TUAT Intensive Course
Data Analysis on Graphs and Networks
Day 2: Basics of and Tools for Data Analysis on Graphs and Networks

Naoki Saito

Department of Mathematics
University of California, Davis

As a part of "Green & Clean Food Production Advancement I"
Fuchu Campus, Tokyo University of Agriculture & Technology
August 28, 2014

Outline

Acknowledgment

- Jeff Irion (UC Davis)
- Risa Naito (TUAT)
- Yuji Nakatsukasa (formerly UC Davis; currently Univ. of Tokyo)
- Kenshi Sakai (TUAT)
- Ernest Woei (formerly UC Davis; currently Flash Foto, Inc.)
- 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 MacTutor History of Mathematics Archive, Wikipedia, ...

Lecture Outline

Day 1 (13:00-16:15, August 27; Fuchu Campus):

- Motivations; Importance of Data Analysis on Networks and Graphs
- Basics (and Some History) of Fourier Analysis
- Basics of Data Representation and Compression on Regular Lattices via Linear Algebra and Fourier Analysis

Day 2 (13:00-18:00, August 28; Fuchu Campus):

- Basics of Graph Theory, Graph Laplacian Eigenvalues/Eigenvectors
- Graph partitioning
- Multiscale Basis Dictionaries on Graphs and Networks
- Applications (signal denoising, morphological analysis of neuronal dendritic trees, etc.)
- Discussions on potential agricultural applications including "Green and Clean Food Productions" with participants

Outline

Introductory Remarks

- For much more details of this part of the lecture, please check my course website on "Harmonic Analysis on Graphs & Networks": `http://www.math.ucdavis.edu/~saito/courses/HarmGraph/`
- Good general references on the graph Laplacian eigenvalues are:
  - R. B. Bapat: *Graphs and Matrices*, Universitext, Springer, 2010.
  - F. R. K. Chung: *Spectral Graph Theory*, Amer. Math. Soc., 1997.
  - D. Cvetković, P. Rowlinson, & S. Simić: *An Introduction to the Theory of Graph Spectra*, Vol. 75, London Mathematical Society Student Texts, Cambridge Univ. Press, 2010.
- As for the graph Laplacian eigenvectors, there are not too many books (although there may be many papers); one of the good books is
  - T. Bıyıkoğlu, J. Leydold, & P. F. Stadler, *Laplacian Eigenvectors of Graphs*, Lecture Notes in Mathematics, vol. 1915, Springer, 2007.
- As for *wavelet-like transforms* on graphs, there are many recent publications including those of my group. The following is a good survey paper:
  - D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, & P. Vandergheynst: "The emerging field of signal processing on graphs," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 83–98, 2013.

Today's Goals

- From my lecture on Day 1, we now know that the *localized* orthonormal transform such as BDCT is quite effective to analyze signals measured on a regular lattice (or equispaced grids).
- Since the optimal window size (or length) to chop the input signal is generally not known a priori, it would be better to develop multiscale orthonormal transforms, which accommodate multiple window sizes.
- Moreover, instead of regular lattices and grids, we now want to develop such multiscale orthonormal transforms on graphs and networks.
- To do so, we need to develop something equivalent to the cosine functions on graphs and networks.
- These turn out to be the eigenvectors of the so-called graph Laplacian matrix defined on a given graph.
- But first, we will go over the basics of graph theory for your convenience.

Today's Goals

- From my lecture on Day 1, we now know that the *localized* orthonormal transform such as BDCT is quite effective to analyze signals measured on a regular lattice (or equispaced grids).
- Since the optimal window size (or length) to chop the input signal is generally not known a priori, it would be better to develop multiscale orthonormal transforms, which accommodate multiple window sizes.
- Moreover, instead of regular lattices and grids, we now want to develop such multiscale orthonormal transforms on graphs and networks.
- To do so, we need to develop something equivalent to the cosine functions on graphs and networks.
- These turn out to be the eigenvectors of the so-called graph Laplacian matrix defined on a given graph.
- But first, we will go over the basics of graph theory for your convenience.

Today's Goals

- From my lecture on Day 1, we now know that the *localized* orthonormal transform such as BDCT is quite effective to analyze signals measured on a regular lattice (or equispaced grids).
- Since the optimal window size (or length) to chop the input signal is generally not known a priori, it would be better to develop multiscale orthonormal transforms, which accommodate multiple window sizes.
- Moreover, instead of regular lattices and grids, we now want to develop such multiscale orthonormal transforms on graphs and networks.
- To do so, we need to develop something equivalent to the cosine functions on graphs and networks.
- These turn out to be the eigenvectors of the so-called graph Laplacian matrix defined on a given graph.
- But first, we will go over the basics of graph theory for your convenience.

