The Hierarchical Graph Laplacian Eigen Transform (HGLET) and Its Relatives for Data Analysis on Graphs and Networks

Naoki Saito & Jeff Irion

Department of Mathematics
University of California, Davis

AMS Special Session on Applied Harmonic Analysis:
Large Data Sets, Signal Processing, and Inverse Problems

Joint Mathematics Meetings, Baltimore, MD

January 15, 2014
Motivations & Aims

Background
- Basic Graph Theory Terminology
- Graph Laplacians
- Graph Partitioning via Spectral Clustering

Multiscale Transforms
- Hierarchical Graph Laplacian Eigen Transform (HGLET)
- Generalized Haar-Walsh Transform (GHWT)

Best-Basis Algorithm for HGLET & GHWT

Approximation Experiments

Summary and Future Work
Motivations & Aims

Background
- Basic Graph Theory Terminology
- Graph Laplacians
- Graph Partitioning via Spectral Clustering

Multiscale Transforms
- Hierarchical Graph Laplacian Eigen Transform (HGLET)
- Generalized Haar-Walsh Transform (GHWT)

Best-Basis Algorithm for HGLET & GHWT

Approximation Experiments

Summary and Future Work
Motivations

Wavelets

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

For example:

- Coifman and Maggioni (2006): diffusion wavelets
  - Bremer et al. (2006): diffusion wavelet packets

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

Common strategy: Develop wavelet-like multiscale transforms
Motivations

Wavelets

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

For example:

- 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
Motivations

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

For example:
- 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
Motivations

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

For example:
- 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
Aims & Objectives

- 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 talk, we will focus on how to construct such dictionaries on graphs and demonstrate their usefulness for data approximation on graphs.
Aims & Objectives

- 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 talk, we will focus on how to construct such dictionaries on graphs and demonstrate their usefulness for data approximation on graphs.
Aims & Objectives

- 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 talk, we will focus on how to construct such dictionaries on graphs and demonstrate their usefulness for data approximation on graphs.
1 Motivations & Aims

2 Background
   • Basic Graph Theory Terminology
   • Graph Laplacians
   • Graph Partitioning via Spectral Clustering

3 Multiscale Transforms
   • Hierarchical Graph Laplacian Eigen Transform (HGLET)
   • Generalized Haar-Walsh Transform (GHWT)

4 Best-Basis Algorithm for HGLET & GHWT

5 Approximation Experiments

6 Summary and Future Work
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 $1 \leq i, j \leq N$.
- $W = W(G) \in \mathbb{R}^{N \times N}$ is the weight matrix, where $w_{ij}$ denotes the edge weight between vertices $i$ and $j$. 
Definitions and Notation

Let $G$ be a graph.

- $V = V(G) = \{v_1, \ldots, v_N\}$ 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 $1 \leq i, j \leq N$.
- $W = W(G) \in \mathbb{R}^{N \times N}$ is the weight matrix, where $w_{ij}$ denotes the edge weight between vertices $i$ and $j$. 

![Graph example]
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 $1 \leq i, j \leq N$.
- $W = W(G) \in \mathbb{R}^{N \times N}$ is the weight matrix, where $w_{ij}$ denotes the edge weight between vertices $i$ and $j$. 

\[ \begin{array}{ccc}
\text{1} & \text{2} & \text{3} \\
\text{5} & & \text{4} \\
& \text{3} & \\
\end{array} \]
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 $1 \leq i, j \leq N$.
- $W = W(G) \in \mathbb{R}^{N \times N}$ is the weight matrix, where $w_{ij}$ denotes the edge weight between vertices $i$ and $j$. 

![Graph Diagram]

saito@math.ucdavis.edu (UC Davis)  Graph Laplacian Eigen Transforms  Jan. 15, 2014  7 / 46
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

\[
    w_{ij} := \frac{1}{\text{dist}(v_i, v_j)} \text{ or } \exp\left(-\frac{\text{dist}(v_i, v_j)^2}{\epsilon^2}\right),
\]

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

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

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

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

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

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

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

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

In this talk, we assume that the graph is

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

The graph may be weighted or unweighted.
Our Assumptions

In this talk, we assume that the graph is

- **connected**. Otherwise, we would simply consider the components separately.

- **undirected**. Edges do not have direction, which means that $w_{ij} = w_{ji}$ and thus $W$ is *symmetric*.

The graph may be weighted or unweighted.
Our Assumptions

In this talk, we assume that the graph is

- **connected**. Otherwise, we would simply consider the components separately.

- **undirected**. Edges do not have direction, which means that $w_{ij} = w_{ji}$ and thus $W$ is symmetric.

The graph may be weighted or unweighted.
Our Assumptions

In this talk, we assume that the graph is

- **connected**. Otherwise, we would simply consider the components separately.

- **undirected**. Edges do not have direction, which means that $w_{ij} = w_{ji}$ and thus $W$ is symmetric.

