Multiscale Basis Dictionaries on Graphs and Their Applications to Matrix Data Analysis and Signal Segmentation

Jeff Irion & Naoki Saito

Department of Mathematics
University of California, Davis

Applied Mathematics Seminar
Yale University
November 17, 2015

Outline

Acknowledgment

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

Today's Goals

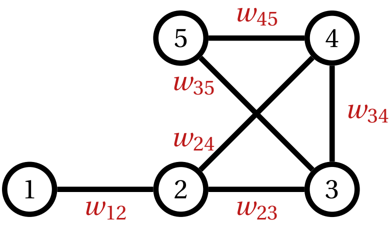
- Briefly review some basic concepts and terminology of graph theory and *graph Laplacians*
- Review our tools that we recently developed:
  - Hierarchical Graph Laplacian Eigen Transform (HGLET) $\approx$ *Hierarchical Block Discrete Cosine Transforms on graphs*;
  - Generalized Haar-Walsh Transform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

Outline

Definitions and Notation

Let G be a graph.

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



Definitions and Notation ...

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

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

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

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

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

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

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

Definitions and Notation ...

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

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

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

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

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

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

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

Our Assumptions

In this talk, we assume that the graph is

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

The graph may be weighted or unweighted.

Matrices Associated with a Graph

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

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

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

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

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

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

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

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

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

Matrices Associated with a Graph

- Let $D = D(G) := \mathrm{diag}(d_1, \dots, d_n)$ 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.

Why Graph Laplacian Eigenfunctions?

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

Why Graph Laplacian Eigenfunctions?

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

Why Graph Laplacian Eigenfunctions?

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

Why Graph Laplacian Eigenfunctions?

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

Why Graph Laplacian Eigenfunctions?

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

Why Graph Laplacian Eigenfunctions?

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

A Simple Yet Important Example: A Path Graph

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

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

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

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 $\boldsymbol{\phi}_0 = \mathbf{1}_n$.
- 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 $\boldsymbol{\phi}_0 = \mathbf{1}_n$.
- 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 $\boldsymbol{\phi}_0 = \mathbf{1}_n$.
- 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 $\boldsymbol{\phi}_0 = \mathbf{1}_n$.
- 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 $\boldsymbol{\phi}_0 = \mathbf{1}_n$.
- 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 $\boldsymbol{\phi}_0 = \mathbf{1}_n$.
- This led M. Fiedler (1973) to define the algebraic connectivity of G by $a(G) := \lambda_1(G)$, viewing it as *a quantitative measure of connectivity*.

Outline

Graph Partitioning via Spectral Clustering

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

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

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

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

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

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

---

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

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

Graph Partitioning via Spectral Clustering

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

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

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

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

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

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

---

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

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

Graph Partitioning via Spectral Clustering

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

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

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

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

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

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

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

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

Graph Partitioning via Spectral Clustering

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

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

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

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

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

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

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

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

Graph Partitioning via Spectral Clustering

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

Definition (Weak Nodal Domain)

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

Corollary (Fiedler (1975))

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

Graph Partitioning via Spectral Clustering

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

Definition (Weak Nodal Domain)

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

Corollary (Fiedler (1975))

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

Graph Partitioning via Spectral Clustering

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

Definition (Weak Nodal Domain)

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

Corollary (Fiedler (1975))

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

Example of Graph Partitioning

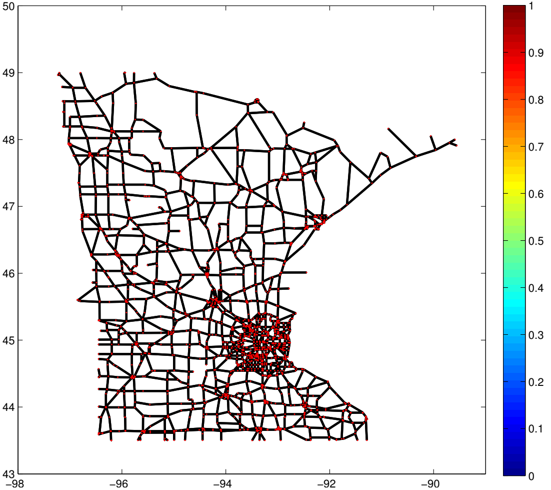


Figure: The MN road network

Example of Graph Partitioning

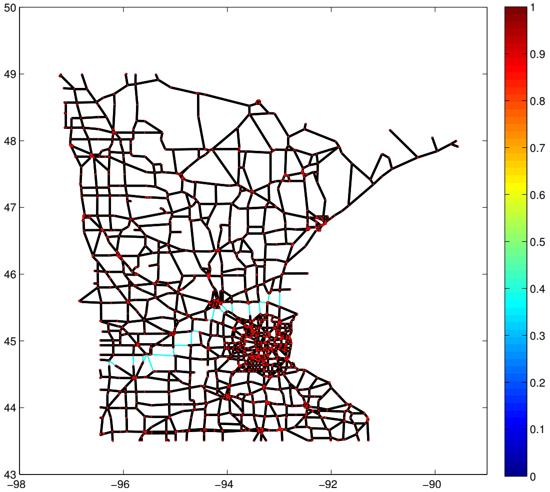
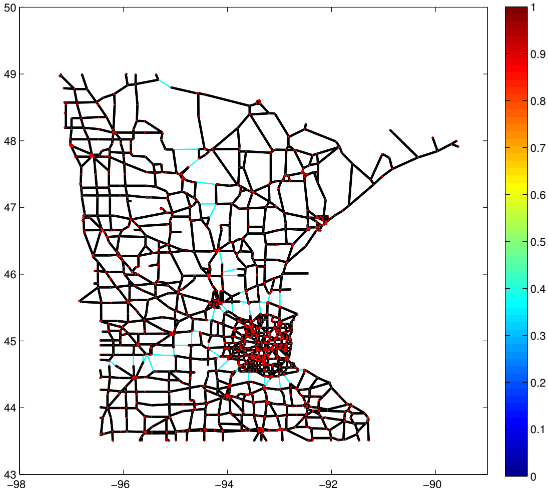


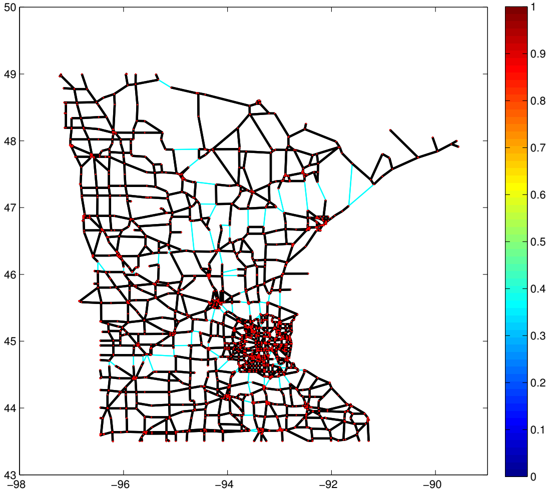
Figure: The MN road network partitioned via the Fiedler vector of L_{rw}

One Can Do This Recursively!



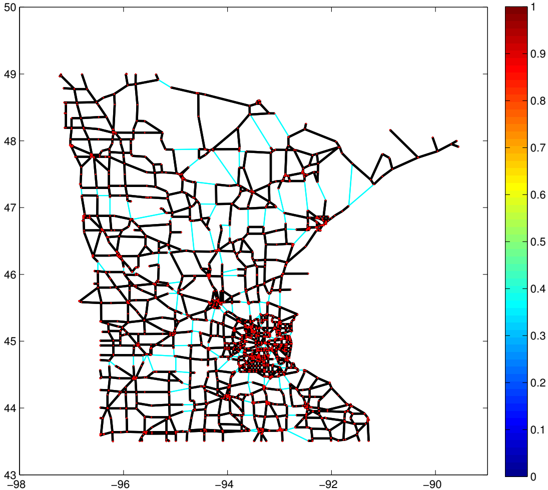
The MN road network recursively partitioned via the Fiedler vectors of L_{rw}'s of subgraphs: $j = 2$

One Can Do This Recursively!



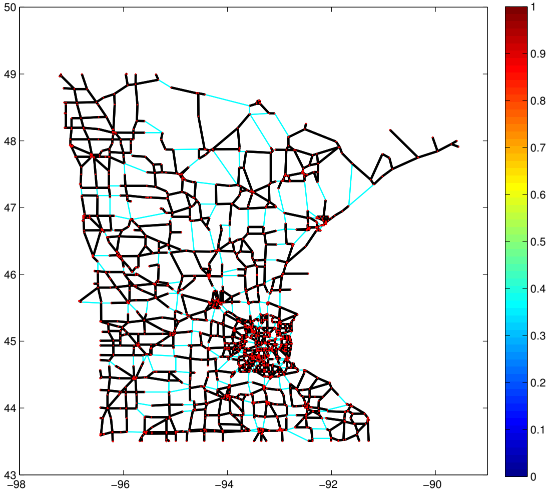
$j = 3$

One Can Do This Recursively!



$j = 4$

One Can Do This Recursively!



$j = 5$

Outline

Motivation: Building Multiscale Basis Dictionaries

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

Key Difficulties to Build Wavelets/Wavelet Packets on Graphs

- Using graph Laplacian eigenvectors as "cosines" or Fourier modes on graphs with eigenvalues as (the square of) their "frequencies" has been popular.
- However, the notion of *frequency* is ill-defined on general graphs and the Fourier transform is not properly defined on graphs
- Graph Laplacian eigenvectors may also exhibit peculiar behaviors depending on *topology* and *structure* of given graphs!
- For example, eigenvectors corresponding to high eigenvalues may be highly *localized*; see: Y. Nakatsukasa, N. Saito, & E. Woei: "Mysteries around graph Laplacian eigenvalue 4," *Linear Algebra and its Applications*, vol. 438, no. 8, pp. 3231–3246, 2013.
- Hence, building wavelets on graphs based on *the Littlewood-Paley theory* is quite challenging
- Moreover, the notion of *smoothness class* of functions (e.g., Sobolev and Besov spaces) is also difficult to define on graphs $\Longrightarrow$ *Spaces of homogeneous type* (e.g., Deng & Han, 2009); the LP theory on more abstract setting (e.g., Mhaskar & Prestin, 2004) ?

Key Difficulties to Build Wavelets/Wavelet Packets on Graphs

- Using graph Laplacian eigenvectors as "cosines" or Fourier modes on graphs with eigenvalues as (the square of) their "frequencies" has been popular.
- However, the notion of *frequency* is ill-defined on general graphs and the Fourier transform is not properly defined on graphs
- Graph Laplacian eigenvectors may also exhibit peculiar behaviors depending on *topology* and *structure* of given graphs!
- For example, eigenvectors corresponding to high eigenvalues may be highly *localized*; see: Y. Nakatsukasa, N. Saito, & E. Woei: "Mysteries around graph Laplacian eigenvalue 4," *Linear Algebra and its Applications*, vol. 438, no. 8, pp. 3231–3246, 2013.
- Hence, building wavelets on graphs based on *the Littlewood-Paley theory* is quite challenging
- Moreover, the notion of *smoothness class* of functions (e.g., Sobolev and Besov spaces) is also difficult to define on graphs $\Longrightarrow$ *Spaces of homogeneous type* (e.g., Deng & Han, 2009); the LP theory on more abstract setting (e.g., Mhaskar & Prestin, 2004) ?

Key Difficulties to Build Wavelets/Wavelet Packets on Graphs

- Using graph Laplacian eigenvectors as "cosines" or Fourier modes on graphs with eigenvalues as (the square of) their "frequencies" has been popular.
- However, the notion of *frequency* is ill-defined on general graphs and the Fourier transform is not properly defined on graphs
- Graph Laplacian eigenvectors may also exhibit peculiar behaviors depending on *topology* and *structure* of given graphs!
- For example, eigenvectors corresponding to high eigenvalues may be highly *localized*; see: Y. Nakatsukasa, N. Saito, & E. Woei: "Mysteries around graph Laplacian eigenvalue 4," *Linear Algebra and its Applications*, vol. 438, no. 8, pp. 3231–3246, 2013.
- Hence, building wavelets on graphs based on *the Littlewood-Paley theory* is quite challenging
- Moreover, the notion of *smoothness class* of functions (e.g., Sobolev and Besov spaces) is also difficult to define on graphs $\Longrightarrow$ *Spaces of homogeneous type* (e.g., Deng & Han, 2009); the LP theory on more abstract setting (e.g., Mhaskar & Prestin, 2004) ?

Key Difficulties to Build Wavelets/Wavelet Packets on Graphs

- Using graph Laplacian eigenvectors as "cosines" or Fourier modes on graphs with eigenvalues as (the square of) their "frequencies" has been popular.
- However, the notion of *frequency* is ill-defined on general graphs and the Fourier transform is not properly defined on graphs
- Graph Laplacian eigenvectors may also exhibit peculiar behaviors depending on *topology* and *structure* of given graphs!
- For example, eigenvectors corresponding to high eigenvalues may be highly *localized*; see: Y. Nakatsukasa, N. Saito, & E. Woei: "Mysteries around graph Laplacian eigenvalue 4," *Linear Algebra and its Applications*, vol. 438, no. 8, pp. 3231–3246, 2013.
- Hence, building wavelets on graphs based on *the Littlewood-Paley theory* is quite challenging
- Moreover, the notion of *smoothness class* of functions (e.g., Sobolev and Besov spaces) is also difficult to define on graphs $\Longrightarrow$ *Spaces of homogeneous type* (e.g., Deng & Han, 2009); the LP theory on more abstract setting (e.g., Mhaskar & Prestin, 2004) ?

