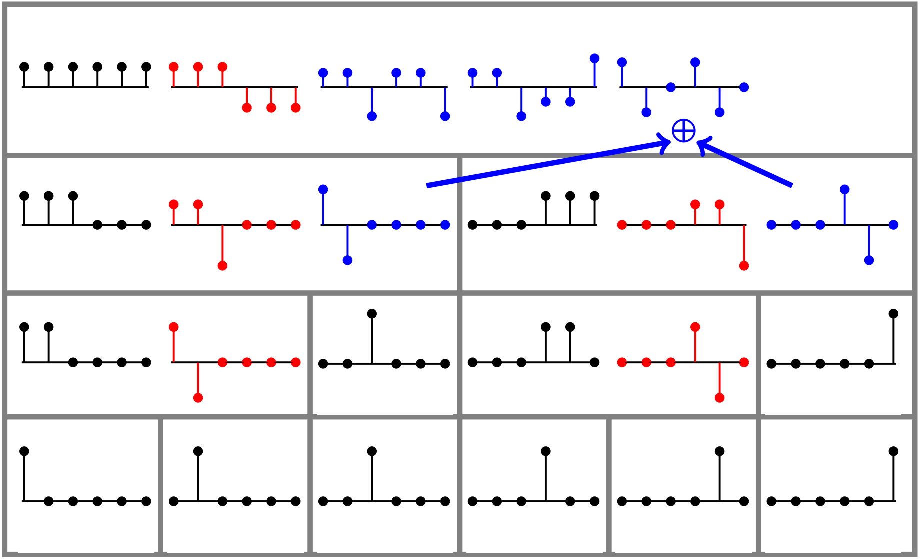
GHWT on P_6



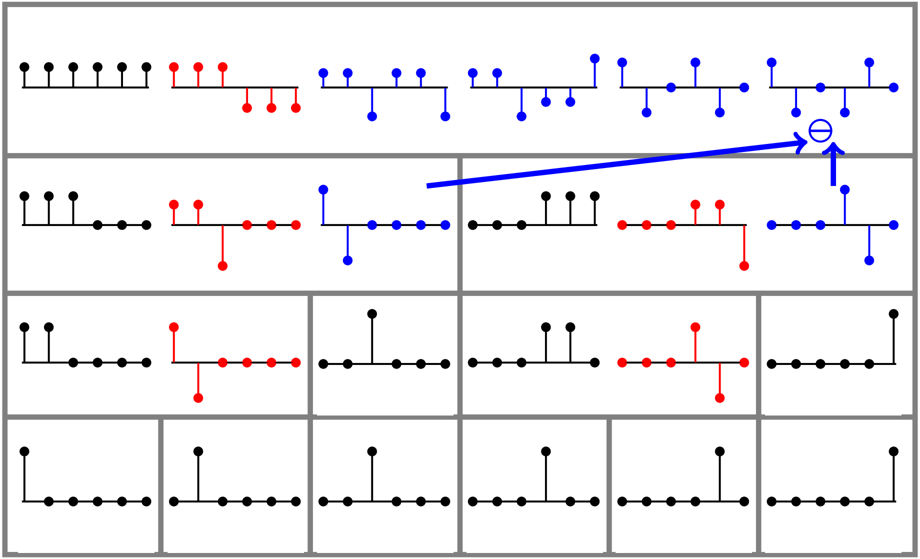
GHWT on P_6



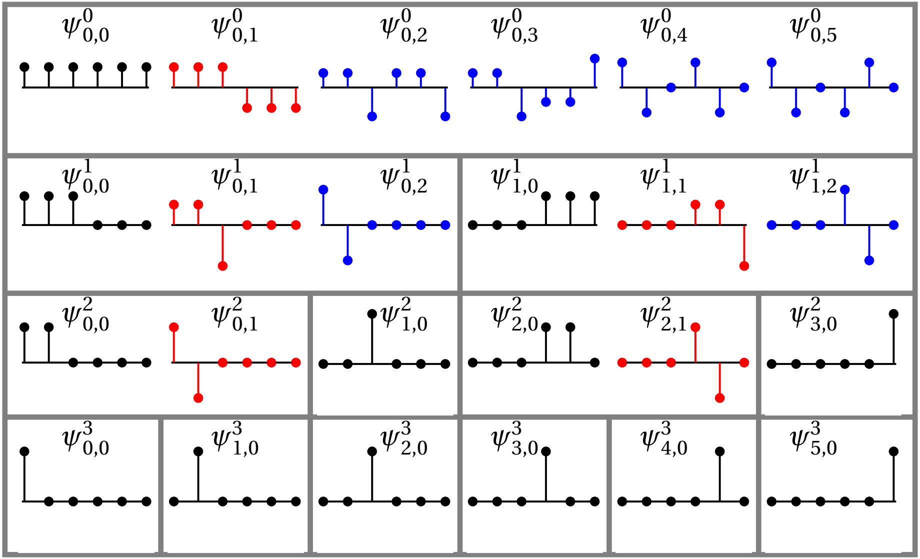
GHWT on P_6



GHWT on P_6



GHWT on P_6



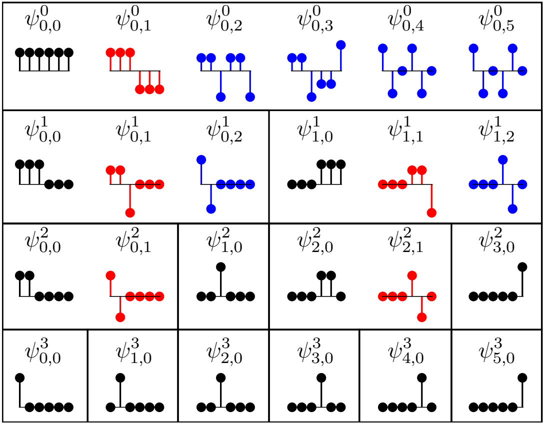
$\psi^0_{0,0}$ $\psi^0_{0,1}$ $\psi^0_{0,2}$ $\psi^0_{0,3}$ $\psi^0_{0,4}$ $\psi^0_{0,5}$
$\psi^1_{0,0}$ $\psi^1_{0,1}$ $\psi^1_{0,2}$ $\psi^1_{1,0}$ $\psi^1_{1,1}$ $\psi^1_{1,2}$
$\psi^2_{0,0}$ $\psi^2_{0,1}$ $\psi^2_{1,0}$ $\psi^2_{2,0}$ $\psi^2_{2,1}$ $\psi^2_{3,0}$
$\psi^3_{0,0}$ $\psi^3_{1,0}$ $\psi^3_{2,0}$ $\psi^3_{3,0}$ $\psi^3_{4,0}$ $\psi^3_{5,0}$