The graph may be weighted or unweighted.
1 Motivations & Aims

2 Background
   - Basic Graph Theory Terminology
   - Graph Laplacians
   - Graph Partitioning via Spectral Clustering

3 Multiscale Transforms
   - Hierarchical Graph Laplacian Eigen Transform (HGLET)
   - Generalized Haar-Walsh Transform (GHWT)

4 Best-Basis Algorithm for HGLET & GHWT

5 Approximation Experiments

6 Summary and Future Work
Graph Laplacians

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

We have:

- sorted eigenvalues \( 0 = \lambda_0 < \lambda_1 \leq \cdots \leq \lambda_{N-1} \)
- associated eigenvectors \( \phi_0, \phi_1, \ldots, \phi_{N-1} \)

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

- since \( L \) is symmetric, the eigenvectors form an orthonormal basis
- \( \phi_0 = 1/\sqrt{N} \)
Graph Laplacians

\[
\begin{align*}
W(G) &= (w_{ij}) \quad \text{the weight matrix} \\
D(G) &:= \text{diag}(d_{v_1}, \ldots, d_{v_n}) \quad \text{the degree matrix, where } d_{v_i} := \sum_{j=1}^{N} w_{ij} \\
L(G) &:= D(G) - W(G) \quad \text{the Laplacian matrix}
\end{align*}
\]

We have:

- sorted eigenvalues \[0 = \lambda_0 < \lambda_1 \leq \cdots \leq \lambda_{N-1}\]
- associated eigenvectors \[\phi_0, \phi_1, \ldots, \phi_{N-1}\]

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

- since \(L\) is symmetric, the eigenvectors form an orthonormal basis
- \(\phi_0 = 1/\sqrt{N}\)
Graph Laplacians

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

\[L(G) := D(G) - W(G) \quad \text{the Laplacian matrix}\]

We have:

- sorted eigenvalues \(0 = \lambda_0 < \lambda_1 \leq \cdots \leq \lambda_{N-1}\)
- associated eigenvectors \(\phi_0, \phi_1, \ldots, \phi_{N-1}\)

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

- since \(L\) is symmetric, the eigenvectors form an orthonormal basis
- \(\phi_0 = 1/\sqrt{N}\)
Why Graph Laplacians?

- Let $f \in \mathbb{R}^N$. Then
  \[ Lf(v_i) = d_{v_i}f(v_i) - \sum_{j \neq i} w_{ij}f(v_j). \]

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

- After all, sines (cosines) are the eigenfunctions of the Laplacian on the rectangular domain with Dirichlet (Neumann) boundary conditions.

- Hence, the expansion of data measured at the vertices using the eigenvectors of a graph Laplacian can be viewed as Fourier (or spectral) analysis of the data on that graph.
Why Graph Laplacians?

• Let $f \in \mathbb{R}^N$. Then

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

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

• After all, sines (cosines) are the eigenfunctions of the Laplacian on the rectangular domain with Dirichlet (Neumann) boundary conditions.

• Hence, the expansion of data measured at the vertices using the eigenvectors of a graph Laplacian can be viewed as Fourier (or spectral) analysis of the data on that graph.
Why Graph Laplacians?

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

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

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

- After all, sines (cosines) are the eigenfunctions of the Laplacian on the rectangular domain with Dirichlet (Neumann) boundary conditions.

- Hence, the expansion of data measured at the vertices using the eigenvectors of a graph Laplacian can be viewed as Fourier (or spectral) analysis of the data on that graph.
A Simple Yet Important Example: A Path Graph

\[
L(G) = \begin{bmatrix}
1 & -1 \\
-1 & 2 & -1 \\
-1 & 2 & -1 \\
& & \ddots & \ddots \\
-1 & 2 & -1 \\
& & & 1
\end{bmatrix}
\]

\[
D(G) - A(G) = \begin{bmatrix}
1 & 2 \\
2 & \ddots \\
& & 2 \\
& & & 1
\end{bmatrix}
\]

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

- \( \lambda_k = 2 - 2 \cos(\pi k/N) = 4 \sin^2(\pi k/2N) \), \( k = 0, 1, \ldots, N-1 \).
- \( \phi_k(\ell) = \sqrt{2/N} \cos(\pi k(\ell + \frac{1}{2})/N) \), \( k, \ell = 0, 1, \ldots, N-1 \).
- \( \lambda \) (eigenvalue) is a monotonic function w.r.t. \( k \) (frequency). However, for general graphs, \( \lambda \) does not have a simple relationship with \( k \).
A Simple Yet Important Example: A Path Graph

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

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

Background
- Basic Graph Theory Terminology
- Graph Laplacians
- Graph Partitioning via Spectral Clustering

Multiscale Transforms
- Hierarchical Graph Laplacian Eigen Transform (HGLET)
- Generalized Haar-Walsh Transform (GHWT)

Best-Basis Algorithm for HGLET & GHWT

Approximation Experiments

Summary and Future Work
**Goal:** split the vertices $V$ into two “good” subsets, $X$ and $X^c$