Key Difficulties to Build Wavelets/Wavelet Packets on Graphs

- Using graph Laplacian eigenvectors as "cosines" or Fourier modes on graphs with eigenvalues as (the square of) their "frequencies" has been popular.
- However, the notion of *frequency* is ill-defined on general graphs and the Fourier transform is not properly defined on graphs
- Graph Laplacian eigenvectors may also exhibit peculiar behaviors depending on *topology* and *structure* of given graphs!
- For example, eigenvectors corresponding to high eigenvalues may be highly *localized*; see: Y. Nakatsukasa, N. Saito, & E. Woei: "Mysteries around graph Laplacian eigenvalue 4," *Linear Algebra and its Applications*, vol. 438, no. 8, pp. 3231–3246, 2013.
- Hence, building wavelets on graphs based on *the Littlewood-Paley theory* is quite challenging
- Moreover, the notion of *smoothness class* of functions (e.g., Sobolev and Besov spaces) is also difficult to define on graphs $\Longrightarrow$ *Spaces of homogeneous type* (e.g., Deng & Han, 2009); the LP theory on more abstract setting (e.g., Mhaskar & Prestin, 2004) ?

Key Difficulties to Build Wavelets/Wavelet Packets on Graphs

- Using graph Laplacian eigenvectors as "cosines" or Fourier modes on graphs with eigenvalues as (the square of) their "frequencies" has been popular.
- However, the notion of *frequency* is ill-defined on general graphs and the Fourier transform is not properly defined on graphs
- Graph Laplacian eigenvectors may also exhibit peculiar behaviors depending on *topology* and *structure* of given graphs!
- For example, eigenvectors corresponding to high eigenvalues may be highly *localized*; see: Y. Nakatsukasa, N. Saito, & E. Woei: "Mysteries around graph Laplacian eigenvalue 4," *Linear Algebra and its Applications*, vol. 438, no. 8, pp. 3231–3246, 2013.
- Hence, building wavelets on graphs based on *the Littlewood-Paley theory* is quite challenging
- Moreover, the notion of *smoothness class* of functions (e.g., Sobolev and Besov spaces) is also difficult to define on graphs $\Longrightarrow$ *Spaces of homogeneous type* (e.g., Deng & Han, 2009); the LP theory on more abstract setting (e.g., Mhaskar & Prestin, 2004) ?

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

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

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

Outline

Hierarchical Graph Laplacian Eigen Transform (HGLET)

Now we present a 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 $\boldsymbol{\phi}^j_{k,1}$ (recall that it naturally *bipartitions* the graph)
3. Generate an orthonormal basis for each of the partitions ⇒ Laplacian eigenvectors
4. Repeat...

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

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 $\boldsymbol{\phi}^{j}_{k,1}$ (recall that it naturally *bipartitions* the graph)
3. Generate an orthonormal basis for each of the partitions ⇒ Laplacian eigenvectors
4. Repeat...

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

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 $\boldsymbol{\phi}^j_{k,1}$ (recall that it naturally *bipartitions* the graph)
3. Generate an orthonormal basis for each of the partitions ⇒ Laplacian eigenvectors
4. Repeat...

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

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

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 $\boldsymbol{\phi}^{j}_{k,1}$ (recall that it naturally *bipartitions* the graph)
3. Generate an orthonormal basis for each of the partitions ⇒ Laplacian eigenvectors
4. Repeat...

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

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

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

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

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

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

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

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

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

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 $\boldsymbol{\phi}^{j}_{k,1}$ (recall that it naturally *bipartitions* the graph)
3. Generate an orthonormal basis for each of the partitions ⇒ Laplacian eigenvectors
4. Repeat...

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

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

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

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

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

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

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

$$\vdots$$

Remarks

- For an unweighted path graph, this exactly yields a *dictionary of the multiscale block DCTs*.
- A union of bases on disjoint subsets is obviously orthonormal.
- Similar to wavelet packet or local cosine dictionaries in that it generates a dictionary of bases (i.e., an *overcomplete system*) from which we can select a particular basis useful for the task at hand ⇒ best-basis algorithm, local discriminant basis algorithm, ...
- One can use any graph bipartition method other than the one based on the Fiedler vectors to construct the HGLET dictionary; ∃ many recent graph partitioning methods, e.g., *diffuse interface model* of Bertozzi & Flenner, ...

Remarks

- For an unweighted path graph, this exactly yields a *dictionary of the multiscale block DCTs*.
- A union of bases on disjoint subsets is obviously orthonormal.
- Similar to wavelet packet or local cosine dictionaries in that it generates a dictionary of bases (i.e., an *overcomplete system*) from which we can select a particular basis useful for the task at hand ⇒ best-basis algorithm, local discriminant basis algorithm, ...
- One can use any graph bipartition method other than the one based on the Fiedler vectors to construct the HGLET dictionary; ∃ many recent graph partitioning methods, e.g., *diffuse interface model* of Bertozzi & Flenner, ...

Remarks

- For an unweighted path graph, this exactly yields a *dictionary of the multiscale block DCTs*.
- A union of bases on disjoint subsets is obviously orthonormal.
- Similar to wavelet packet or local cosine dictionaries in that it generates a dictionary of bases (i.e., an *overcomplete system*) from which we can select a particular basis useful for the task at hand ⇒ best-basis algorithm, local discriminant basis algorithm, . . .
- One can use any graph bipartition method other than the one based on the Fiedler vectors to construct the HGLET dictionary; ∃ many recent graph partitioning methods, e.g., *diffuse interface model* of Bertozzi & Flenner, . . .

Remarks

- For an unweighted path graph, this exactly yields a *dictionary of the multiscale block DCTs*.
- A union of bases on disjoint subsets is obviously orthonormal.
- Similar to wavelet packet or local cosine dictionaries in that it generates a dictionary of bases (i.e., an *overcomplete system*) from which we can select a particular basis useful for the task at hand $\Rightarrow$ best-basis algorithm, local discriminant basis algorithm, ...
- One can use any graph bipartition method other than the one based on the Fiedler vectors to construct the HGLET dictionary; $\exists$ many recent graph partitioning methods, e.g., *diffuse interface model* of Bertozzi & Flenner, ...

Outline

Generalized Haar-Walsh Transform (GHWT)

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

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

The algorithm can be summarized as follows...

Generalized Haar-Walsh Transform (GHWT)

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

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

The algorithm can be summarized as follows...

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

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

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

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

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

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

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

$$\begin{bmatrix} \boldsymbol{\psi}^0_{0,0} & \boldsymbol{\psi}^0_{0,1} & \boldsymbol{\psi}^0_{0,2} & \boldsymbol{\psi}^0_{0,3} & \cdots & \boldsymbol{\psi}^0_{0,n-2} & \boldsymbol{\psi}^0_{0,n-1} \end{bmatrix}$$

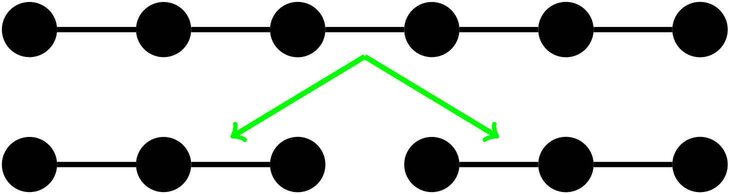
$$\vdots$$

$$\begin{bmatrix} \boldsymbol{\psi}^{j_{\max}-1}_{0,0} & \boldsymbol{\psi}^{j_{\max}-1}_{0,1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\psi}^{j_{\max}-1}_{1,0} & \boldsymbol{\psi}^{j_{\max}-1}_{1,1} \end{bmatrix} \cdots \begin{bmatrix} \boldsymbol{\psi}^{j_{\max}-1}_{K^{j_{\max}-1}-1,0} & \boldsymbol{\psi}^{j_{\max}-1}_{K^{j_{\max}-1}-1,1} \end{bmatrix}$$

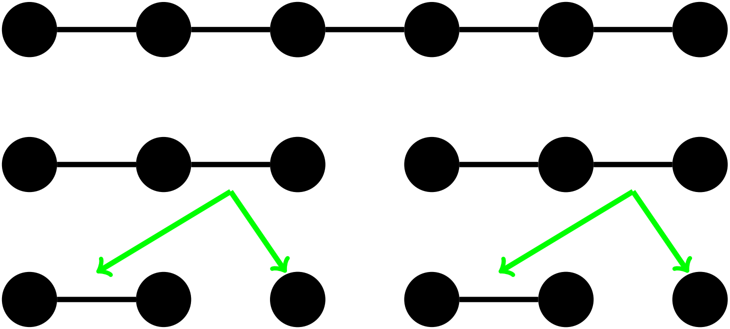
$$\begin{bmatrix} \boldsymbol{\psi}^{j_{\max}}_{0,0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\psi}^{j_{\max}}_{1,0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\psi}^{j_{\max}}_{2,0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\psi}^{j_{\max}}_{3,0} \end{bmatrix} \cdots \begin{bmatrix} \boldsymbol{\psi}^{j_{\max}}_{K^{j_{\max}}-2,0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\psi}^{j_{\max}}_{K^{j_{\max}}-1,0} \end{bmatrix}$$