Remarks

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

Remarks

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



Remarks

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

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

Remarks

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

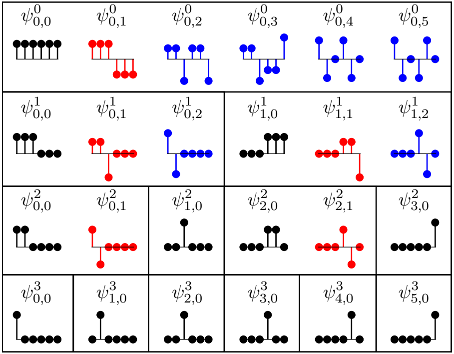


Figure: Default dictionary; i.e., coarse-to-fine

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

Remarks

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

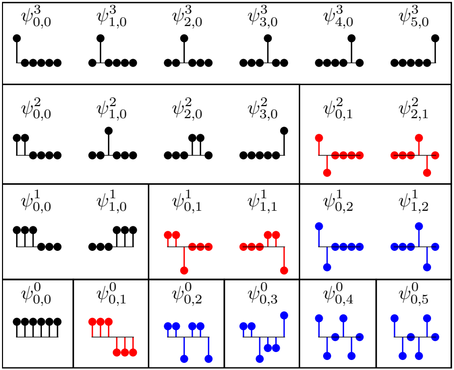


Figure: Reordered & regrouped dictionary; i.e., fine-to-coarse

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

Remarks

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

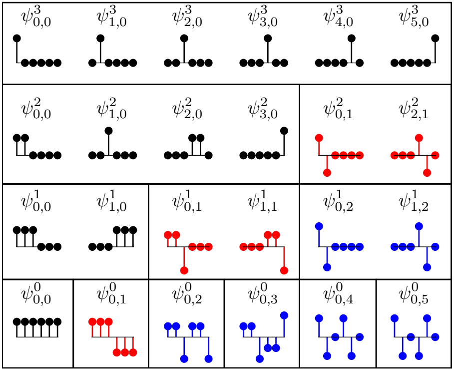


Figure: Reordered & regrouped dictionary; i.e., fine-to-coarse

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

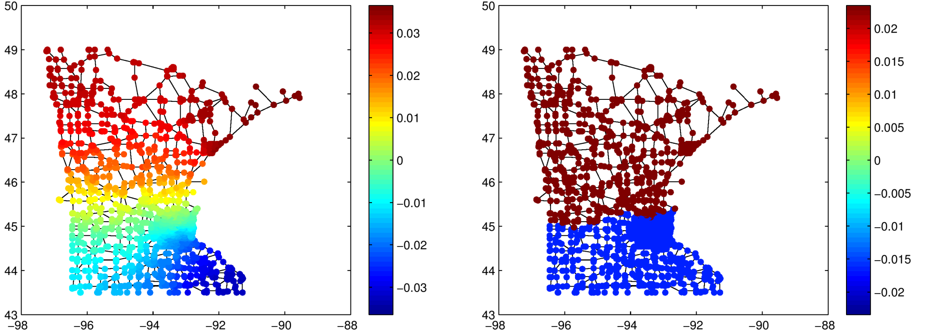
HGLET vs. GHWT

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

HGLET vs. GHWT

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

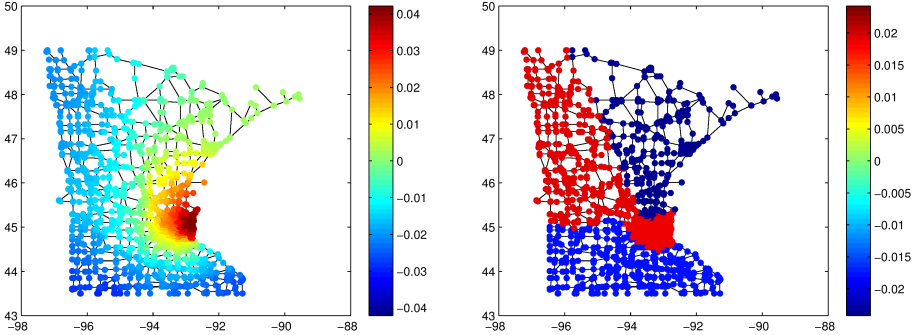
Level $j = 0$, Region $k = 0$, $l = 1$



HGLET vs. GHWT

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

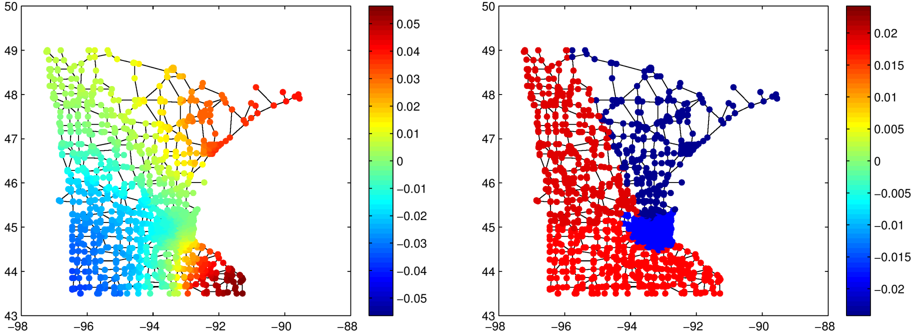