**Plan:** use the signs of the entries in $\phi_1$, which is known as the Fiedler vector

**Why?** Using $\phi_1$ to generate $X$ and $X^c$ yields an approximate minimizer of the RatioCut function$^{1,2}$:

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

where

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

---


$^2$We could also use the signs of $\phi_1$ for $L_{rw}$ (equivalently, $L_{sym}$), which yield an approximate minimizer of the Normalized Cut function.
**Goal:** split the vertices $V$ into two “good” subsets, $X$ and $X^c$

**Plan:** use the signs of the entries in $\phi_1$, which is known as the Fiedler vector

**Why?** Using $\phi_1$ to generate $X$ and $X^c$ yields an approximate minimizer of the RatioCut function$^1, 2$:

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

where

$$\text{cut}(X, X^c) := \sum_{\substack{v_i \in X \atop v_j \in X^c}} W_{i,j}$$

---


$^2$We could also use the signs of $\phi_1$ for $L_{rw}$ (equivalently, $L_{sym}$), which yield an approximate minimizer of the Normalized Cut function.
**Goal:** split the vertices $V$ into two “good” subsets, $X$ and $X^c$

**Plan:** use the signs of the entries in $\phi_1$, which is known as the Fiedler vector

**Why?** Using $\phi_1$ to generate $X$ and $X^c$ yields an approximate minimizer of the RatioCut function$^{1,2}$:

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

where

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

---


$^2$We could also use the signs of $\phi_1$ for $L_{\text{rw}}$ (equivalently, $L_{\text{sym}}$), which yield an approximate minimizer of the Normalized Cut function.
Example of Graph Partitioning

Figure: The MN road network
Example of Graph Partitioning

Figure: The MN road network partitioned via the Fiedler vector of $L$
Motivations & Aims

Background
- Basic Graph Theory Terminology
- Graph Laplacians
- Graph Partitioning via Spectral Clustering

Multiscale Transforms
- Hierarchical Graph Laplacian Eigen Transform (HGLET)
- Generalized Haar-Walsh Transform (GHWT)

Best-Basis Algorithm for HGLET & GHWT

Approximation Experiments

Summary and Future Work
Our transforms involve 2 main steps:

1. **Recursively partition the graph**

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

2. **Using the regions on each level of the graph partitioning, generate a set of orthonormal bases for the graph**
Our transforms involve 2 main steps:

1. Recursively partition the graph

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

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

2 Background
   - Basic Graph Theory Terminology
   - Graph Laplacians
   - Graph Partitioning via Spectral Clustering

3 Multiscale Transforms
   - Hierarchical Graph Laplacian Eigen Transform (HGLET)
   - Generalized Haar-Walsh Transform (GHWT)

4 Best-Basis Algorithm for HGLET & GHWT

5 Approximation Experiments

6 Summary and Future Work
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 $\phi_{k,l}^j$ with $j = 0$)

2. Partition the graph using the Fiedler vector $\phi_{k,1}^j$

3. Generate an orthonormal basis for each of the partitions ⇒ Laplacian eigenvectors

4. Repeat...

5. Select an orthonormal basis from this collection of orthonormal bases

\[
\begin{bmatrix}
\phi_{0,0}^0 & \phi_{0,1}^0 & \phi_{0,2}^0 & \cdots & \phi_{0,N_0^0-1}^0 \\
\end{bmatrix}
\]
1. Generate an orthonormal basis for the entire graph ⇒ **Laplacian eigenvectors** (Notation is $\phi_{k,l}^j$ with $j = 0$)

2. Partition the graph using the Fiedler vector $\phi^j_{k,1}$

3. Generate an orthonormal basis for each of the partitions ⇒ **Laplacian eigenvectors**

4. Repeat...

5. Select an orthonormal basis from this collection of orthonormal bases

\[
\begin{bmatrix}
\phi^0_{0,0} & \phi^0_{0,1} & \phi^0_{0,2} & \cdots & \phi^0_{0,N_0^0-1} \\
\phi^1_{0,0} & \phi^1_{0,1} & \phi^1_{0,2} & \cdots & \phi^1_{0,N_0^1-1} \\
\phi^2_{0,0} & \phi^2_{0,1} & \phi^2_{0,2} & \cdots & \phi^2_{0,N_0^2-1} \\
\phi^3_{0,0} & \phi^3_{0,1} & \phi^3_{0,2} & \cdots & \phi^3_{0,N_0^3-1}
\end{bmatrix}
\]
1. Generate an orthonormal basis for the entire graph ⇒ Laplacian eigenvectors (Notation is $\phi_{k,l}^j$ with $j = 0$)