Today's Goals

- From my lecture on Day 1, we now know that the *localized* orthonormal transform such as BDCT is quite effective to analyze signals measured on a regular lattice (or equispaced grids).
- Since the optimal window size (or length) to chop the input signal is generally not known a priori, it would be better to develop multiscale orthonormal transforms, which accommodate multiple window sizes.
- Moreover, instead of regular lattices and grids, we now want to develop such multiscale orthonormal transforms on graphs and networks.
- To do so, we need to develop something equivalent to the cosine functions on graphs and networks.
- These turn out to be the eigenvectors of the so-called graph Laplacian matrix defined on a given graph.
- But first, we will go over the basics of graph theory for your convenience.

Today's Goals

- From my lecture on Day 1, we now know that the *localized* orthonormal transform such as BDCT is quite effective to analyze signals measured on a regular lattice (or equispaced grids).
- Since the optimal window size (or length) to chop the input signal is generally not known a priori, it would be better to develop multiscale orthonormal transforms, which accommodate multiple window sizes.
- Moreover, instead of regular lattices and grids, we now want to develop such multiscale orthonormal transforms on graphs and networks.
- To do so, we need to develop something equivalent to the cosine functions on graphs and networks.
- These turn out to be the eigenvectors of the so-called graph Laplacian matrix defined on a given graph.
- But first, we will go over the basics of graph theory for your convenience.

Today's Goals

- From my lecture on Day 1, we now know that the *localized* orthonormal transform such as BDCT is quite effective to analyze signals measured on a regular lattice (or equispaced grids).
- Since the optimal window size (or length) to chop the input signal is generally not known a priori, it would be better to develop multiscale orthonormal transforms, which accommodate multiple window sizes.
- Moreover, instead of regular lattices and grids, we now want to develop such multiscale orthonormal transforms on graphs and networks.
- To do so, we need to develop something equivalent to the cosine functions on graphs and networks.
- These turn out to be the eigenvectors of the so-called graph Laplacian matrix defined on a given graph.
- But first, we will go over the basics of graph theory for your convenience.

Outline

Basic Definitions

- A graph G consists of a set of vertices (or nodes) V and a set of edges E connecting some pairs of vertices in V. We write $G = (V, E)$.
- An edge connecting a vertex $x \in V$ and itself is called a loop.
- For $x, y \in V$, if $\exists$ more than one edge connecting x and y, they are called multiple edges.
- A graph having loops or multiple edges is called a multiple graph (or multigraph); otherwise it is called a simple graph.

- In this lecture, we shall only deal with simple graphs. So, when we say a graph, we mean a simple graph.

Basic Definitions

- A graph G consists of a set of vertices (or nodes) V and a set of edges E connecting some pairs of vertices in V. We write $G = (V, E)$.
- An edge connecting a vertex $x \in V$ and itself is called a loop.
- For $x, y \in V$, if $\exists$ more than one edge connecting x and y, they are called multiple edges.
- A graph having loops or multiple edges is called a multiple graph (or multigraph); otherwise it is called a simple graph.

- In this lecture, we shall only deal with simple graphs. So, when we say a graph, we mean a simple graph.

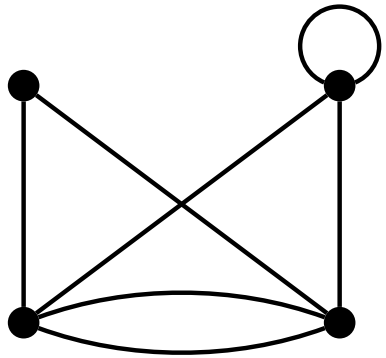
Basic Definitions

- A graph G consists of a set of vertices (or nodes) V and a set of edges E connecting some pairs of vertices in V. We write $G = (V, E)$.
- An edge connecting a vertex $x \in V$ and itself is called a loop.
- For $x, y \in V$, if $\exists$ more than one edge connecting x and y, they are called multiple edges.
- A graph having loops or multiple edges is called a multiple graph (or multigraph); otherwise it is called a simple graph.