Level $j = 0$, Region $k = 0$, $l = 2$



HGLET vs. GHWT

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

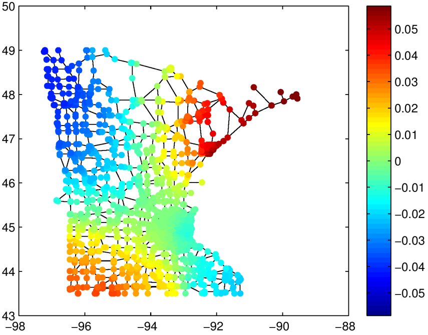
Level $j = 0$, Region $k = 0$, $l = 3$

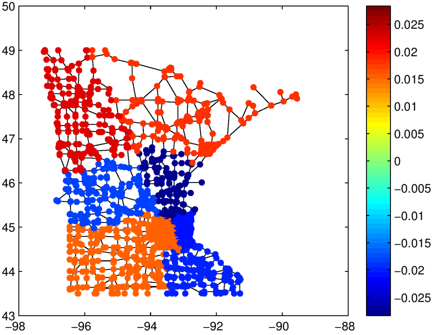


HGLET vs. GHWT

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

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

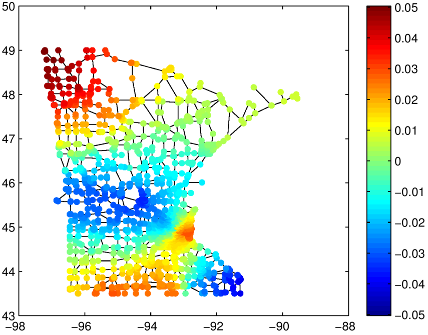


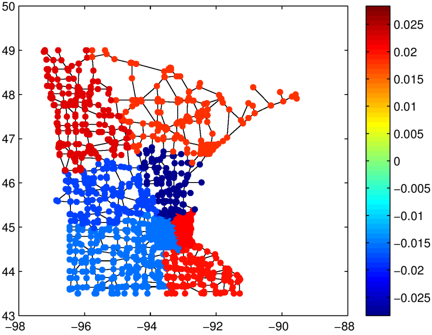


HGLET vs. GHWT

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

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

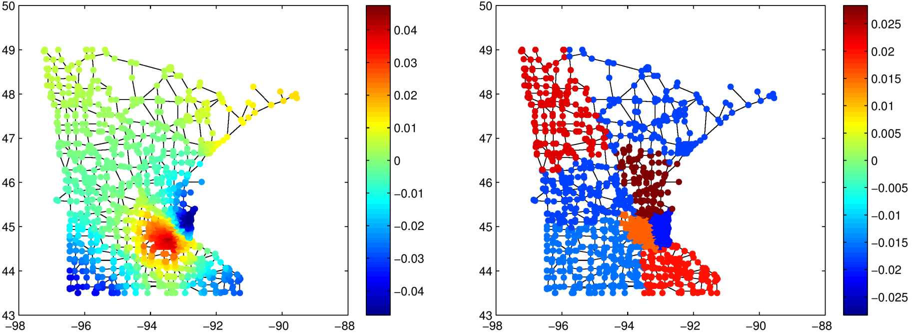




HGLET vs. GHWT

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

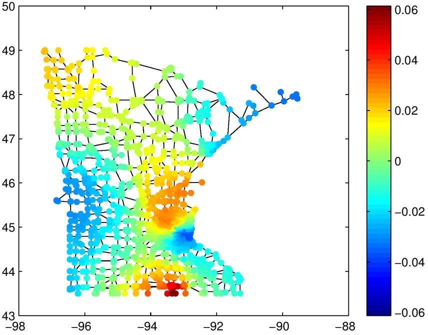
Level $j = 0$, Region $k = 0$, $l = 6$

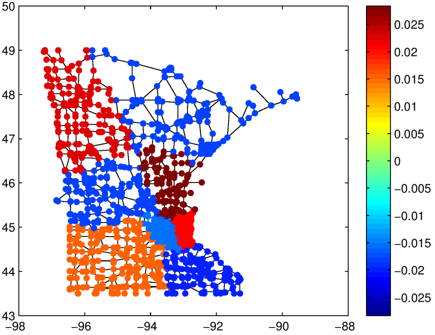


HGLET vs. GHWT

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

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

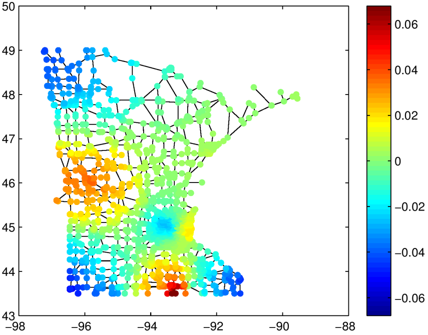


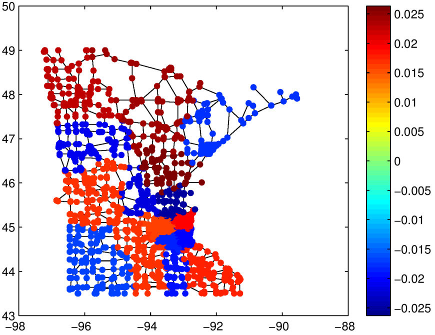


HGLET vs. GHWT

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

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

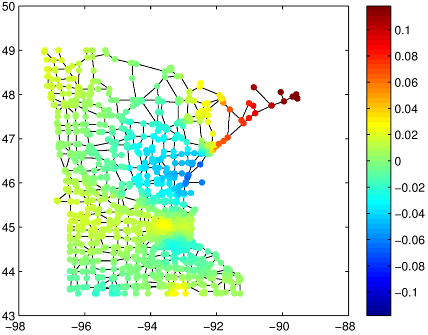


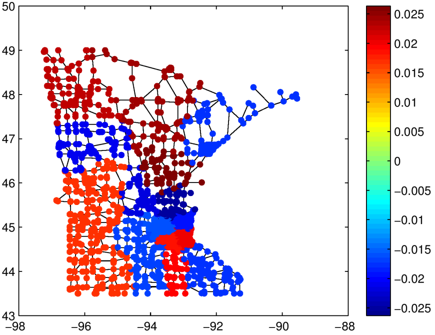


HGLET vs. GHWT

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