2. Partition the graph using the Fiedler vector $\phi_{k,1}^j$

3. Generate an orthonormal basis for each of the partitions ⇒ Laplacian eigenvectors

4. Repeat...

5. Select an orthonormal basis from this collection of orthonormal bases

\[
\begin{bmatrix}
\phi_{0,0}^0 & \phi_{0,1}^0 & \phi_{0,2}^0 & \cdots & \phi_{0,N_0^0-1}^0 \\
\phi_{0,0}^1 & \phi_{0,1}^1 & \phi_{0,2}^1 & \cdots & \phi_{0,N_0^1-1}^1 \\
\phi_{1,0}^1 & \phi_{1,1}^1 & \phi_{1,2}^1 & \cdots & \phi_{1,N_1^1-1}^1
\end{bmatrix}
\]
1. Generate an orthonormal basis for the entire graph ⇒ Laplacian eigenvectors (Notation is $\phi^j_{k,l}$ with $j = 0$)

2. Partition the graph using the Fiedler vector $\phi^j_{k,1}$

3. Generate an orthonormal basis for each of the partitions ⇒ Laplacian eigenvectors

4. Repeat...

5. Select an orthonormal basis from this collection of orthonormal bases

\[
\begin{bmatrix}
\phi^0_{0,0} & \phi^0_{0,1} & \phi^0_{0,2} & \cdots & \phi^0_{0,N_0^0-1} \\
\phi^1_{0,0} & \phi^1_{0,1} & \phi^1_{0,2} & \cdots & \phi^1_{0,N_0^1-1} \\
\phi^2_{0,0} & \phi^2_{0,1} & \phi^2_{0,2} & \cdots & \phi^2_{0,N_2^0-1} \\
\phi^3_{0,0} & \phi^3_{0,1} & \phi^3_{0,2} & \cdots & \phi^3_{0,N_3^0-1}
\end{bmatrix}
\begin{bmatrix}
\phi^1_{1,0} & \phi^1_{1,1} & \phi^1_{1,2} & \cdots & \phi^1_{1,N_1^1-1}
\end{bmatrix}
\]
1. Generate an orthonormal basis for the entire graph ⇒ \textbf{Laplacian eigenvectors} (Notation is $\phi_{k,l}^j$ with $j = 0$)

2. Partition the graph using the Fiedler vector $\phi_{k,1}^j$

3. Generate an orthonormal basis for each of the partitions ⇒ \textbf{Laplacian eigenvectors}

4. Repeat...

5. Select an orthonormal basis from this collection of orthonormal bases

\[
\begin{bmatrix}
\phi_{0,0}^0 & \phi_{0,1}^0 & \phi_{0,2}^0 & \cdots & \phi_{0,N_0^0-1}^0 \\
\phi_{1,0}^1 & \phi_{1,1}^1 & \phi_{1,2}^1 & \cdots & \phi_{1,N_1^1-1}^1 \\
\phi_{2,0}^2 & \phi_{2,1}^2 & \phi_{2,2}^2 & \cdots & \phi_{2,N_2^2-1}^2 \\
\phi_{3,0}^3 & \phi_{3,1}^3 & \phi_{3,2}^3 & \cdots & \phi_{3,N_3^3-1}^3 \\
\end{bmatrix}
\]
1. Generate an orthonormal basis for the entire graph ⇒ **Laplacian eigenvectors** (Notation is $\phi^j_{k,l}$ with $j = 0$)

2. Partition the graph using the Fiedler vector $\phi^j_{k,1}$

3. Generate an orthonormal basis for each of the partitions ⇒ **Laplacian eigenvectors**

4. Repeat...

5. Select an orthonormal basis from this collection of orthonormal bases

\[
\begin{bmatrix}
\phi^0_{0,0} & \phi^0_{0,1} & \phi^0_{0,2} & \ldots & \phi^0_{0,N_0^0-1} \\
\phi^1_{0,0} & \phi^1_{0,1} & \phi^1_{0,2} & \ldots & \phi^1_{0,N_0^1-1} \\
\phi^2_{0,0} & \phi^2_{0,1} & \phi^2_{0,2} & \ldots & \phi^2_{0,N_0^2-1} \\
\phi^3_{0,0} & \phi^3_{0,1} & \phi^3_{0,2} & \ldots & \phi^3_{0,N_0^3-1} \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
\phi^1_{1,0} & \phi^1_{1,1} & \phi^1_{1,2} & \ldots & \phi^1_{1,N_1^1-1} \\
\phi^2_{1,0} & \phi^2_{1,1} & \phi^2_{1,2} & \ldots & \phi^2_{1,N_1^2-1} \\
\phi^3_{1,0} & \phi^3_{1,1} & \phi^3_{1,2} & \ldots & \phi^3_{1,N_1^3-1} \\
\end{bmatrix}
\]
1. Generate an orthonormal basis for the entire graph ⇒ Laplacian eigenvectors (Notation is $\phi_{k,l}^j$ with $j = 0$)