GHWT on P_6

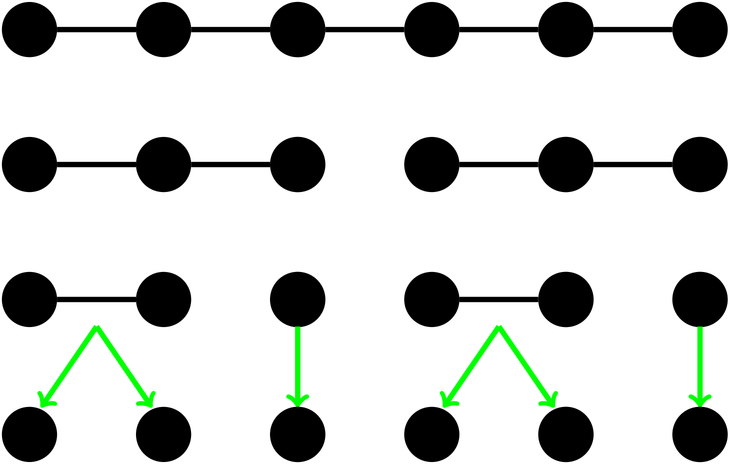
GHWT on P_6



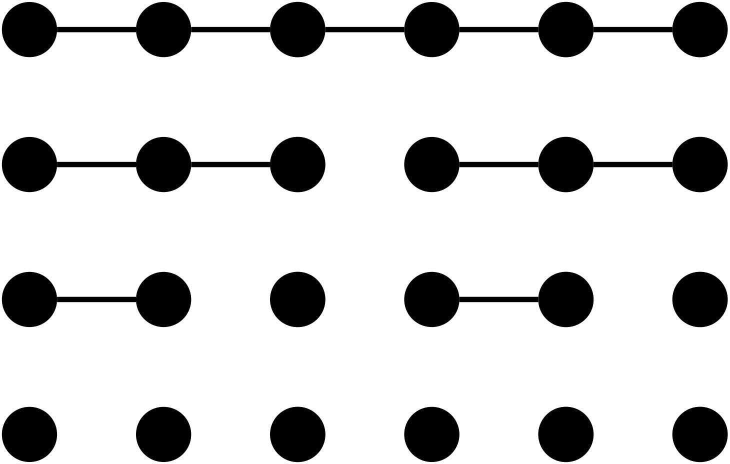
GHWT on P_6



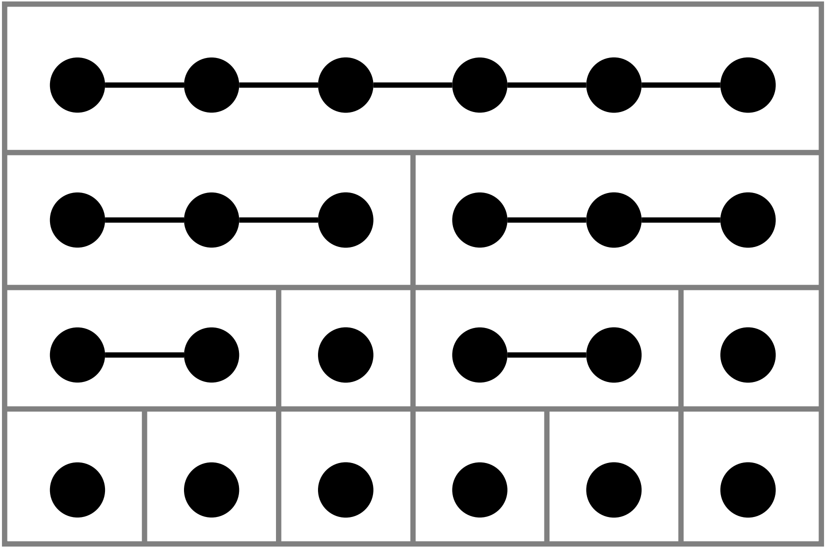
GHWT on P_6



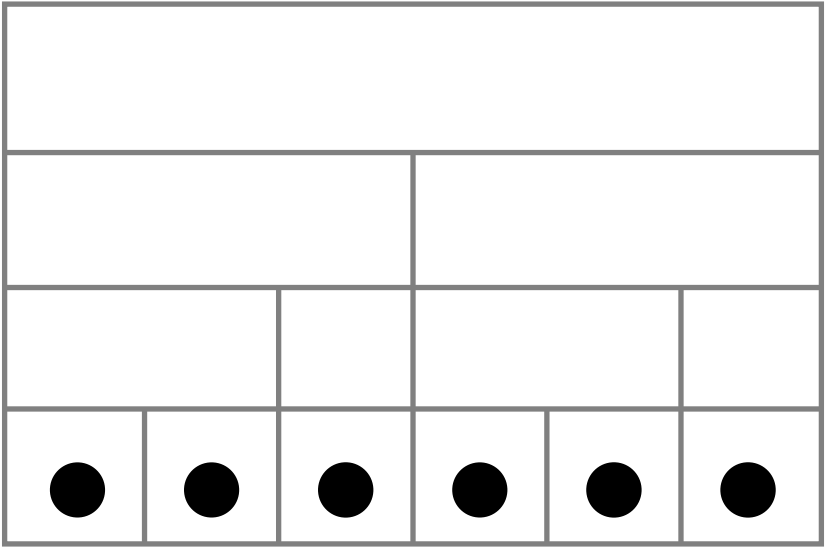
GHWT on P_6



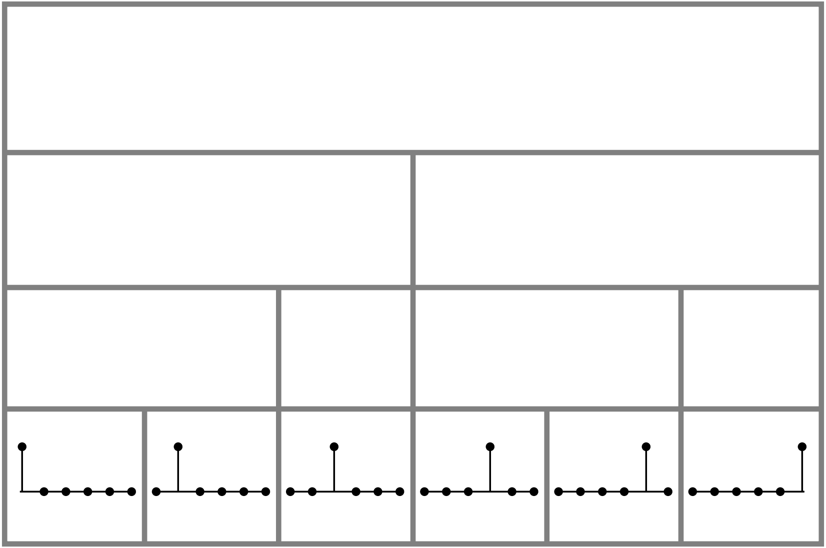
GHWT on P_6



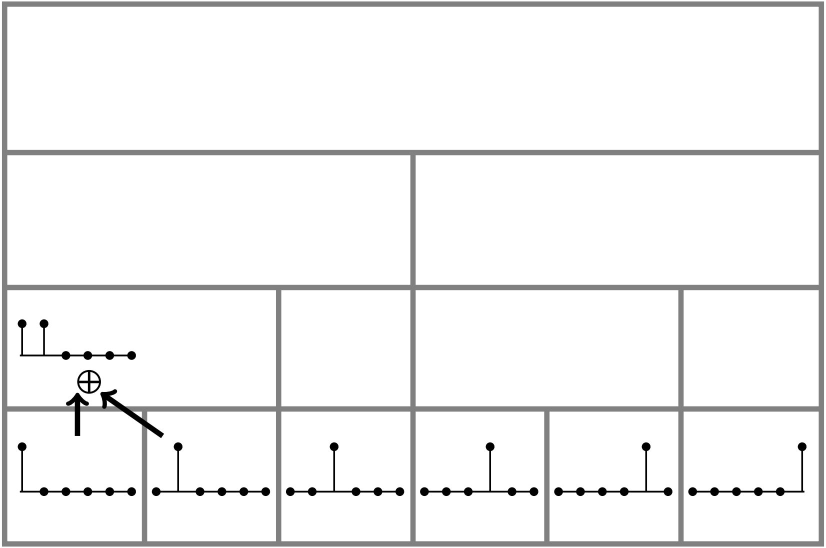
GHWT on P_6



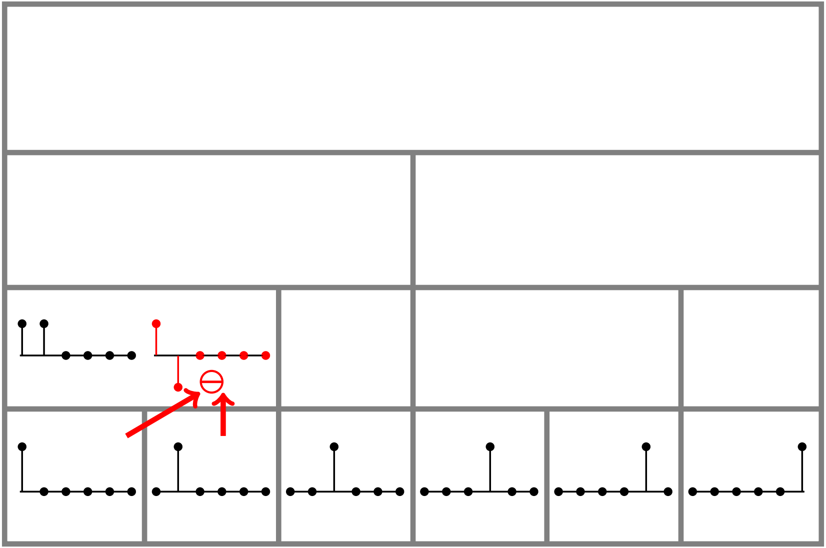
GHWT on P_6



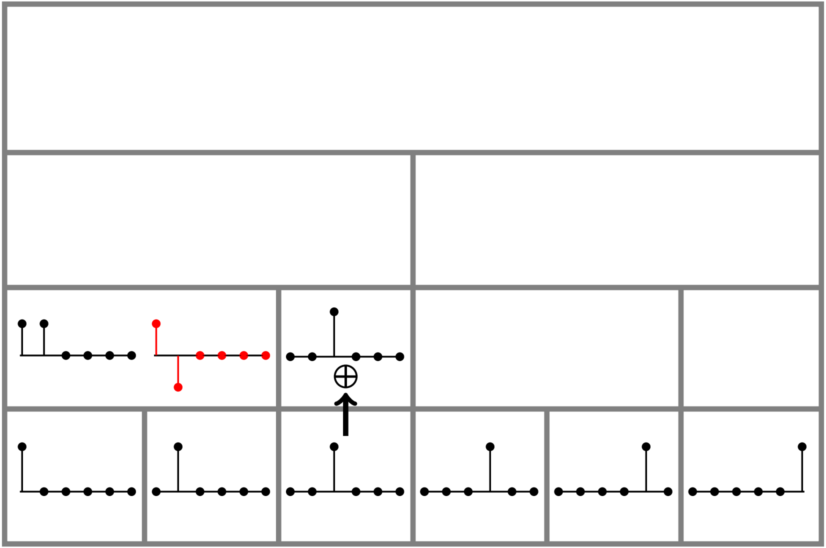
GHWT on P_6



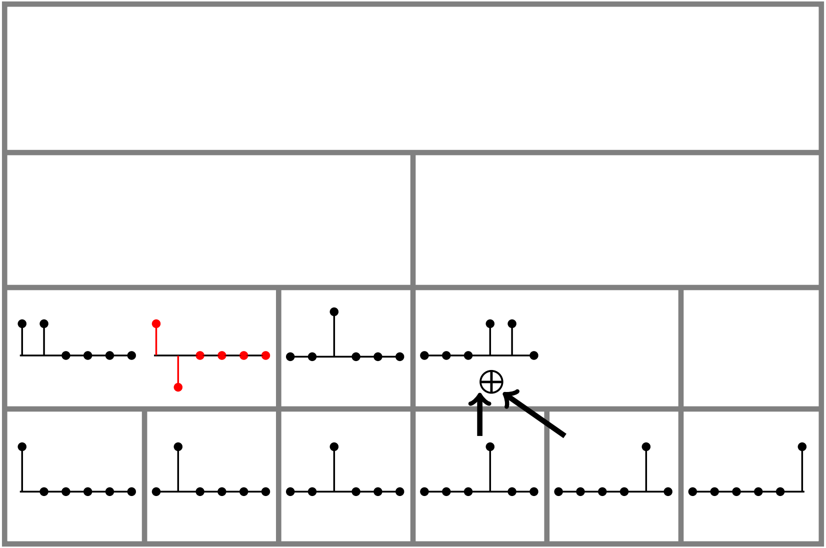
GHWT on P_6



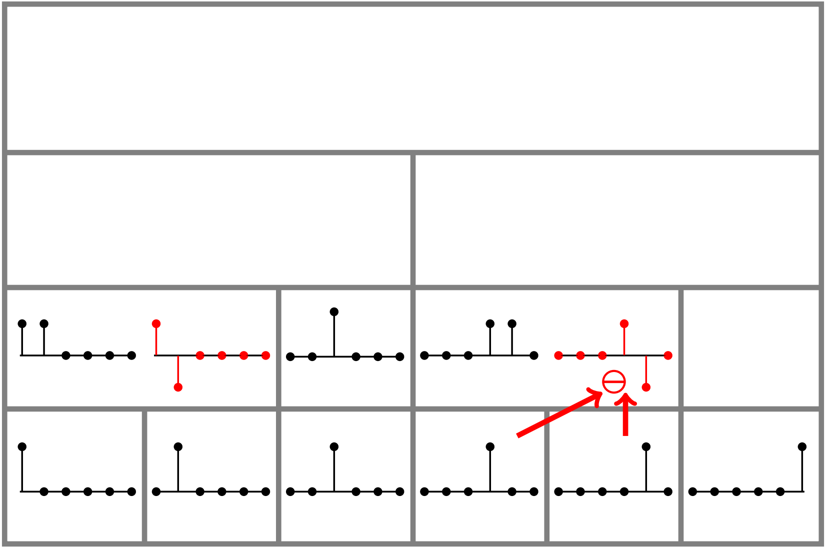
GHWT on P_6



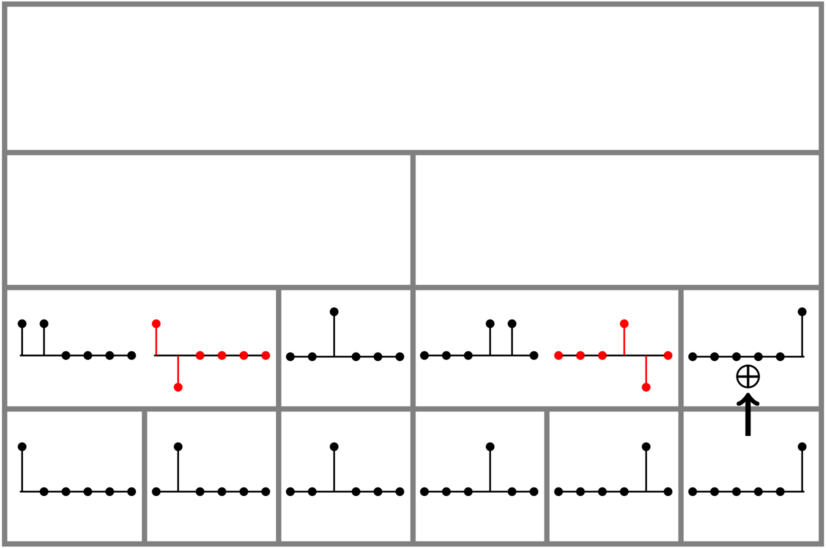
GHWT on P_6



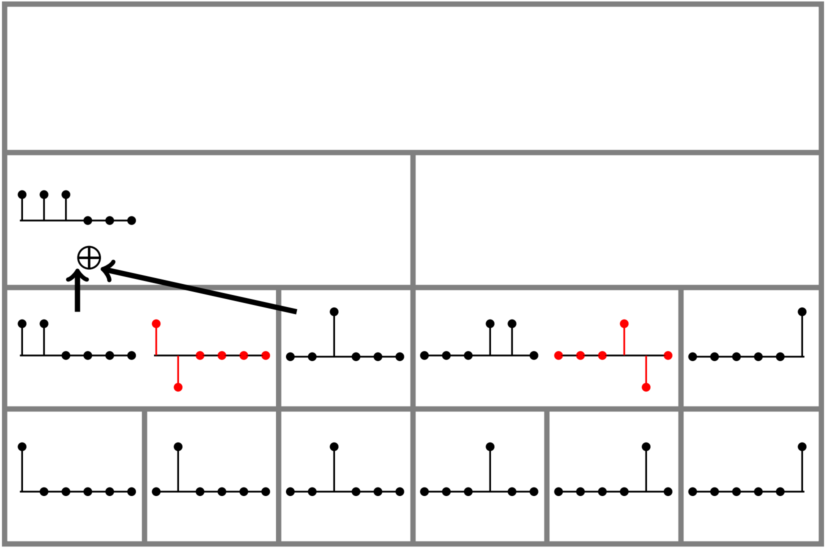
GHWT on P_6



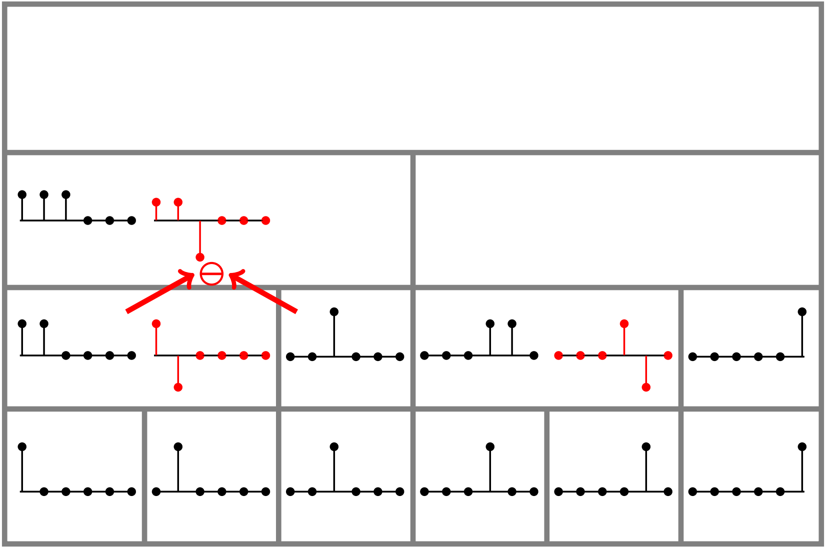
GHWT on P_6



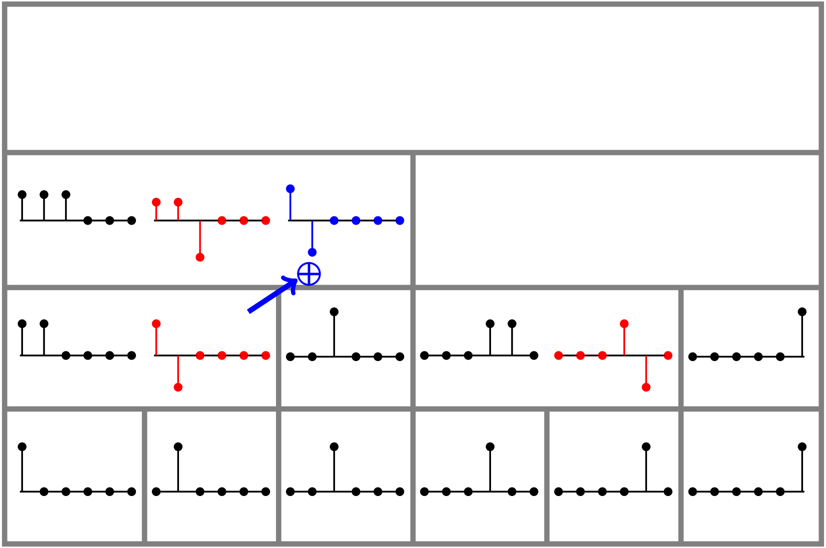
GHWT on P_6



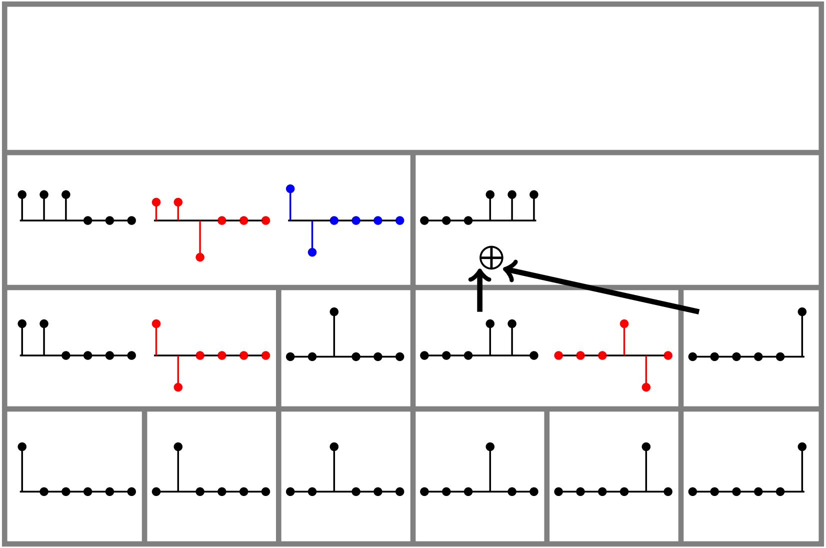
GHWT on P_6



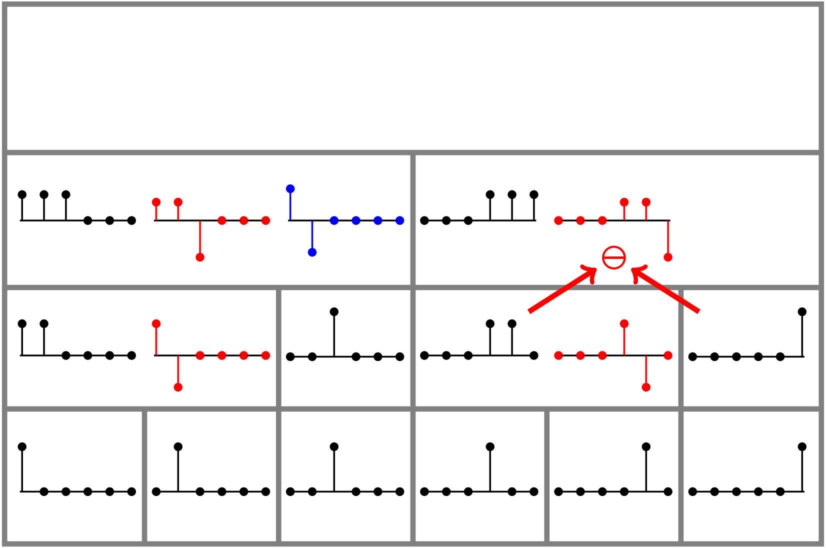