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

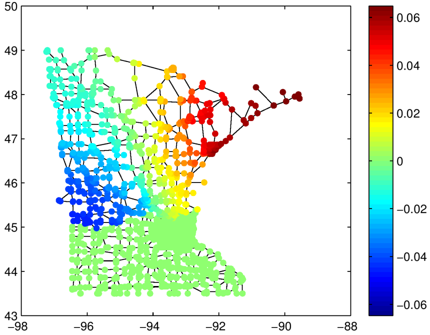


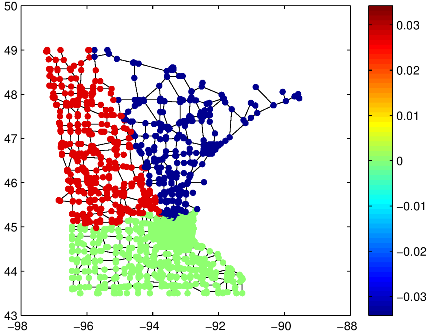


HGLET vs. GHWT

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

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

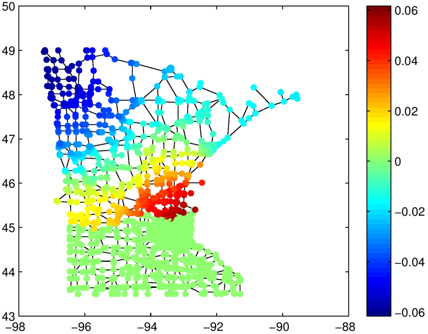


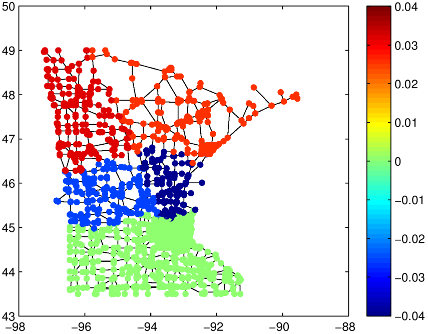


HGLET vs. GHWT

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

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

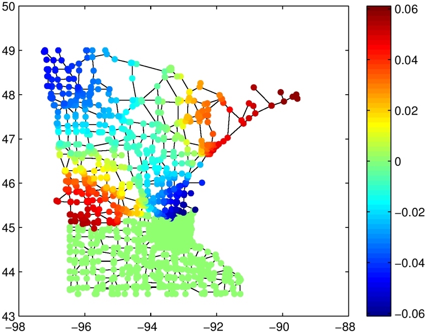


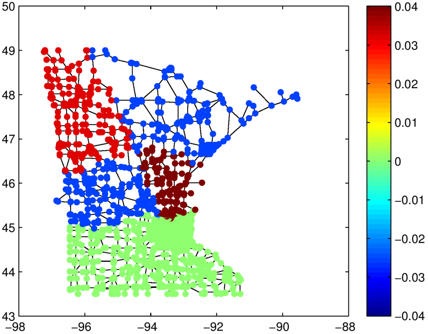


HGLET vs. GHWT

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

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

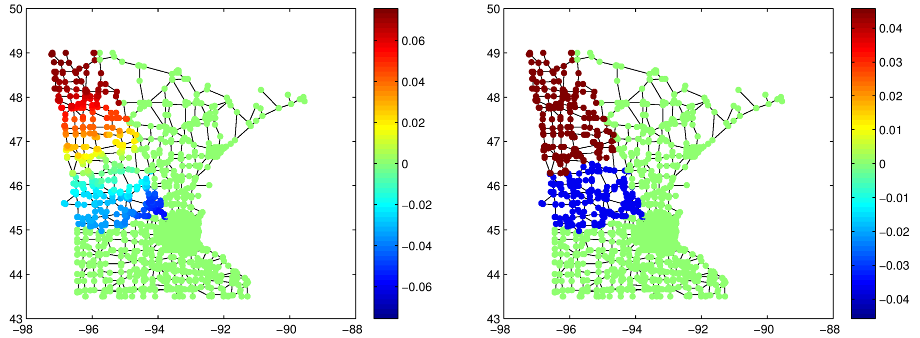




HGLET vs. GHWT

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

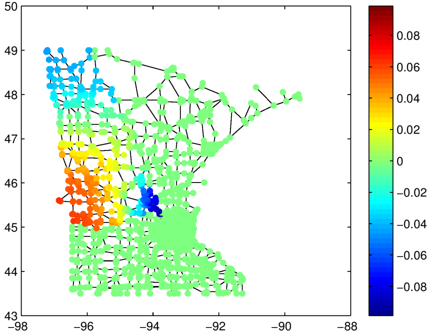
Level $j = 2$, Region $k = 0$, $l = 1$

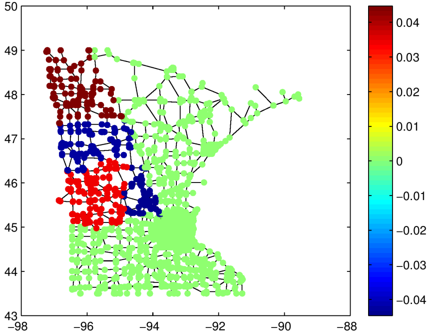


HGLET vs. GHWT

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

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

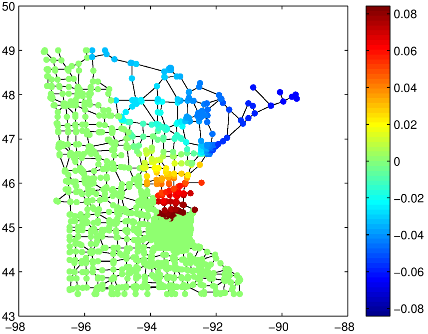


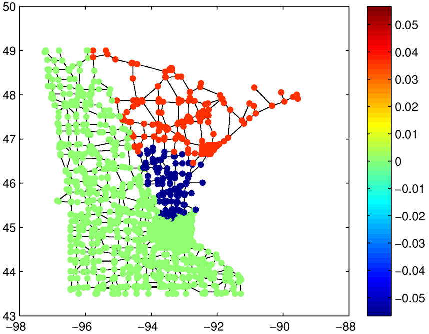


HGLET vs. GHWT

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

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

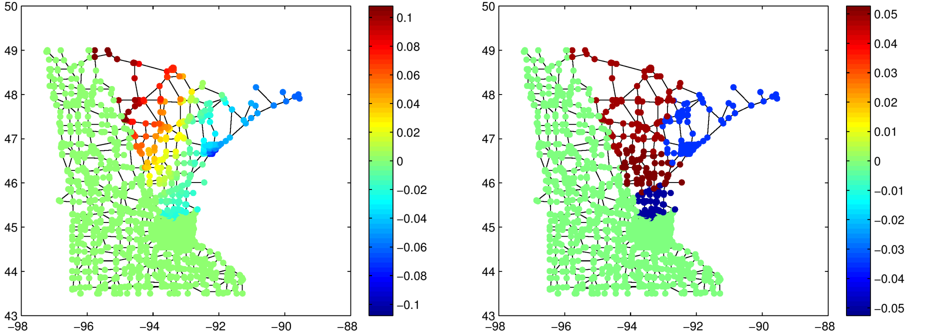




HGLET vs. GHWT