2. Partition the graph using the Fiedler vector $\phi_{k,1}^j$

3. Generate an orthonormal basis for each of the partitions ⇒ Laplacian eigenvectors

4. Repeat...

5. Select an orthonormal basis from this collection of orthonormal bases

\[
\begin{bmatrix}
\phi_{0,0}^0 & \phi_{0,1}^0 & \phi_{0,2}^0 & \cdots & \phi_{0,N_0^0-1}^0 \\
\phi_{1,0}^1 & \phi_{1,1}^1 & \phi_{1,2}^1 & \cdots & \phi_{1,N_1^1-1}^1 \\
\phi_{2,0}^2 & \phi_{2,1}^2 & \phi_{2,2}^2 & \cdots & \phi_{2,N_2^2-1}^2 \\
\end{bmatrix}
\]
1. Generate an orthonormal basis for the entire graph ⇒ Laplacian eigenvectors (Notation is $\phi_{k,l}^j$ with $j = 0$)
2. Partition the graph using the Fiedler vector $\phi_{k,1}^j$
3. Generate an orthonormal basis for each of the partitions ⇒ Laplacian eigenvectors
4. Repeat...
5. Select an orthonormal basis from this collection of orthonormal bases

\[
\begin{bmatrix}
\phi_{0,0}^0 & \phi_{0,1}^0 & \phi_{0,2}^0 & \cdots & \phi_{0,N_0^0-1}^0 \\
\phi_{0,0}^1 & \phi_{0,1}^1 & \phi_{0,2}^1 & \cdots & \phi_{0,N_0^1-1}^1 \\
\phi_{1,0}^1 & \phi_{1,1}^1 & \phi_{1,2}^1 & \cdots & \phi_{1,N_1^1-1}^1 \\
\phi_{0,0}^2 & \phi_{0,1}^2 & \cdots & \phi_{0,N_0^2-1}^2 \\
\phi_{1,0}^2 & \phi_{1,1}^2 & \cdots & \phi_{1,N_1^2-1}^2 \\
\phi_{2,0}^2 & \phi_{2,1}^2 & \cdots & \phi_{2,N_2^2-1}^2 \\
\phi_{3,0}^2 & \phi_{3,1}^2 & \cdots & \phi_{3,N_3^2-1}^2 \\
\vdots & & & & \ddots
\end{bmatrix}
\]
1. Generate an orthonormal basis for the entire graph ⇒ Laplacian eigenvectors (Notation is $\phi_{k,l}^j$ with $j = 0$)

2. Partition the graph using the Fiedler vector $\phi_{k,1}^j$

3. Generate an orthonormal basis for each of the partitions ⇒ Laplacian eigenvectors

4. Repeat...

5. Select an orthonormal basis from this collection of orthonormal bases

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

\[
\begin{bmatrix}
\phi_{1,0}^1 & \phi_{1,1}^1 & \phi_{1,2}^1 & \cdots & \phi_{1,N_1^1-1}^1 \\
\phi_{1,0}^2 & \phi_{1,1}^2 & \phi_{1,2}^2 & \cdots & \phi_{1,N_1^2-1}^2 \\
\phi_{1,0}^3 & \phi_{1,1}^3 & \phi_{1,2}^3 & \cdots & \phi_{1,N_1^3-1}^3 \\
\end{bmatrix}
\]

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

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

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

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

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

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

\[
\begin{bmatrix}
\phi_{0,0}^0 & \phi_{0,1}^0 & \phi_{0,2}^0 & \cdots & \phi_{0,N_0^0-1}^0 \\
\phi_{0,0}^1 & \phi_{0,1}^1 & \phi_{0,2}^1 & \cdots & \phi_{0,N_0^1-1}^1 \\
\phi_{0,0}^2 & \cdots & \phi_{0,N_0^2-1}^2 & \phi_{1,0}^2 & \phi_{1,1}^2 & \phi_{1,2}^2 & \cdots & \phi_{1,N_1^1-1}^2 \\
\phi_{0,0}^3 & \cdots & \phi_{0,N_0^3-1}^3 & \phi_{1,0}^3 & \phi_{1,1}^3 & \phi_{1,2}^3 & \cdots & \phi_{1,N_1^2-1}^3 \\
\phi_{0,0}^4 & \cdots & \phi_{0,N_0^4-1}^4 & \phi_{1,0}^4 & \phi_{1,1}^4 & \phi_{1,2}^4 & \cdots & \phi_{1,N_1^3-1}^4
\end{bmatrix}
\]
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

\[
\begin{bmatrix}
\phi^0_{0,0} & \phi^0_{0,1} & \phi^0_{0,2} & \cdots & \phi^0_{0,N_0^0-1} \\
\phi^1_{0,0} & \phi^1_{0,1} & \phi^1_{0,2} & \cdots & \phi^1_{0,N_1^1-1} \\
\phi^2_{0,0} & \cdots & \phi^2_{0,N_0^2-1} & \cdots & \phi^2_{0,N_2^3-1} \\
\end{bmatrix}
\begin{bmatrix}
\phi^0_{1,0} & \phi^0_{1,1} & \phi^0_{1,2} & \cdots & \phi^0_{1,N_0^0-1} \\
\phi^1_{1,0} & \phi^1_{1,1} & \phi^1_{1,2} & \cdots & \phi^1_{1,N_1^1-1} \\
\phi^2_{1,0} & \cdots & \phi^2_{1,N_0^2-1} & \cdots & \phi^2_{1,N_2^3-1} \\
\end{bmatrix}
\begin{bmatrix}
\phi^0_{2,0} & \cdots & \phi^0_{2,N_0^2-1} & \cdots & \phi^0_{2,N_2^3-1} \\
\phi^1_{2,0} & \cdots & \phi^1_{2,N_0^2-1} & \cdots & \phi^1_{2,N_2^3-1} \\
\phi^2_{2,0} & \cdots & \phi^2_{2,N_0^2-1} & \cdots & \phi^2_{2,N_2^3-1} \\
\end{bmatrix}
\]
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 $\Rightarrow$ best-basis algorithm, local discriminant basis algorithm, ...
- A union of bases on disjoint subsets is obviously orthonormal