GHWT on P_6



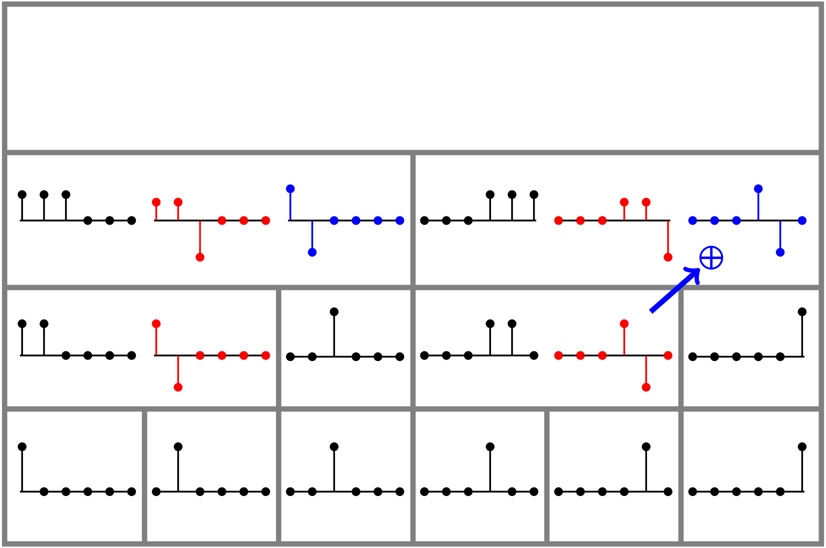
GHWT on P_6



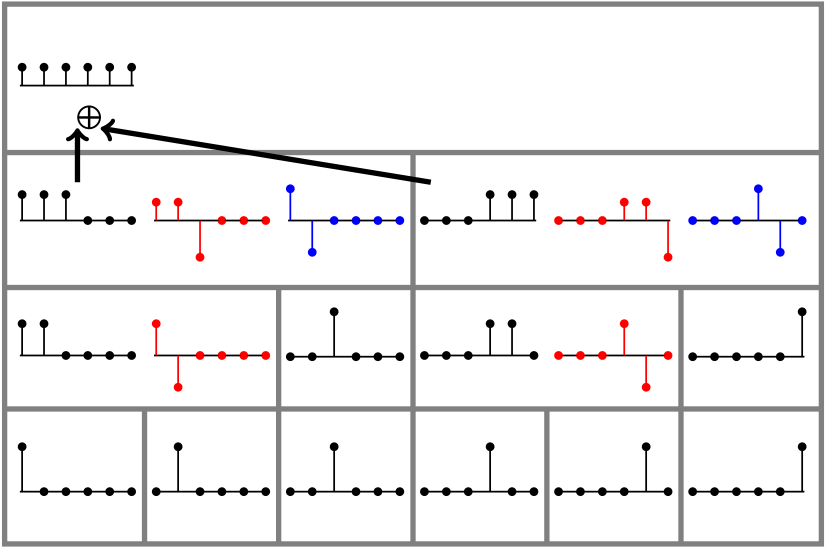
GHWT on P_6



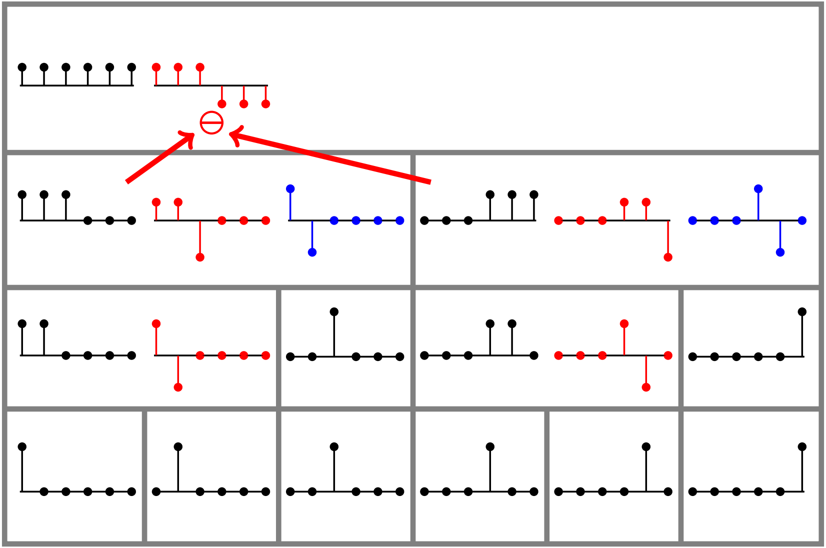
GHWT on P_6



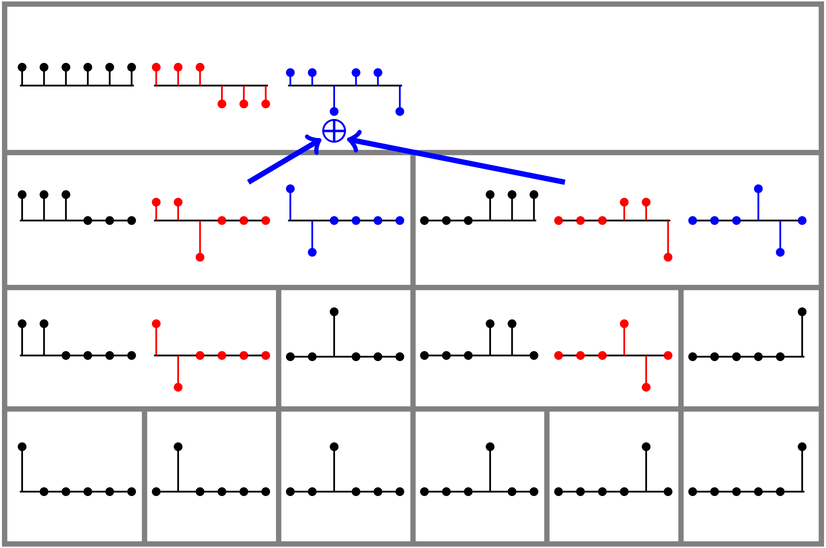
GHWT on P_6



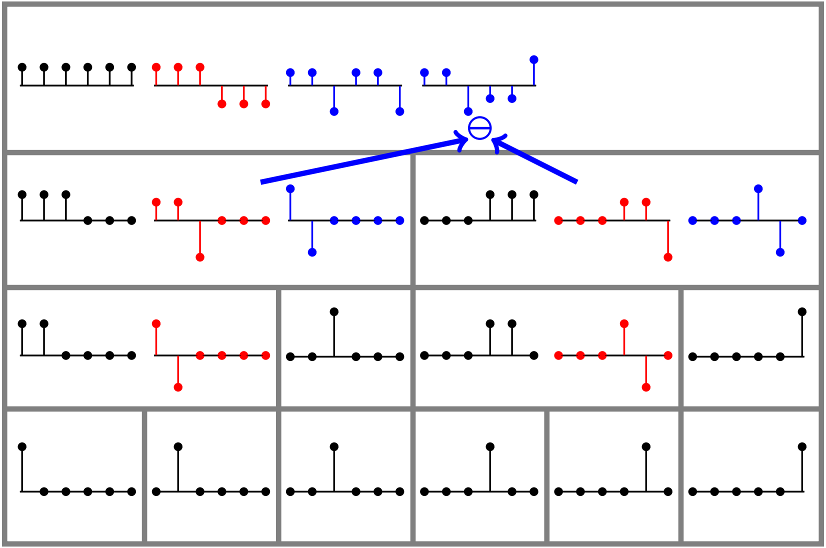
GHWT on P_6



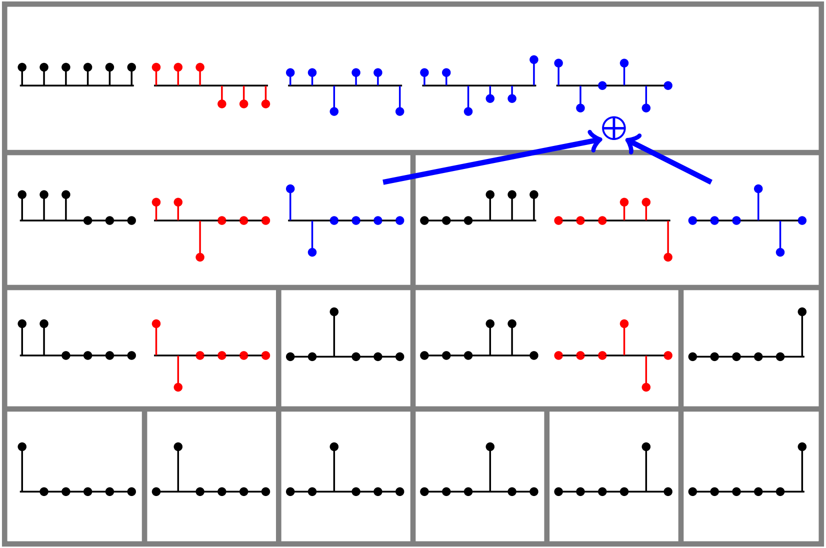
GHWT on P_6



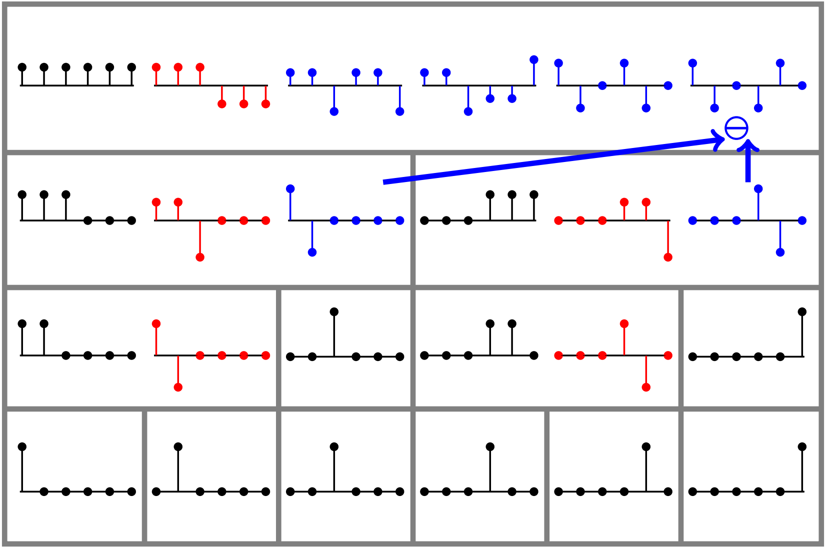
GHWT on P_6



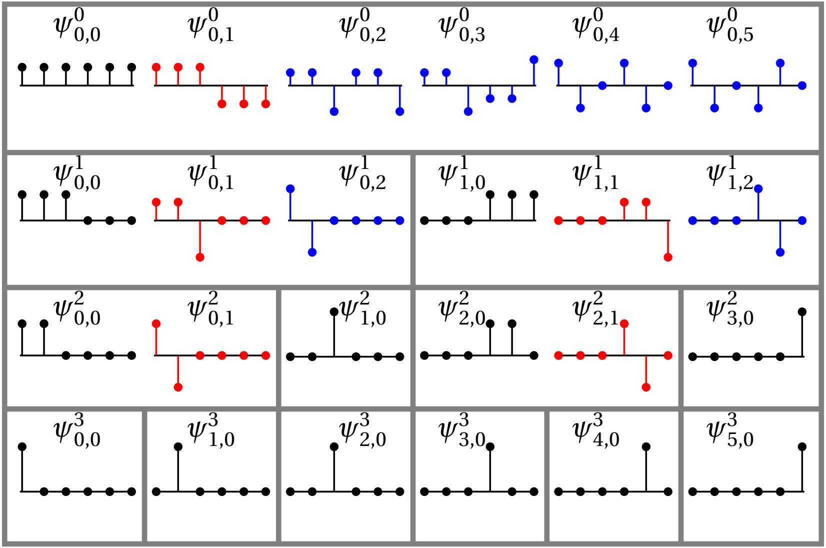
GHWT on P_6



GHWT on P_6



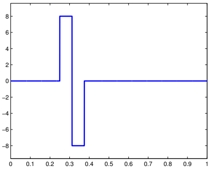
GHWT on P_6


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

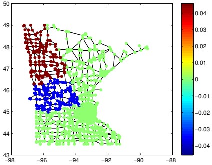
Basis Vector & Coefficient Notation

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

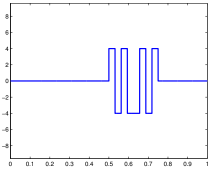
- $\ell = 0 \Rightarrow$ **scaling coefficient/basis vector**
- $\ell = 1 \Rightarrow$ **Haar-like coefficient/basis vector**