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

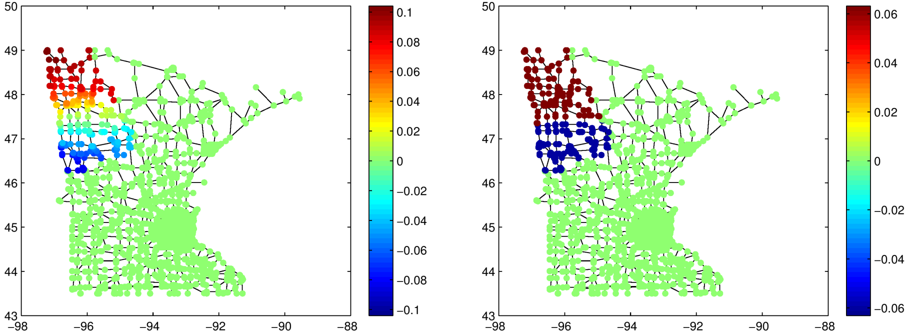
Level $j = 2$, Region $k = 1$, $l = 2$



HGLET vs. GHWT

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

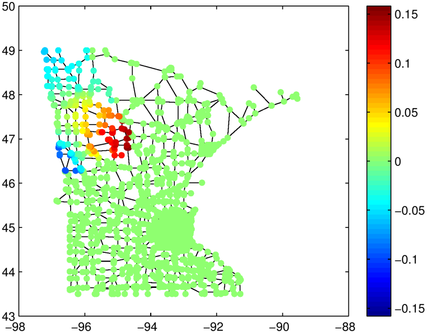
Level $j = 3$, Region $k = 0$, $l = 1$

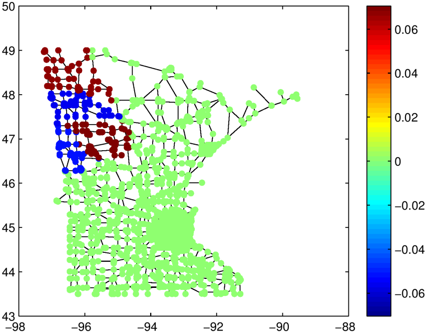


HGLET vs. GHWT

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

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

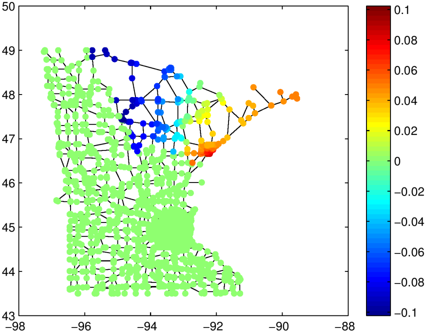


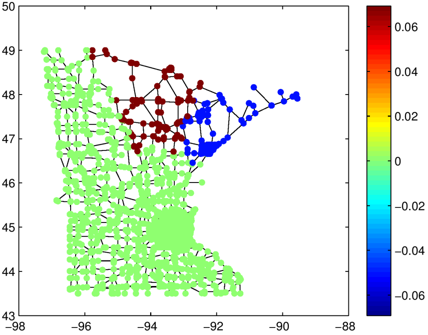


HGLET vs. GHWT

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

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

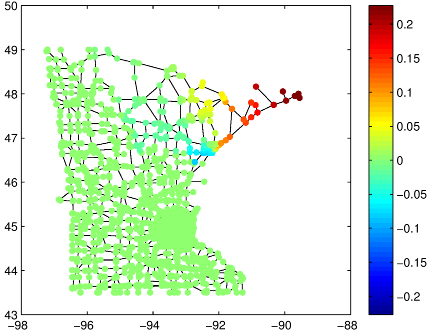


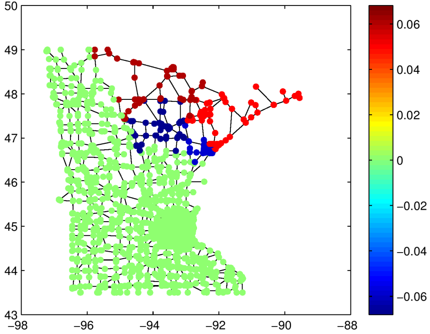


HGLET vs. GHWT

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

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





Computational Complexity: GHWT

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

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

Related Work

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$d^0_{0,0} \quad d^0_{0,1} \quad d^0_{0,2} \quad \cdots \quad d^0_{0,N^0_0-1}$$

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