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

- For an unweighted path graph, this yields a dictionary of the block DCT-II.

- Similar to wavelet packet or local cosine dictionaries in that it generates an *overcomplete basis* from which we can select a basis useful for the task at hand \( \Rightarrow \) best-basis algorithm, local discriminant basis algorithm, \ldots

- A union of bases on disjoint subsets is obviously orthonormal

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

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

Related Work

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


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 the 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

<table>
<thead>
<tr>
<th>HGLET (redundant)</th>
<th>Computational Complexity</th>
<th>Run Time for MN$^1$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$O(N^3)$</td>
<td>67 sec</td>
</tr>
</tbody>
</table>

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

2 Background
- Basic Graph Theory Terminology
- Graph Laplacians
- Graph Partitioning via Spectral Clustering

3 Multiscale Transforms
- Hierarchical Graph Laplacian Eigen Transform (HGLET)
- Generalized Haar-Walsh Transform (GHWT)

4 Best-Basis Algorithm for HGLET & GHWT

5 Approximation Experiments

6 Summary and Future Work
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...
HGLET is a generalization of the block DCT, and it generates basis vectors that are \textit{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 \textit{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 $j_{\text{max}}$ (the finest level) ⇒ *scaling vectors* on the single-node regions
   - As with HGLET, the notation is $\psi_{k,l}^j$

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

$$
\begin{bmatrix}
\psi_{0,0}^{j_{\text{max}}} & \psi_{1,0}^{j_{\text{max}}} & \psi_{2,0}^{j_{\text{max}}} & \psi_{3,0}^{j_{\text{max}}} & \cdots & \psi_{K_{j_{\text{max}}-1},0}^{j_{\text{max}}-2} & \psi_{K_{j_{\text{max}}-1},0}^{j_{\text{max}}-1}
\end{bmatrix}
$$
1. Generate a full recursive partitioning of the graph ⇒ Fiedler vectors
2. Generate an orthonormal basis for level $j_{\text{max}}$ (the finest level) ⇒ *scaling vectors* on the single-node regions
   - As with HGLET, the notation is $\psi^j_{k,l}$
3. Using the basis for level $j_{\text{max}}$, generate an orthonormal basis for level $j_{\text{max}} - 1$ ⇒ *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

\[
\begin{bmatrix}
\psi^0_{0,0} & \psi^0_{0,1} \\
\psi^0_{1,0} & \psi^0_{1,1} \\
\vdots & \vdots \\
\psi^{K_{j_{\text{max}}}-2,0} & \psi^{K_{j_{\text{max}}}-1,0} \\
\psi^{0,0} & \psi^{1,0} & \psi^{2,0} & \psi^{3,0} & \ldots & \psi^{K_{j_{\text{max}}}-2,0} & \psi^{K_{j_{\text{max}}}-1,0} \\
\end{bmatrix}
\]
1. Generate a full recursive partitioning of the graph ⇒ Fiedler vectors
2. Generate an orthonormal basis for level $j_{\text{max}}$ (the finest level) ⇒ *scaling vectors* on the single-node regions
   - As with HGLET, the notation is $\psi_{j,k,l}^j$
3. Using the basis for level $j_{\text{max}}$, generate an orthonormal basis for level $j_{\text{max}} - 1$ ⇒ *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

\[
\begin{bmatrix}
\psi_{0,0}^0 & \psi_{0,1}^0 & \psi_{0,2}^0 & \psi_{0,3}^0 & \cdots & \psi_{0,N-2}^0 & \psi_{0,N-1}^0 \\
\vdots & & & & & & \\
\psi_{j_{\text{max}}-1,0}^{j_{\text{max}}-1} & \psi_{j_{\text{max}}-1,1}^{j_{\text{max}}-1} & \cdots & \psi_{K_{j_{\text{max}}-1}^{j_{\text{max}}-1},0}^{j_{\text{max}}-1} & \psi_{K_{j_{\text{max}}-1}^{j_{\text{max}}-1},1}^{j_{\text{max}}-1} \\
\psi_{j_{\text{max}},0} & \psi_{j_{\text{max}},1} & \psi_{j_{\text{max}},2} & \psi_{j_{\text{max}},3} & \cdots & \psi_{K_{j_{\text{max}}-1}^{j_{\text{max}}-1},0}^{j_{\text{max}}-1} & \psi_{K_{j_{\text{max}}-1}^{j_{\text{max}}-1},1}^{j_{\text{max}}-1} \\
\end{bmatrix}
\]
Generate a full recursive partitioning of the graph ⇒ Fiedler vectors