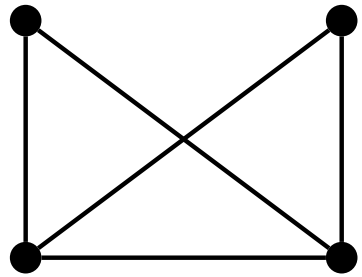
- In this lecture, we shall only deal with simple graphs. So, when we say a graph, we mean a simple graph.

Basic Definitions

- A graph G consists of a set of vertices (or nodes) V and a set of edges E connecting some pairs of vertices in V. We write $G = (V, E)$.
- An edge connecting a vertex $x \in V$ and itself is called a loop.
- For $x, y \in V$, if $\exists$ more than one edge connecting x and y, they are called multiple edges.
- A graph having loops or multiple edges is called a multiple graph (or multigraph); otherwise it is called a simple graph.



A multiple graph

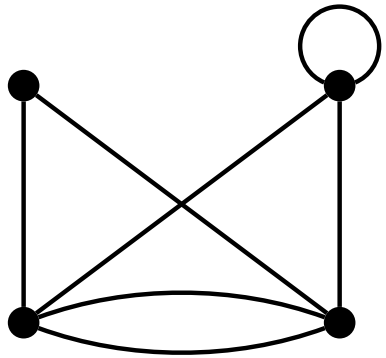


A simple graph

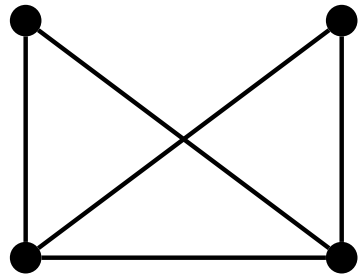
- In this lecture, we shall only deal with simple graphs. So, when we say a graph, we mean a simple graph.

Basic Definitions

- A graph G consists of a set of vertices (or nodes) V and a set of edges E connecting some pairs of vertices in V. We write $G = (V, E)$.
- An edge connecting a vertex $x \in V$ and itself is called a loop.
- For $x, y \in V$, if $\exists$ more than one edge connecting x and y, they are called multiple edges.
- A graph having loops or multiple edges is called a multiple graph (or multigraph); otherwise it is called a simple graph.



A multiple graph



A simple graph

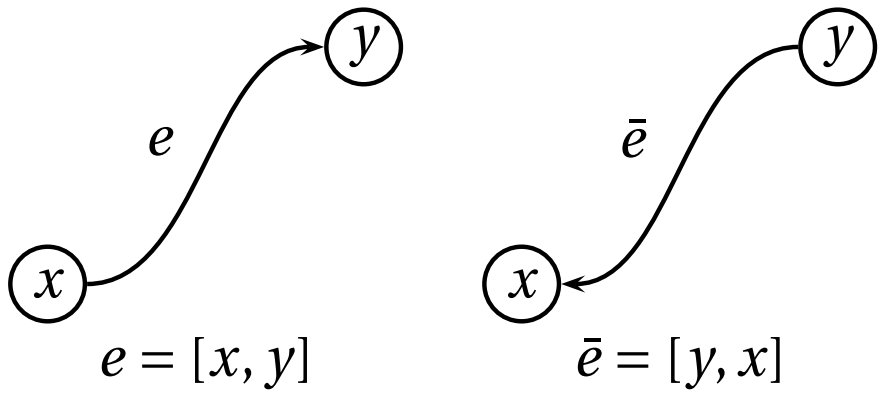
- In this lecture, we shall only deal with simple graphs. So, when we say a graph, we mean a simple graph.

- If two distinct vertices $x, y \in V$ are connected by an edge e, then x, y are called the endpoints (or ends) of e, and x, y are said to be adjacent, and we write $x \sim y$. We also say an edge e is incident with x and y, and e joins x and y.
- The number of edges that are incident with x (i.e., have x as their endpoint) = the degree (or valency) of x and write $d(x)$ or d_x.
- If the number of vertices $|V| < \infty$, then G is called a finite graph; otherwise an infinite graph.
- If each edge in E has a direction, G is called a directed graph or digraph, and such E is written as $\vec{E}$.
- If $e = [x, y]$, then x and y are called a tail and a head, respectively.

- If two distinct vertices $x, y \in V$ are connected by an edge e, then x, y are called the endpoints (or ends) of e, and x, y are said to be adjacent, and we write $x \sim y$. We also say an edge e is incident with x and y, and e joins x and y.
- The number of edges that are incident with x (i.e., have x as their endpoint) = the degree (or valency) of x and write $d(x)$ or d_x.
- If the number of vertices $|V| < \infty$, then G is called a finite graph; otherwise an infinite graph.
- If each edge in E has a direction, G is called a directed graph or digraph, and such E is written as $\vec{E}$.
- If $e = [x, y]$, then x and y are called a tail and a head, respectively.