$$d^1_{0,0} \quad d^1_{0,1} \quad d^1_{0,2} \quad \cdots \quad d^1_{0,N^1_0-1} \qquad d^1_{1,0} \quad d^1_{1,1} \quad d^1_{1,2} \quad \cdots \quad d^1_{1,N^1_1-1}$$

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

$$d^2_{2,0} \; d^2_{2,1} \cdots d^2_{2,N^2_2-1} \qquad d^2_{3,0} \; d^2_{3,1} \cdots d^2_{3,N^2_3-1}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

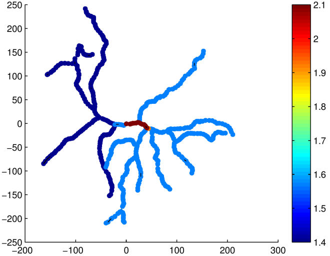
According to cost functional $\mathscr{J}$, this is the best basis for approximation.

$$\left[\begin{matrix} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,N^1_0-1} \end{matrix}\right]$$
$$\begin{matrix} d^1_{0,0} & d^1_{0,1} & d^1_{0,2} & \cdots & d^1_{0,N^1_0-1} \end{matrix}$$

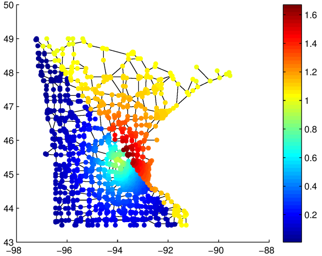
$$\left[\boldsymbol{\phi}^2_{2,0}\,\boldsymbol{\phi}^2_{2,1}\cdots\boldsymbol{\phi}^2_{2,N^2_2-1}\right] \quad \left[\boldsymbol{\phi}^2_{3,0}\,\boldsymbol{\phi}^2_{3,1}\cdots\boldsymbol{\phi}^2_{3,N^2_3-1}\right]$$
$$d^2_{2,0}\; d^2_{2,1}\cdots d^2_{2,N^2_2-1} \quad d^2_{3,0}\; d^2_{3,1}\cdots d^2_{3,N^2_3-1}$$

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

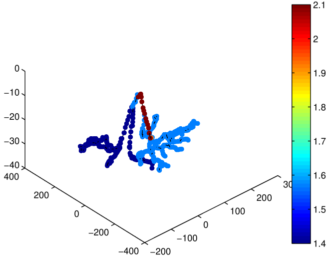
- With the GHWT bases, we run the best-basis algorithm on both the default (coarse-to-fine) dictionary and the reorganized (fine-to-coarse) dictionary and then compare the cost of the 2 bases to determine the best-basis.