Generate an orthonormal basis for level \( j_{\text{max}} \) (the finest level) ⇒ *scaling vectors* on the single-node regions

- As with HGLET, the notation is \( \psi_{k,l}^j \)

Using the basis for level \( j_{\text{max}} \), generate an orthonormal basis for level \( j_{\text{max}} - 1 \) ⇒ *scaling* and *Haar-like* vectors

Repeat... Using the basis for level \( j \), generate an orthonormal basis for level \( j - 1 \) ⇒ *scaling*, *Haar-like*, and *Walsh-like* vectors

Select an orthonormal basis from this collection of orthonormal bases

\[
\begin{bmatrix}
\psi_{0,0}^0 & \psi_{0,1}^0 & \psi_{0,2}^0 & \psi_{0,3}^0 & \cdots & \psi_{0,N-2}^0 & \psi_{0,N-1}^0 \\
\vdots \\
\psi_{j_{\text{max}}-1,0}^{j_{\text{max}}-1} & \psi_{j_{\text{max}}-1,1}^{j_{\text{max}}-1} \\
\psi_{1,0}^{j_{\text{max}}-1} & \psi_{1,1}^{j_{\text{max}}-1} & \cdots & \psi_{K_{j_{\text{max}}-1}-1,0}^{j_{\text{max}}-1} & \psi_{K_{j_{\text{max}}-1}-1,1}^{j_{\text{max}}-1} \\
\psi_{0,0}^{j_{\text{max}}} & \psi_{1,0}^{j_{\text{max}}} & \psi_{2,0}^{j_{\text{max}}} & \psi_{3,0}^{j_{\text{max}}} & \cdots & \psi_{K_{j_{\text{max}}}-2,0}^{j_{\text{max}}} & \psi_{K_{j_{\text{max}}}-1,0}^{j_{\text{max}}}
\end{bmatrix}
\]
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).\(^1\)

---

\(^1\)Full details will be presented in a forthcoming article.
Remarks

- We can also reorder and regroup the vectors on each level of the GHWT dictionary according to their type ({\textit{scaling}}, {\textit{Haar-like}}, or {\textit{Walsh-like}})\(^1\)

\[\text{Figure: Default dictionary; i.e., coarse-to-fine}\]

- This reorganization gives us \textit{more options for choosing a good basis}.

\(^1\)Full details will be presented in a forthcoming article.
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)\(^1\)

![Reordered & regrouped dictionary](image)

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

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

---

\(^1\)Full details will be presented in a forthcoming article.
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.

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

1Full details will be presented in a forthcoming article.
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

<table>
<thead>
<tr>
<th></th>
<th>Computational Complexity</th>
<th>Run Time for MN(^1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HGLET (redundant)</td>
<td>(O(N^3))</td>
<td>67 sec</td>
</tr>
<tr>
<td><strong>GHWT</strong> (redundant)</td>
<td>(O(N^2))</td>
<td>10 sec</td>
</tr>
</tbody>
</table>

\(^1\)Computations performed on a personal laptop (4.00 GB RAM, 2.26 GHz), \(N = 2640\) and \(\text{nnz}(\bar{W}) = 6604\).
The following articles also discussed the Haar-like transform on graphs and trees, but *not the Walsh-Hadamard transform* on them:


1 Motivations & Aims

2 Background
   • Basic Graph Theory Terminology
   • Graph Laplacians
   • Graph Partitioning via Spectral Clustering

3 Multiscale Transforms
   • Hierarchical Graph Laplacian Eigen Transform (HGLET)
   • Generalized Haar-Walsh Transform (GHWT)

4 Best-Basis Algorithm for HGLET & GHWT

5 Approximation Experiments

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

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

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

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

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

saito@math.ucdavis.edu (UC Davis)  Graph Laplacian Eigen Transforms  Jan. 15, 2014  34 / 46
Coifman and Wickerhauser (1992) developed the best-basis algorithm as a means of selecting the basis from a dictionary of wavelet packets that is “best” for approximation/compression.

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

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

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

For our approximation experiments in the following pages, we used $p = 0.1$. 
Coifman and Wickerhauser (1992) developed the best-basis algorithm as a means of selecting the basis from a dictionary of wavelet packets that is “best” for approximation/compression.