- $\ell \geq 2 \Rightarrow$ **Walsh-like coefficient/basis vector**



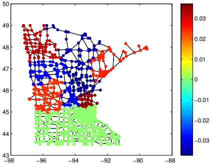
(a) Haar function on $\mathbb{R}$



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



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



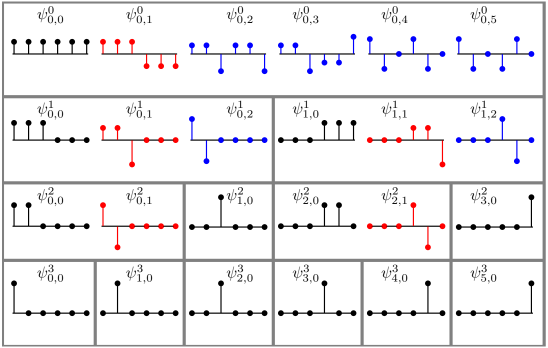
(d) Walsh-like vector $\boldsymbol{\psi}^1_{0,5}$

Remarks

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

Remarks

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



Remarks . . .

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

Remarks . . .

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

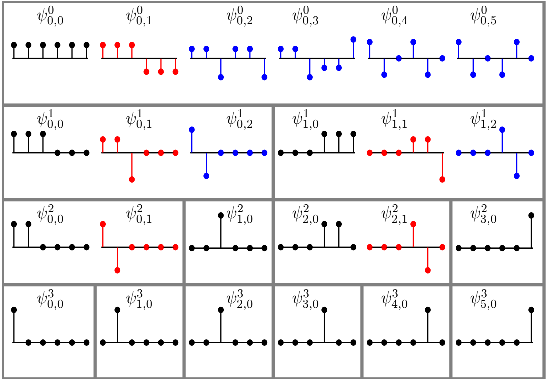


Figure: Default dictionary; i.e., coarse-to-fine

Remarks . . .

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

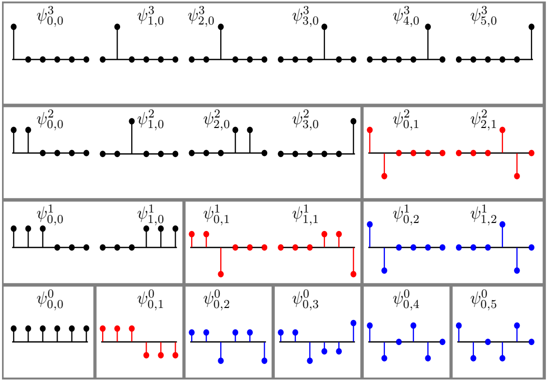


Figure: Reordered & regrouped dictionary; i.e., fine-to-coarse

Remarks . . .