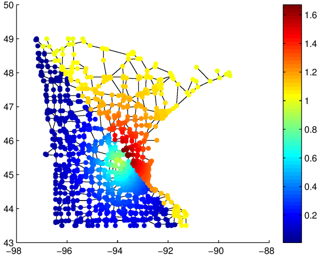
(a) Thickness data on a dendritic tree



(b) A mutilated Gaussian on the MN road network

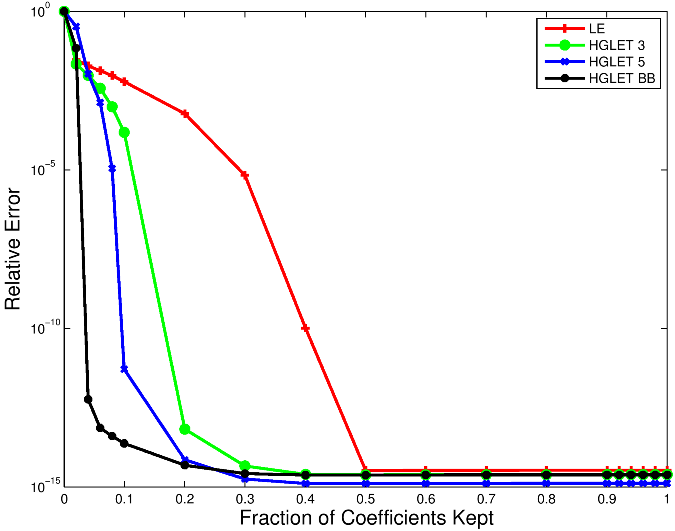


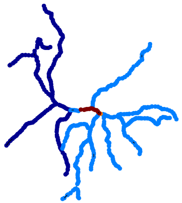
(a) Thickness data on a dendritic tree



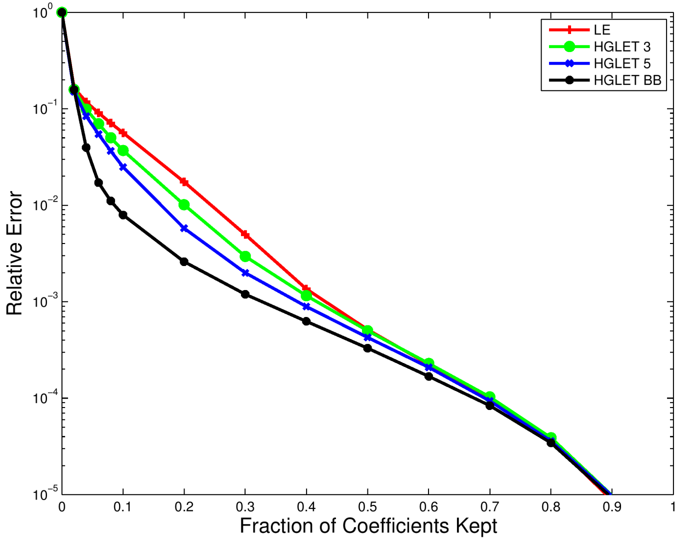
(b) A mutilated Gaussian on the MN road network

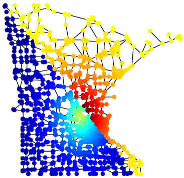
HGLET on Dendrite (weights = inv. Euclidean dist.)



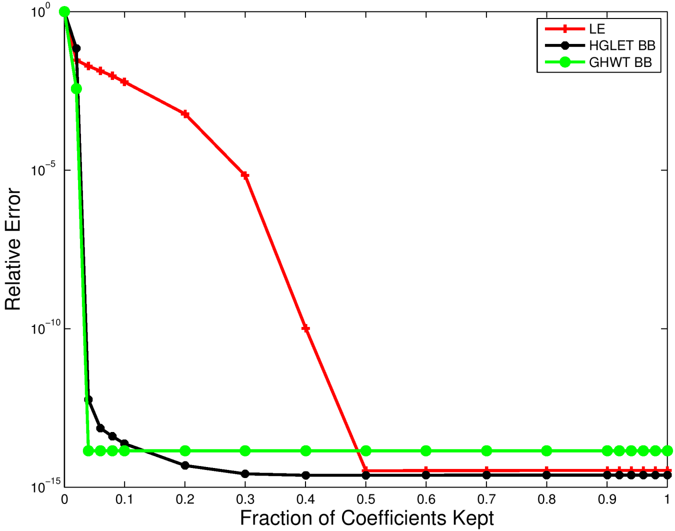


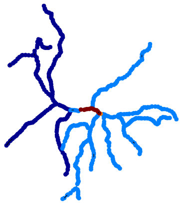
HGLET on MN Mutilated Gaussian (weights = inv. Euclidean dist.)



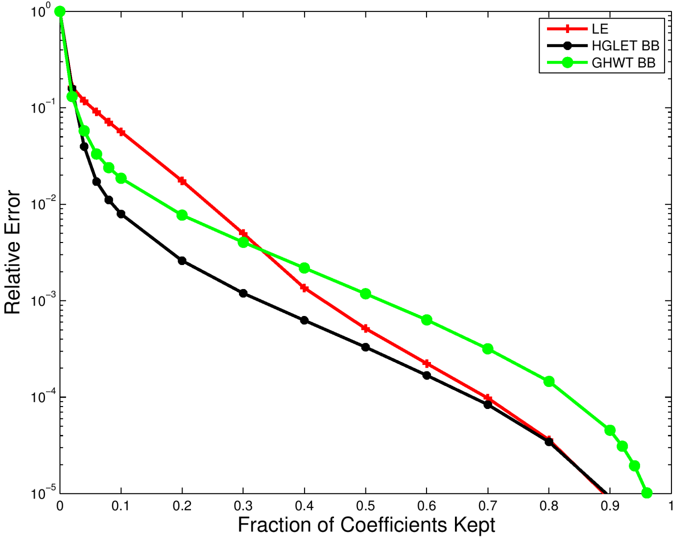


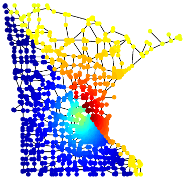
GHWT vs. HGLET on Dendrite





GHWT vs. HGLET on MN Mutilated Gaussian





Discussion of Approximation Results

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

Discussion of Approximation Results

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

Discussion of Approximation Results

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

Discussion of Approximation Results

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

Summary

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

Summary

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

Summary

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

Summary

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

Summary

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

Further Developments

- A good signal segmentation algorithm based on HGLET
- Matrix data analysis (e.g., *term-document matrices*) using the GHWT best basis
- Generalization of *adapted time-frequency tilings* to the graph setting $\Longrightarrow$ *eGHWT*
- Generalization of *Shannon wavelet packets* to the graph setting $\Longrightarrow$ *Natural Graph Wavelet Packets* by hierarchical groupings of the graph Laplacian eigenvectors
- Generalization of *GHWT* and *HGLET* to signals measured on *edges, triangles, tetrahedra, ...* instead of nodes
- For the details of above projects, please check our papers at my website!