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

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

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

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

saito@math.ucdavis.edu (UC Davis)
According to cost functional $J$, 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.
According to cost functional $J$, 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.
According to cost functional $J$, 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.

\[
\begin{bmatrix}
\phi^0_{0,0} & \phi^0_{0,1} & \phi^0_{0,2} & \ldots & \phi^0_{0,N^0_0 - 1} \\
\phi^0_{0,0} & \phi^0_{0,1} & \phi^0_{0,2} & \ldots & \phi^0_{0,N^0_0 - 1} \\
\phi^1_{0,0} & \phi^1_{0,1} & \phi^1_{0,2} & \ldots & \phi^1_{0,N^1_0 - 1} \\
\phi^1_{0,0} & \phi^1_{0,1} & \phi^1_{0,2} & \ldots & \phi^1_{0,N^1_0 - 1} \\
\phi^2_{0,0} & \phi^2_{0,1} & \phi^2_{0,2} & \ldots & \phi^2_{0,N^2_0 - 1} \\
\phi^2_{0,0} & \phi^2_{0,1} & \phi^2_{0,2} & \ldots & \phi^2_{0,N^2_0 - 1} \\
\phi^2_{1,0} & \phi^2_{1,1} & \phi^2_{1,2} & \ldots & \phi^2_{1,N^2_1 - 1} \\
\phi^2_{1,0} & \phi^2_{1,1} & \phi^2_{1,2} & \ldots & \phi^2_{1,N^2_1 - 1} \\
\phi^2_{2,0} & \phi^2_{2,1} & \phi^2_{2,2} & \ldots & \phi^2_{2,N^2_2 - 1} \\
\phi^2_{2,0} & \phi^2_{2,1} & \phi^2_{2,2} & \ldots & \phi^2_{2,N^2_2 - 1} \\
\phi^2_{3,0} & \phi^2_{3,1} & \phi^2_{3,2} & \ldots & \phi^2_{3,N^2_3 - 1} \\
\phi^2_{3,0} & \phi^2_{3,1} & \phi^2_{3,2} & \ldots & \phi^2_{3,N^2_3 - 1}
\end{bmatrix}
\]
According to cost functional $J$, 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.
According to cost functional $J$, 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 two bases to determine the best-basis.
According to the cost functional $J$, 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.
According to cost functional $J$, 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 two bases to determine the best-basis.

Best-Basis Algorithm for HGLET & GHWT
According to cost functional $J$, 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.

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

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

$$
\begin{bmatrix}
\phi_{3,0}^2 & \phi_{3,1}^2 & \cdots & \phi_{3,N_3^2-1}^2 \\
 d_{3,0}^2 & d_{3,1}^2 & \cdots & d_{3,N_3^2-1}^2
\end{bmatrix}
$$
According to cost functional $\mathcal{I}$, this is the best basis for approximation.
According to cost functional $\mathcal{J}$, this is the best basis for approximation.

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

2 Background
  • Basic Graph Theory Terminology
  • Graph Laplacians
  • Graph Partitioning via Spectral Clustering

3 Multiscale Transforms
  • Hierarchical Graph Laplacian Eigen Transform (HGLET)
  • Generalized Haar-Walsh Transform (GHWT)

4 Best-Basis Algorithm for HGLET & GHWT

5 Approximation Experiments

6 Summary and Future Work
(a) Thickness data on a dendritic tree

(b) A mutilated Gaussian on the MN road network
(a) Thickness data on a dendritic tree  
(b) A mutilated Gaussian on the MN road network
HGLET on Dendrite (weights = inv. Euclidean dist.)
HGLET on MN Mutilated Gaussian (weights = inv. Euclidean dist.)
GHWT vs. HGLET on Dendrite
Approximation Experiments

GHWT vs. HGLET on MN Mutilated Gaussian

Graph Laplacian Eigen Transforms

Jan. 15, 2014 41 / 46
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 Multilated 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 Multilated 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 Multilated 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 Multilated Gaussian dataset while the GHWT best-basis outperformed the others on the Dendrite dataset
- These performances make a strong case for using localized basis vectors on multiple scales
- Also, these indicate that the smoothness of the basis vectors matters depending on the smoothness inherent in data
1 Motivations & Aims

2 Background
   • Basic Graph Theory Terminology
   • Graph Laplacians
   • Graph Partitioning via Spectral Clustering

3 Multiscale Transforms
   • Hierarchical Graph Laplacian Eigen Transform (HGLET)
   • Generalized Haar-Walsh Transform (GHWT)

4 Best-Basis Algorithm for HGLET & GHWT

5 Approximation Experiments

6 Summary and Future Work
Summary

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

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

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

saito@math.ucdavis.edu (UC Davis)  Graph Laplacian Eigen Transforms  Jan. 15, 2014  44 / 46
Summary

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

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

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

- Perform classification experiments and compare the results using HGLET and GHWT.
- Explore other methods for graph partitioning:
  - Allow for splitting of a region into an arbitrary number of subregions;
  - Consider a bottom-up clustering method, rather than a top-down partitioning method;
Future Work

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

- http://www.math.ucdavis.edu/~saito/courses/HarmGraph/ contains my course slides and useful information on “Harmonic Analysis on Graphs and Networks”
- Also visit http://www.math.ucdavis.edu/~saito/publications/ for various related publications including:

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