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

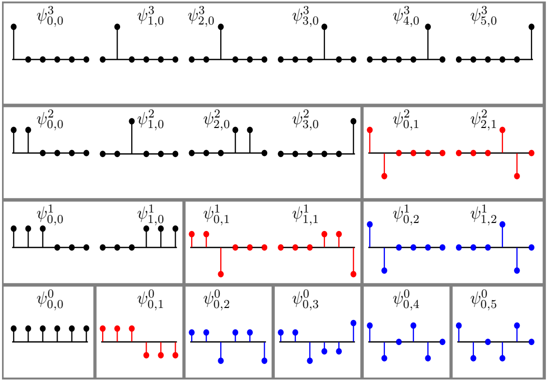


Figure: Reordered & regrouped dictionary; i.e., fine-to-coarse

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

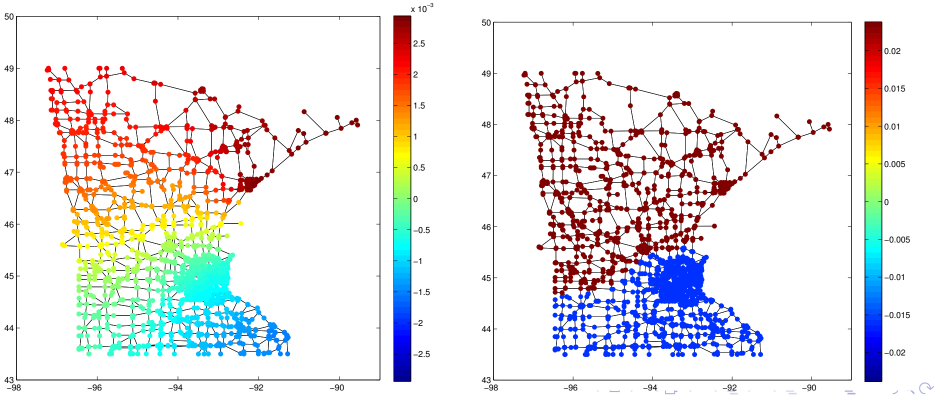
HGLET vs. GHWT

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

HGLET vs. GHWT

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

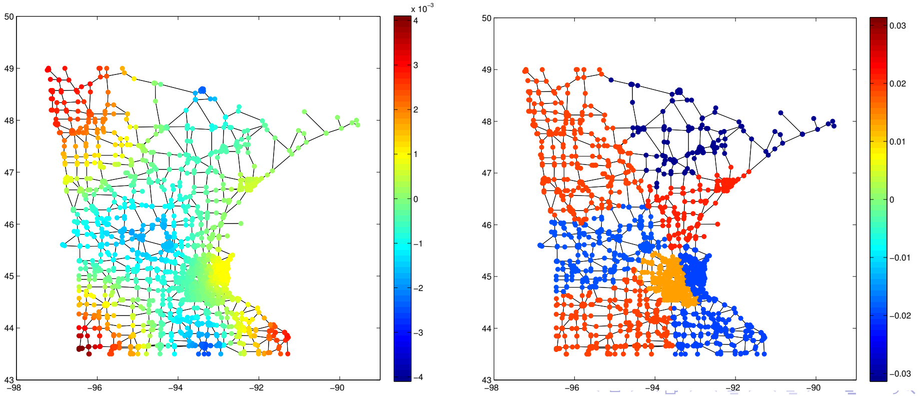
Level $j = 0$, Region $k = 0$, $l = 1$



HGLET vs. GHWT

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

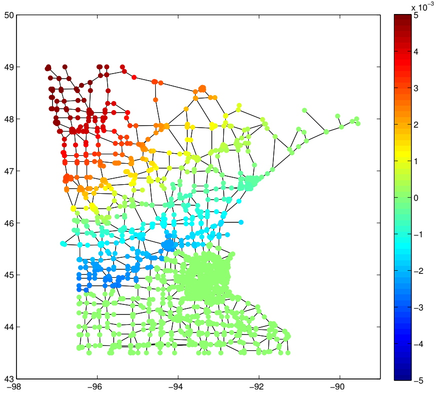
Level $j = 0$, Region $k = 0$, $l = 7$

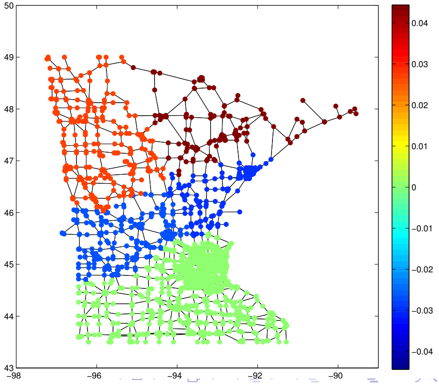


HGLET vs. GHWT

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

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

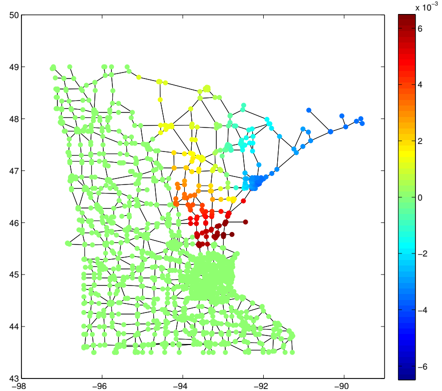


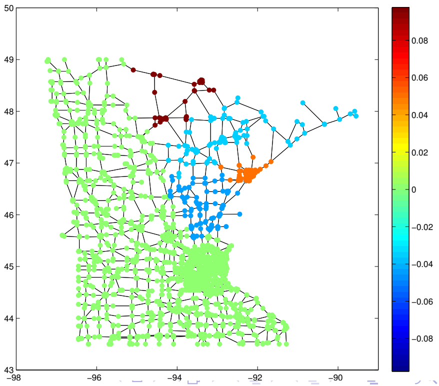


HGLET vs. GHWT

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

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

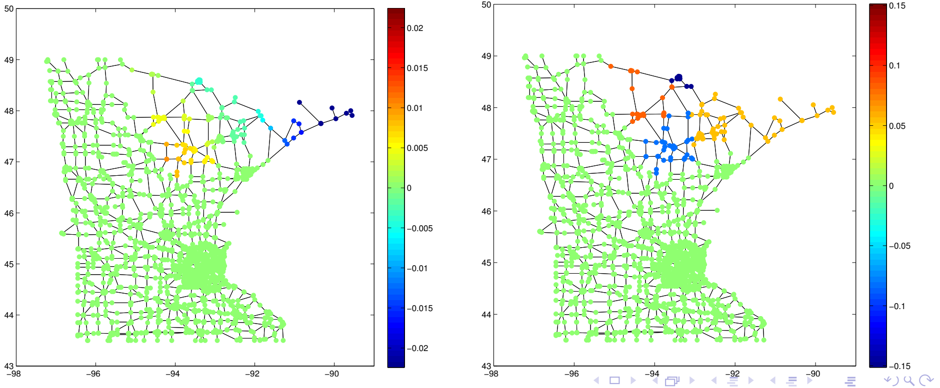




HGLET vs. GHWT

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

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



Computational Complexity: HGLET vs. GHWT

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

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

Related Work

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

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

Outline

Best-Basis Algorithms for HGLET & GHWT

- Coifman and Wickerhauser (1992) developed the best-basis algorithm as a means of selecting the basis from a dictionary of wavelet packets that is "best" for approximation/compression.
- We generalize this approach, developing and implementing an algorithm for selecting the basis from the dictionary of HGLET / GHWT bases that is "best" for approximation and compression.
- We require an appropriate cost functional $\mathscr{J}$. For example:

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

- Another example cost functional is based on the *Minimum Description Length (MDL)*.

The Minimum Description Length (MDL) Criterion

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

⇒ Need to specify the model class for a given signal

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$c^0_{0,0} \quad c^0_{0,1} \quad c^0_{0,2} \quad \cdots \quad c^0_{0,n^0_0-1}$$

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

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

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

$$c^2_{2,0} \quad c^2_{2,1} \quad \cdots \quad c^2_{2,n^2_2-1} \qquad c^2_{3,0} \quad c^2_{3,1} \quad \cdots \quad c^2_{3,n^2_3-1}$$

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

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

$$c^0_{0,0} \quad c^0_{0,1} \quad c^0_{0,2} \quad \cdots \quad c^0_{0,n^0_0-1}$$

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

$$c^1_{0,0} \quad c^1_{0,1} \quad c^1_{0,2} \quad \cdots \quad c^1_{0,n^1_0-1}$$

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

$$c^2_{2,0}\;c^2_{2,1}\cdots c^2_{2,n^2_2-1} \quad c^2_{3,0}\;c^2_{3,1}\cdots c^2_{3,n^2_3-1}$$

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

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

$$c^0_{0,0} \quad c^0_{0,1} \quad c^0_{0,2} \quad \cdots \quad c^0_{0,n^0_0-1}$$

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

$$c^1_{0,0} \quad c^1_{0,1} \quad c^1_{0,2} \quad \cdots \quad c^1_{0,n^1_0-1}$$

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

$$c^2_{2,0} \quad c^2_{2,1} \cdots c^2_{2,n^2_2-1} \qquad c^2_{3,0} \quad c^2_{3,1} \cdots c^2_{3,n^2_3-1}$$

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

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

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

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

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

$$c^1_{0,0} \quad c^1_{0,1} \quad c^1_{0,2} \quad \cdots \quad c^1_{0,n^1_0-1}$$

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

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

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

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

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

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

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

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

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

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

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

Outline

Motivation

There are many examples of data in matrix format:

- Images
- Ratings/Reviews
  - Rows → Netflix users
  - Columns → movies
  - A_{ij} → user i's rating of movie j on a 1-5 scale
- Spatiotemporal data
  - Rows → sensors
  - Columns → times
  - A_{ij} → sensor i's temperature reading at time j

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

Method

1. Use the matrix data to recursively partition the rows and the columns (explained on next slide)
2. Use the GHWT and best-basis algorithm to analyze the matrix
   - Analyze along the rows and extract the best basis
   - Analyze the row best basis coefficients along the columns and extract the best basis
3. Analyze the expansion coefficients for a variety of tasks, e.g., compression, classification, regression, etc.

Method

1. Use the matrix data to recursively partition the rows and the columns (explained on next slide)
2. Use the GHWT and best-basis algorithm to analyze the matrix
   - Analyze along the rows and extract the best basis
   - Analyze the row best basis coefficients along the columns and extract the best basis
3. Analyze the expansion coefficients for a variety of tasks, e.g., compression, classification, regression, etc.

Method

1. Use the matrix data to recursively partition the rows and the columns (explained on next slide)
2. Use the GHWT and best-basis algorithm to analyze the matrix
   - Analyze along the rows and extract the best basis
   - Analyze the row best basis coefficients along the columns and extract the best basis
3. Analyze the expansion coefficients for a variety of tasks, e.g., compression, classification, regression, etc.

Method

1. Use the matrix data to recursively partition the rows and the columns (explained on next slide)
2. Use the GHWT and best-basis algorithm to analyze the matrix
   - Analyze along the rows and extract the best basis
   - Analyze the row best basis coefficients along the columns and extract the best basis
3. Analyze the expansion coefficients for a variety of tasks, e.g., compression, classification, regression, etc.

Method

1. Use the matrix data to recursively partition the rows and the columns (explained on next slide)
2. Use the GHWT and best-basis algorithm to analyze the matrix
   - Analyze along the rows and extract the best basis
   - Analyze the row best basis coefficients along the columns and extract the best basis
3. Analyze the expansion coefficients for a variety of tasks, e.g., compression, classification, regression, etc.

Matrix Partitioning à la Dhillon (2001)[1]

- Given a matrix $A \in \mathbb{R}^{N_R \times N_C}$, the rows and columns are viewed as the two sets of nodes in a *bipartite* graph.
- A_{ij} denotes the weight between the node for row i and the node for column j (If A is a term-document matrix, then A_{ij} is a relative frequency of occurrence of term i in the document j).
- Then, matrices associated with this bipartite graph can be written as:

$$W = \begin{bmatrix} O & A \\ A^{\mathsf{T}} & O \end{bmatrix}$$

$$D = \begin{bmatrix} D_R & O \\ O & D_C \end{bmatrix} \qquad D_R := \mathrm{diag}(A\mathbf{1}); D_C := \mathrm{diag}(A^{\mathsf{T}}\mathbf{1})$$

$$L = D - W = \begin{bmatrix} D_R & -A \\ -A^{\mathsf{T}} & D_C \end{bmatrix}$$

[1] I. S. Dhillon: Co-clustering documents and words using Bipartite Spectral Graph Partitioning, *Proc. 7th ACM SIGKDD*, pp. 269–274, 2001.

Matrix Partitioning à la Dhillon (2001)[1]

- Given a matrix $A \in \mathbb{R}^{N_R \times N_C}$, the rows and columns are viewed as the two sets of nodes in a *bipartite* graph.
- A_{ij} denotes the weight between the node for row i and the node for column j (If A is a term-document matrix, then A_{ij} is a relative frequency of occurrence of term i in the document j).
- Then, matrices associated with this bipartite graph can be written as:

$$W = \begin{bmatrix} O & A \\ A^{\mathsf{T}} & O \end{bmatrix}$$

$$D = \begin{bmatrix} D_R & O \\ O & D_C \end{bmatrix} \quad D_R := \mathrm{diag}(A\mathbf{1}); D_C := \mathrm{diag}(A^{\mathsf{T}}\mathbf{1})$$

$$L = D - W = \begin{bmatrix} D_R & -A \\ -A^{\mathsf{T}} & D_C \end{bmatrix}$$

[1] I. S. Dhillon: Co-clustering documents and words using Bipartite Spectral Graph Partitioning, *Proc. 7th ACM SIGKDD*, pp. 269–274, 2001.

Matrix Partitioning à la Dhillon (2001)[1]

- Given a matrix $A \in \mathbb{R}^{N_R \times N_C}$, the rows and columns are viewed as the two sets of nodes in a *bipartite* graph.
- A_{ij} denotes the weight between the node for row i and the node for column j (If A is a term-document matrix, then A_{ij} is a relative frequency of occurrence of term i in the document j).
- Then, matrices associated with this bipartite graph can be written as:

$$W = \begin{bmatrix} O & A \\ A^{\mathsf{T}} & O \end{bmatrix}$$

$$D = \begin{bmatrix} D_R & O \\ O & D_C \end{bmatrix} \qquad D_R := \mathrm{diag}(A\mathbf{1}); D_C := \mathrm{diag}(A^{\mathsf{T}}\mathbf{1})$$

$$L = D - W = \begin{bmatrix} D_R & -A \\ -A^{\mathsf{T}} & D_C \end{bmatrix}$$

[1] I. S. Dhillon: Co-clustering documents and words using Bipartite Spectral Graph Partitioning, *Proc. 7th ACM SIGKDD*, pp. 269–274, 2001.

Matrix Partitioning à la Dhillon (2001)

- The Fiedler vector of L_{rw} bipartitions this bipartite graph:

$$\boldsymbol{\phi}_1 = \begin{bmatrix} D_R^{-1/2}\boldsymbol{u} \\ D_C^{-1/2}\boldsymbol{v} \end{bmatrix},$$

where $\boldsymbol{u}$ and $\boldsymbol{v}$ are the second left and right singular vectors of $D_R^{-1/2} A D_C^{-1/2}$.

- The rows and the columns are partitioned *simultaneously*.
- We *recursively* apply this bipartitioning method to generate a full partitioning of the rows and columns of the matrix.

Matrix Partitioning à la Dhillon (2001)

- The Fiedler vector of L_{rw} bipartitions this bipartite graph:

$$\boldsymbol{\phi}_1 = \begin{bmatrix} D_R^{-1/2}\boldsymbol{u} \\ D_C^{-1/2}\boldsymbol{v} \end{bmatrix},$$

  where $\boldsymbol{u}$ and $\boldsymbol{v}$ are the second left and right singular vectors of $D_R^{-1/2} A D_C^{-1/2}$.
- The rows and the columns are partitioned *simultaneously*.
- We *recursively* apply this bipartitioning method to generate a full partitioning of the rows and columns of the matrix.

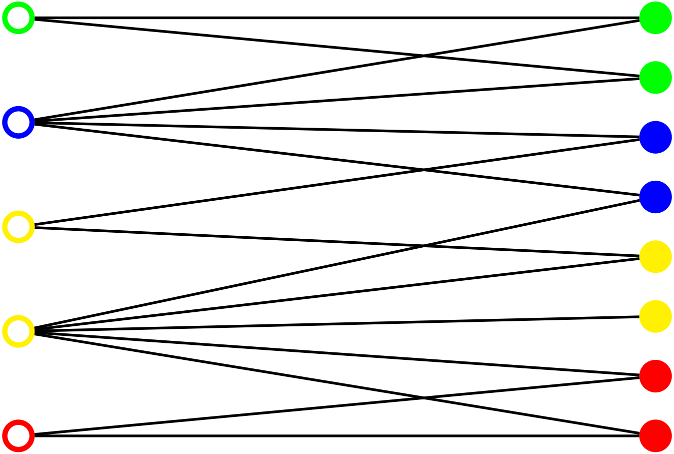
Matrix Partitioning à la Dhillon (2001)

- The Fiedler vector of L_{rw} bipartitions this bipartite graph:

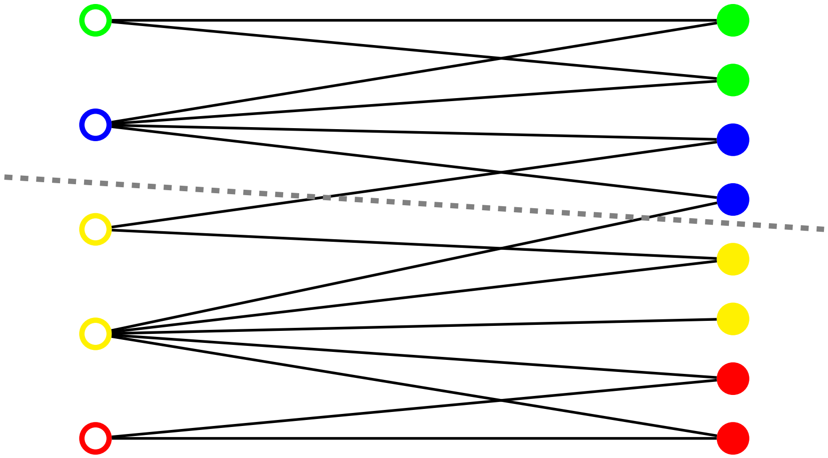
$$\boldsymbol{\phi}_1 = \begin{bmatrix} D_R^{-1/2}\boldsymbol{u} \\ D_C^{-1/2}\boldsymbol{v} \end{bmatrix},$$

  where $\boldsymbol{u}$ and $\boldsymbol{v}$ are the second left and right singular vectors of $D_R^{-1/2} A D_C^{-1/2}$.
- The rows and the columns are partitioned *simultaneously*.
- We *recursively* apply this bipartitioning method to generate a full partitioning of the rows and columns of the matrix.