Further Developments

- A good signal segmentation algorithm based on HGLET
- Matrix data analysis (e.g., *term-document matrices*) using the GHWT best basis
- Generalization of *adapted time-frequency tilings* to the graph setting $\Longrightarrow$ *eGHWT*
- Generalization of *Shannon wavelet packets* to the graph setting $\Longrightarrow$ *Natural Graph Wavelet Packets* by hierarchical groupings of the graph Laplacian eigenvectors
- Generalization of *GHWT* and *HGLET* to signals measured on *edges, triangles, tetrahedra, . . .* instead of nodes
- For the details of above projects, please check our papers at my website!

Further Developments

- A good signal segmentation algorithm based on HGLET
- Matrix data analysis (e.g., *term-document matrices*) using the GHWT best basis
- Generalization of *adapted time-frequency tilings* to the graph setting $\Longrightarrow$ *eGHWT*
- Generalization of *Shannon wavelet packets* to the graph setting $\Longrightarrow$ *Natural Graph Wavelet Packets* by hierarchical groupings of the graph Laplacian eigenvectors
- Generalization of *GHWT* and *HGLET* to signals measured on *edges, triangles, tetrahedra, . . .* instead of nodes
- For the details of above projects, please check our papers at my website!

Further Developments

- A good signal segmentation algorithm based on HGLET
- Matrix data analysis (e.g., *term-document matrices*) using the GHWT best basis
- Generalization of *adapted time-frequency tilings* to the graph setting $\Longrightarrow$ *eGHWT*
- Generalization of *Shannon wavelet packets* to the graph setting $\Longrightarrow$ *Natural Graph Wavelet Packets* by hierarchical groupings of the graph Laplacian eigenvectors
- Generalization of *GHWT* and *HGLET* to signals measured on *edges, triangles, tetrahedra, . . .* instead of nodes
- For the details of above projects, please check our papers at my website!

Further Developments

- A good signal segmentation algorithm based on HGLET
- Matrix data analysis (e.g., *term-document matrices*) using the GHWT best basis
- Generalization of *adapted time-frequency tilings* to the graph setting $\Longrightarrow$ *eGHWT*
- Generalization of *Shannon wavelet packets* to the graph setting $\Longrightarrow$ *Natural Graph Wavelet Packets* by hierarchical groupings of the graph Laplacian eigenvectors
- Generalization of *GHWT* and *HGLET* to signals measured on *edges, triangles, tetrahedra, . . .* instead of nodes
- For the details of above projects, please check our papers at my website!

Further Developments

- A good signal segmentation algorithm based on HGLET
- Matrix data analysis (e.g., *term-document matrices*) using the GHWT best basis
- Generalization of *adapted time-frequency tilings* to the graph setting $\Longrightarrow$ *eGHWT*
- Generalization of *Shannon wavelet packets* to the graph setting $\Longrightarrow$ *Natural Graph Wavelet Packets* by hierarchical groupings of the graph Laplacian eigenvectors
- Generalization of *GHWT* and *HGLET* to signals measured on *edges, triangles, tetrahedra, . . .* instead of nodes
- For the details of above projects, please check our papers at my website!

References

- http://www.math.ucdavis.edu/~saito/courses/HarmGraph/ contains my course slides and useful information on "Harmonic Analysis on Graphs and Networks"
- Also visit http://www.math.ucdavis.edu/~saito/publications/ and check, e.g.:

  - J. Irion & N. Saito: "Hierarchical graph Laplacian eigen transforms," *JSIAM Letters*, vol. 6, pp. 21–24, 2014.
  - J. Irion & N. Saito: "The generalized Haar-Walsh transform," *Proc. 2014 IEEE Workshop on Statistical Signal Processing*, pp. 488-491, 2014.
  - J. Irion & N. Saito: "Applied and computational harmonic analysis on graphs and networks," in *Wavelets and Sparsity XVI, Proc. SPIE 9597*, Paper # 95971F, 2015.
  - J. Irion & N. Saito: "Efficient approximation and denoising of graph signals using the multiscale basis dictionaries," *IEEE Trans. Signal and Inform. Process. Netw.*, vol. 3, no. 3, pp. 607–616, 2017.
  - J. Irion & N. Saito: "Learning sparsity and structure of matrices with multiscale graph basis dictionaries," in *Proc. the 2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6, 2016.
  - A. Cloninger, H. Li, & N. Saito: "Natural graph wavelet packet dictionaries," *Journal of Fourier Analysis and Applications*, vol. 27, Article #41, 2021.
  - N. Saito & Y. Shao: "eGHWT: The extended Generalized Haar-Walsh Transform," *Journal of Mathematical Imaging and Vision*, vol. 64, no. 3, pp. 261–283, 2022.
  - N. Saito, S. C. Schonsheck, & E. Shvarts: "Multiscale transforms for signals on simplicial complexes," *arXiv:2301.02136 [cs.SI]*, 2023.

Acknowledgment

- Support from Office of Naval Research grants: N00014-20-1-2381; N00014-12-1-0177; N00014-16-1-2255
- Support from National Science Foundation grants: CCF-1934568; DMS-1912747; DMS-1418779
- Support for Jeff Irion from National Defense Science and Engineering Graduate Fellowship, 32 CFR 168a via AFOSR FA9550-11-C-0028