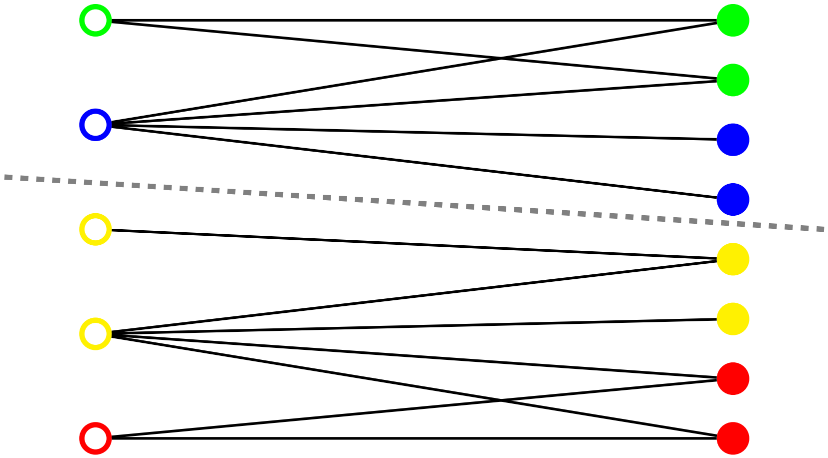
Matrix Partitioning à la Dhillon (2001)



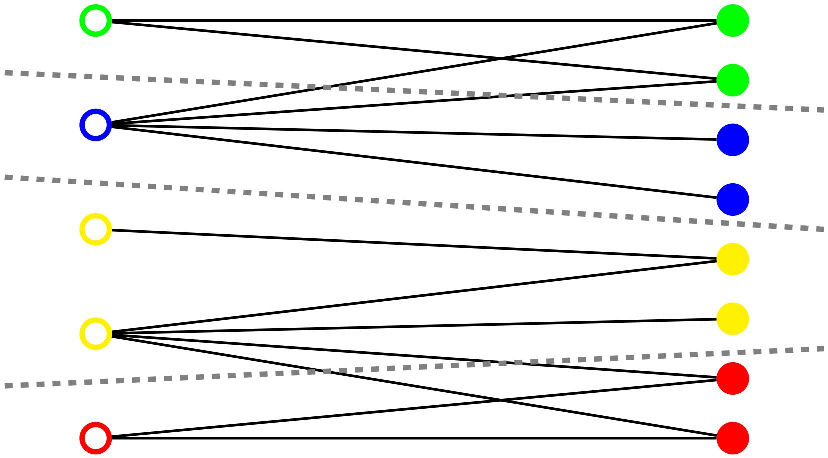
Matrix Partitioning à la Dhillon (2001)



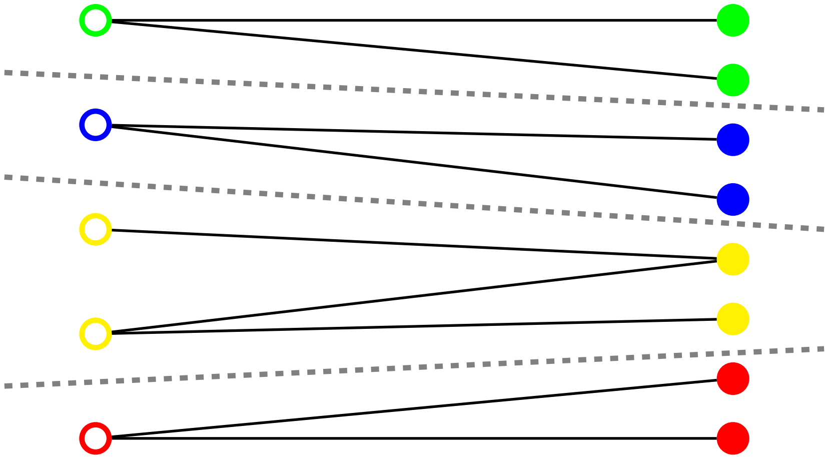
Matrix Partitioning à la Dhillon (2001)



Matrix Partitioning à la Dhillon (2001)



Matrix Partitioning à la Dhillon (2001)



Example 1: Science News Dataset

Dataset: the Science News database (1153×1042)

- Rows → (appropriately chosen) words
- Columns → article abstracts from 8 fields: Anthropology; Astronomy; Behavioral Sciences; Earth Sciences; Life Sciences; Math & CS; Medicine; Physics
- A_{ij} → the relative frequency of word i appears in abstract j ⇒ all column sums are 1

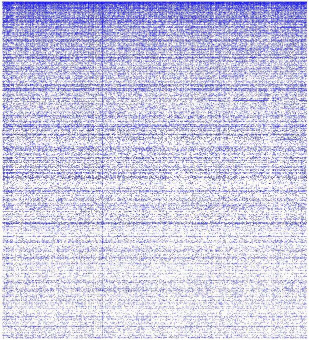


Figure: Science News database (original order)

Example 1: Science News Dataset

Dataset: the Science News database (1153×1042)

- Rows → (appropriately chosen) words
- Columns → article abstracts from 8 fields: Anthropology; Astronomy; Behavioral Sciences; Earth Sciences; Life Sciences; Math & CS; Medicine; Physics
- A_{ij} → the relative frequency of word i appears in abstract j ⇒ all column sums are 1

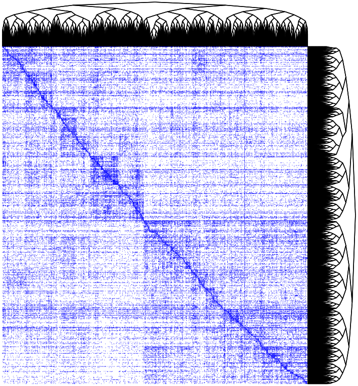


Figure: Science News database (reordered rows and columns)

Example 1: Science News Dataset

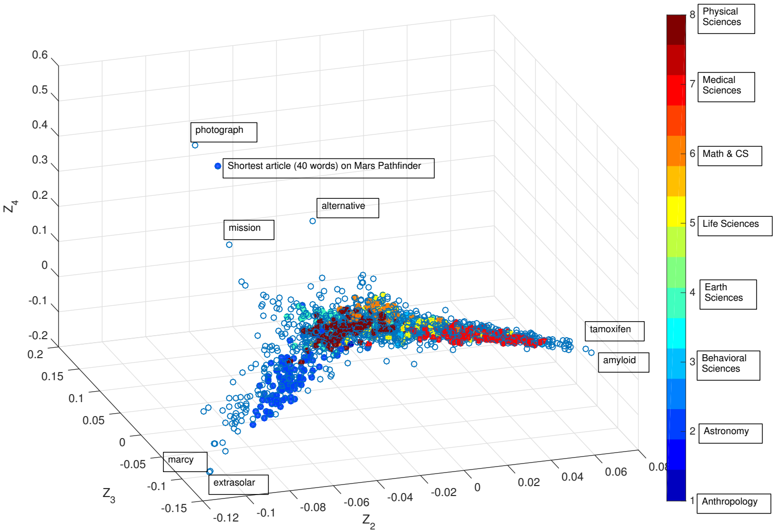


Figure: Words and abstracts embedded in $\{\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \boldsymbol{\phi}_3\}$ at the top level.

Example 1: Science News Dataset

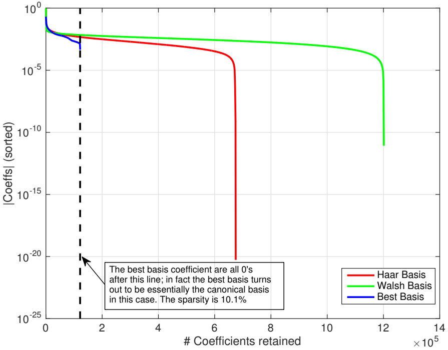


Figure: Haar basis vs. Walsh basis vs. GHWT best basis approximation results. The vertical line denotes the percentage of nonzero entries in the matrix (**10.1%**).

- Cost functional: 1-norm
- Total number of orthonormal bases searched: $> 10^{370}$
- **62.3%** of the Haar coefficients and **100%** of the Walsh coefficients must be kept to achieve perfect reconstruction, compared to **10.1%** for the GHWT best basis

⇒ The Haar and Walsh bases could not efficiently capture the underlying structure of this Science News dataset under the current matrix partitioning strategy!

Example 1: Science News Dataset

The GHWT best basis is almost exactly the canonical basis.

Combined Rows:

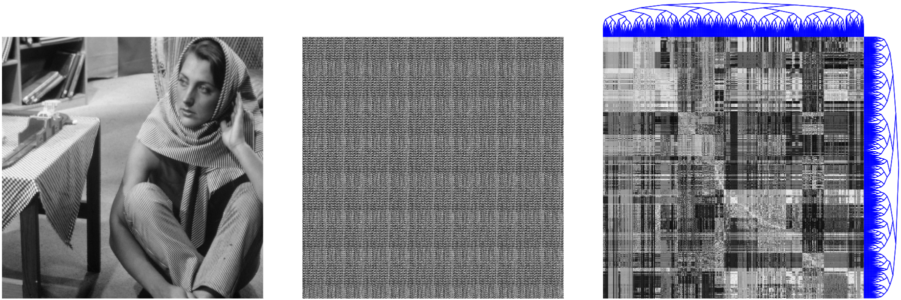
- "el" and "niño"
- "la" and "niña"
- "meteor" and "shower"

Combined Columns:

- "Science Talent Search announces Finalists" and "Talent Search: Student Finalists' Flair for science to be rewarded"

Example 2: The Shuffled Barbara Image

Dataset: the 512×512 "Barbara" image with the rows and columns shuffled.



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

Example 2: The Shuffled Barbara Image

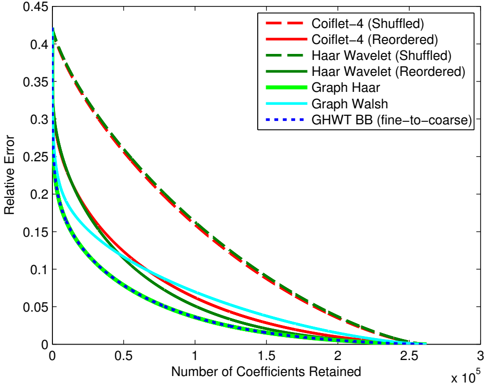


Figure: Approximation results. The "shuffled" and "reordered" results are for the cases that the shuffled image (middle figure on previous page) and reordered image (figure on the right) was analyzed, respectively.

- Cost functional: 1-norm
- Total number of ONBs searched: $> 6.37 \times 10^{173}$
- The GHWT BB nearly matches the graph Haar basis and performs better than the graph Walsh basis
- The GHWT BB performs much better than the Coiflet and Haar bases directly applied on the image, which are fixed and therefore cannot account for *nondyadic* geometry of the data

Example 2: The Shuffled Barbara Image

We can also use the GHWT and best basis algorithm to ascertain information about the spatial structure of the matrix data.

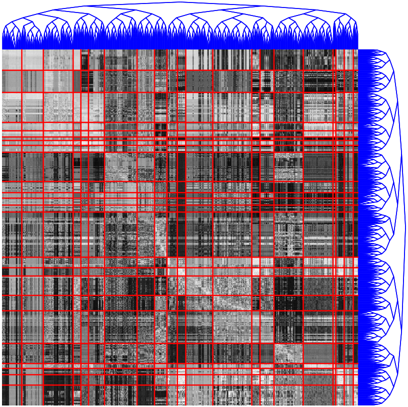


Figure: The coarse-to-fine row and column best bases for "Barbara" using the 0.1-quasinorm as our cost functional.

Example 2: The Shuffled Barbara Image

We can obtain different results by using a different cost functional.

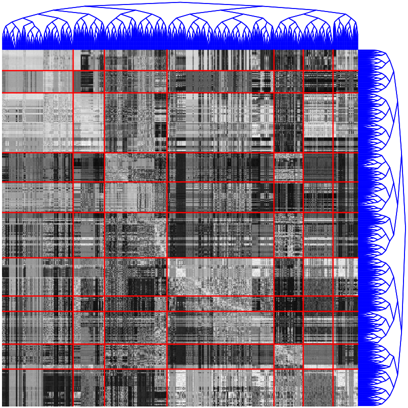


Figure: The coarse-to-fine row and column best bases for "Barbara" using the 0.5-quasinorm as our cost functional.

Example 2: The Shuffled Barbara Image

Another option is to not consider regions with fewer than $N_{\min}$ nodes.

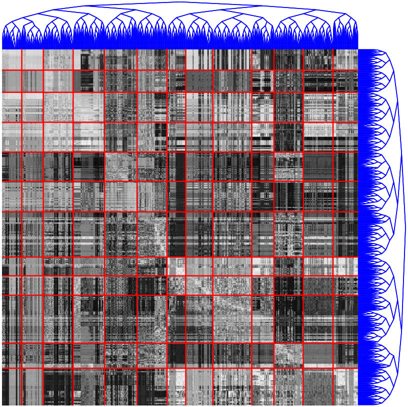


Figure: The coarse-to-fine row and column best bases for "Barbara" using the 0.1-quasinorm as our cost functional; regions with fewer than $[N_R/20] = [N_C/20] = 26$ nodes were not considered in the best basis search.

Discussion

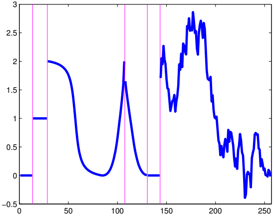
- Originally developed for signals on graphs, here we have shown the effectiveness of the GHWT for analyzing matrix data
- The GHWT best-basis algorithm searches over an immense number of orthonormal bases, including the graph Haar/Walsh bases
- When selected using an appropriate cost functional, the GHWT best basis equals or outperforms the graph Haar/Walsh bases
- This demonstrates the importance/advantage of a *data-adaptive basis dictionary* from which one can select the most suitable basis for one's task at hand!
- Should we add a *regularization* term in the cost functional to obtain a more *meaningful* basis, e.g., what combinations of words and articles are well captured by the top basis vectors selected as the best basis? $\Longrightarrow$ Local Regression Basis (LRB) of Saito and Coifman?

Outline

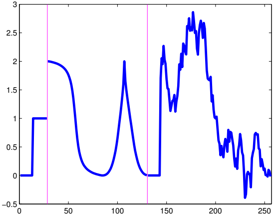
Motivation

- Thanks to the versatility of graphs, graph-based techniques have been used to tackle classical problems, e.g., the nonlocal means algorithm for image denoising can be viewed as a graph-based technique.
- Here, we demonstrate the versatility of our graph methods by applying the HGLET and hybrid best-basis algorithm to the problem of denoising and segmenting a 1-D signal sampled on a regular lattice into *meaningful* parts.

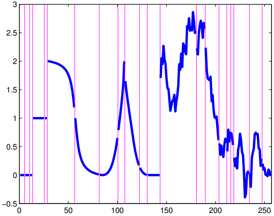
Simply put, the goal is to partition a given 1-D signal into segments based on the characteristics of the signal, which may help interpretation, analysis, compression, etc.



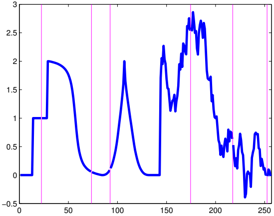
(a) Good



(b) Bad – too few segments



(c) Bad – too many segments



(d) Bad – segmentation lines are poorly placed

Method

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

Iterate until the best-basis segmentation converges:

1. **Recursively partition the graph:** Construct a recursive bipartitioning by minimizing *NCut* (*without* using the Fiedler vectors).
2. **Perform the 3 HGLET transforms:** Use the eigenvectors of L, L_{rw}, and L_{sym} of the *unweighted path graph*, i.e., three types of the DCTs (no eigenvector computation necessary).
3. **Find the hybrid best basis:** Use the MDL cost functional to search among the coefficients from the 3 HGLET variations.
4. **Modify the graph's edge weights:** Cut the edges that are 5% and 10% to the left and right of each partition in the best basis.

Reconstruct: synthesize the signal using the MDL-quantized coefficients.

Method

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

Iterate until the best-basis segmentation converges:

1. **Recursively partition the graph:** Construct a recursive bipartitioning by minimizing *NCut* (*without* using the Fiedler vectors).
2. **Perform the 3 HGLET transforms:** Use the eigenvectors of L, L_{rw}, and L_{sym} of the *unweighted path graph*, i.e., three types of the DCTs (no eigenvector computation necessary).
3. **Find the hybrid best basis:** Use the MDL cost functional to search among the coefficients from the 3 HGLET variations.
4. **Modify the graph's edge weights:** Cut the edges that are 5% and 10% to the left and right of each partition in the best basis.

Reconstruct: synthesize the signal using the MDL-quantized coefficients.

Method

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

Iterate until the best-basis segmentation converges:

1. **Recursively partition the graph:** Construct a recursive bipartitioning by minimizing *NCut* (*without* using the Fiedler vectors).
2. **Perform the 3 HGLET transforms:** Use the eigenvectors of L, L_{rw}, and L_{sym} of the *unweighted path graph*, i.e., three types of the DCTs (no eigenvector computation necessary).
3. **Find the hybrid best basis:** Use the MDL cost functional to search among the coefficients from the 3 HGLET variations.
4. **Modify the graph's edge weights:** Cut the edges that are 5% and 10% to the left and right of each partition in the best basis.

Reconstruct: synthesize the signal using the MDL-quantized coefficients.

Method

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

Iterate until the best-basis segmentation converges:

1. **Recursively partition the graph:** Construct a recursive bipartitioning by minimizing *NCut* (*without* using the Fiedler vectors).
2. **Perform the 3 HGLET transforms:** Use the eigenvectors of L, L_{rw}, and L_{sym} of the *unweighted path graph*, i.e., three types of the DCTs (no eigenvector computation necessary).
3. **Find the hybrid best basis:** Use the MDL cost functional to search among the coefficients from the 3 HGLET variations.
4. **Modify the graph's edge weights:** Cut the edges that are 5% and 10% to the left and right of each partition in the best basis.

Reconstruct: synthesize the signal using the MDL-quantized coefficients.

Method

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

Iterate until the best-basis segmentation converges:

1. **Recursively partition the graph:** Construct a recursive bipartitioning by minimizing *NCut* (*without* using the Fiedler vectors).
2. **Perform the 3 HGLET transforms:** Use the eigenvectors of L, L_{rw}, and L_{sym} of the *unweighted path graph*, i.e., three types of the DCTs (no eigenvector computation necessary).
3. **Find the hybrid best basis:** Use the MDL cost functional to search among the coefficients from the 3 HGLET variations.
4. **Modify the graph's edge weights:** Cut the edges that are 5% and 10% to the left and right of each partition in the best basis.

Reconstruct: synthesize the signal using the MDL-quantized coefficients.

Method

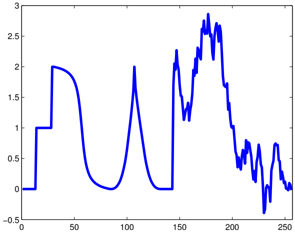
We view a 1-D classical signal as signal on an unweighted path graph and proceed as follows.

Iterate until the best-basis segmentation converges:

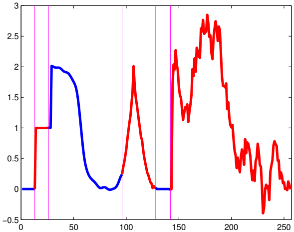
1. **Recursively partition the graph:** Construct a recursive bipartitioning by minimizing *NCut* (*without* using the Fiedler vectors).
2. **Perform the 3 HGLET transforms:** Use the eigenvectors of L, L_{rw}, and L_{sym} of the *unweighted path graph*, i.e., three types of the DCTs (no eigenvector computation necessary).
3. **Find the hybrid best basis:** Use the MDL cost functional to search among the coefficients from the 3 HGLET variations.
4. **Modify the graph's edge weights:** Cut the edges that are 5% and 10% to the left and right of each partition in the best basis.

Reconstruct: synthesize the signal using the MDL-quantized coefficients.

Results: Msignal ($n = 256$)



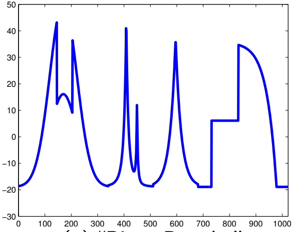
(a) Msignal



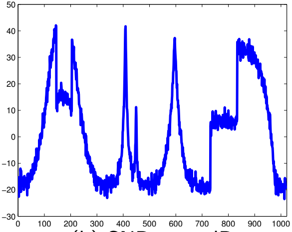
(b) Reconstruction with segmentation

Figure: **HGLET** L and **HGLET** L_{rw} segments; no segments are captured by **HGLET** L_{sym}.

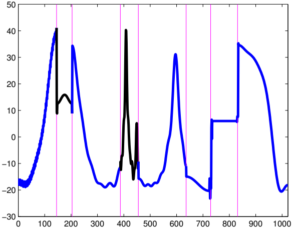
Results: Piece-Regular ($n = 1021$)



(a) "Piece-Regular"



(b) SNR $= 20$ dB



(c) SNR $= 23.85$ dB

Figure: **HGLET** L and **HGLET** L_{sym} segments (no **HGLET** L_{rw} segments).

Results: "Blocks" ($n = 2048$)

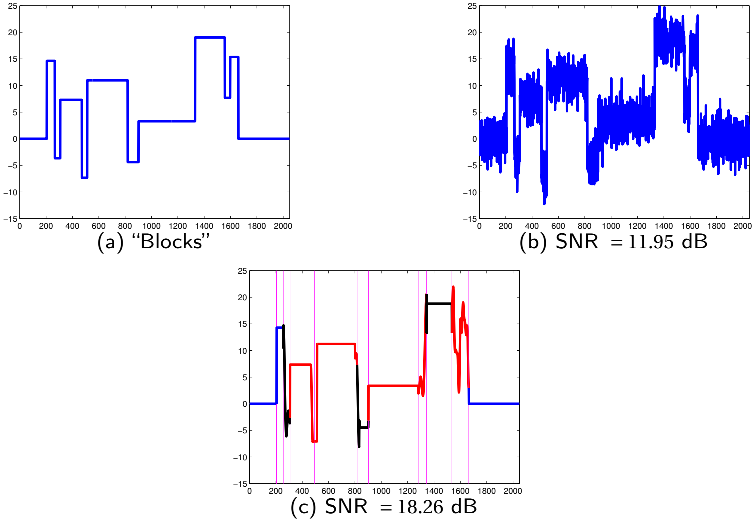


Figure: **HGLET** L, **HGLET** L_{rw}, and **HGLET** L_{sym} segments.

A Real Signal Example ($n = 2048$)

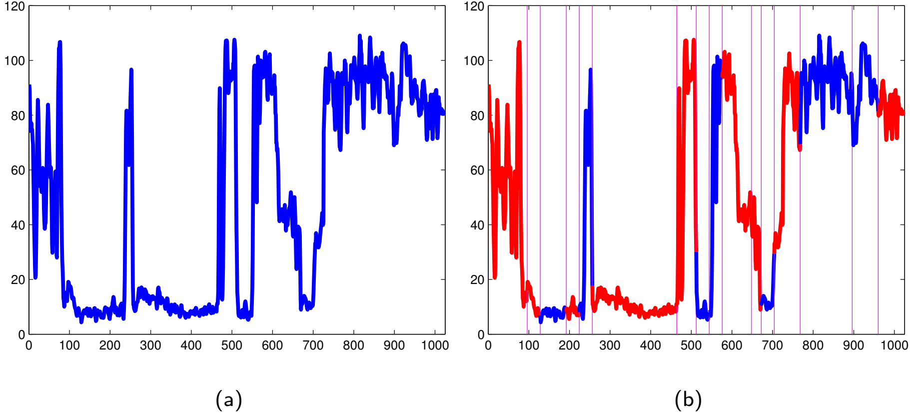


Figure: Gamma-ray log from North Sea subsurface formations. $\Delta z = 6$inches.

Discussion

- The MDL seeks an efficient way to represent the signal $\Rightarrow$ dissimilar regions are more efficiently represented separately than together.
- Partitions have a cost, and so regions will be merged unless keeping them separate offers a savings in cost that warrants the extra cost of the partition.
- Forcing artificial cuts around the BB partitions work like a perturbation similar to the spin-cycle method in denoising.
- No eigenvalue solver is necessary; everything is explicit, i.e., the true NCut computation and the fast DCTs $\Longrightarrow$ What to do with *image* segmentation on a 2D lattice?

Outline

Summary

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

References

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

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

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

References

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

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

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

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

- J. Irion & N. Saito: "Applied and computational harmonic analysis on graphs and networks," in *Wavelets and Sparsity XVI, Proc. SPIE 9597*, Paper # 95971F, 2015.
- N. Saito & E. Woei: "Tree simplification and the 'plateaux' phenomenon of graph Laplacian eigenvalues," *Linear Algebra and its Applications*, vol. 481, pp. 263–279, 2015.
- 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: "Hierarchical graph Laplacian eigen transforms," *JSIAM Letters*, vol. 6, pp. 21–24, 2014.
- Y. Nakatsukasa, N. Saito, & E. Woei: "Mysteries around graph Laplacian eigenvalue 4," *Linear Algebra and its Applications*, vol. 438, no. 8, pp. 3231–3246, 2013.
- N. Saito & E. Woei: "Analysis of neuronal dendrite patterns using eigenvalues of graph Laplacians," *JSIAM Letters*, vol. 1, pp. 13–16, 2009.

Thank you very much for your attention!