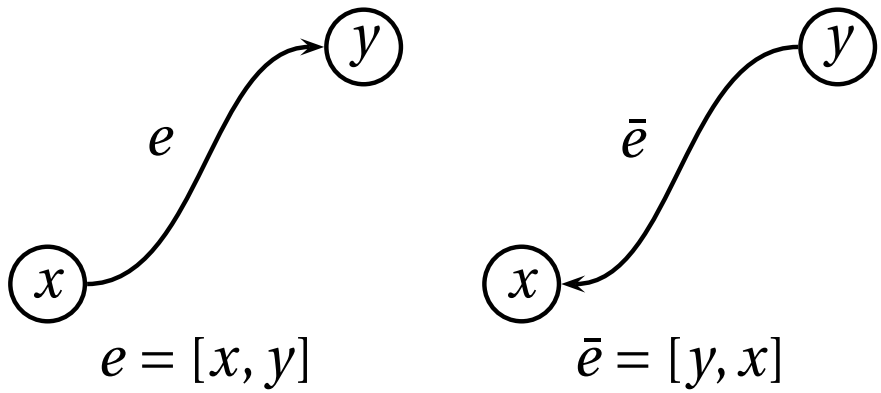
- If two distinct vertices $x, y \in V$ are connected by an edge e, then x, y are called the endpoints (or ends) of e, and x, y are said to be adjacent, and we write $x \sim y$. We also say an edge e is incident with x and y, and e joins x and y.
- The number of edges that are incident with x (i.e., have x as their endpoint) = the degree (or valency) of x and write $d(x)$ or d_x.
- If the number of vertices $|V| < \infty$, then G is called a finite graph; otherwise an infinite graph.
- If each edge in E has a direction, G is called a directed graph or digraph, and such E is written as $\vec{E}$.
- If $e = [x, y]$, then x and y are called a tail and a head, respectively.

- If two distinct vertices $x, y \in V$ are connected by an edge e, then x, y are called the endpoints (or ends) of e, and x, y are said to be adjacent, and we write $x \sim y$. We also say an edge e is incident with x and y, and e joins x and y.
- The number of edges that are incident with x (i.e., have x as their endpoint) = the degree (or valency) of x and write $d(x)$ or d_x.
- If the number of vertices $|V| < \infty$, then G is called a finite graph; otherwise an infinite graph.
- If each edge in E has a direction, G is called a directed graph or digraph, and such E is written as $\boldsymbol{E}$.



- If $e = [x, y]$, then x and y are called a tail and a head, respectively.

- If two distinct vertices $x, y \in V$ are connected by an edge e, then x, y are called the endpoints (or ends) of e, and x, y are said to be adjacent, and we write $x \sim y$. We also say an edge e is incident with x and y, and e joins x and y.
- The number of edges that are incident with x (i.e., have x as their endpoint) = the degree (or valency) of x and write $d(x)$ or d_x.
- If the number of vertices $|V| < \infty$, then G is called a finite graph; otherwise an infinite graph.
- If each edge in E has a direction, G is called a directed graph or digraph, and such E is written as $\boldsymbol{E}$.



- If $e = [x, y]$, then x and y are called a tail and a head, respectively.

- If an edge e does not have a direction, we write $e = (x, y)$.
- If each edge $e = (x, y)$ of G has a weight (normally positive), written as $w_e = w_{xy}$, then G is called a weighted graph. G is said to be unweighted if $w_e = \text{const.}$ for each $e \in E$, and normally w_e is set to 1.
- A path from x to y in a graph G is a subgraph of G consisting of a sequence of distinct vertices starting with x and ending with y such that consecutive vertices are adjacent. A path starting from x that returns to x (but is not a loop) is called a cycle.
- For any two vertices in V, if $\exists$ a path connecting them, then such a graph G is said to be connected. In the case of a digraph, it is said to be strongly connected.
- A tree is a connected graph without cycles, and is often denoted by T instead of G. For a tree T, we have $|E(T)| = |V(T)| - 1$, where $|\cdot|$ denotes a cardinality of a set.
- The length (or cost) $\ell(P)$ of a path P is the sum of its corresponding edge weights, i.e., $\ell(P) := \sum_{e \in E(P)} w_e$. Let $\mathscr{P}_{xy}$ be a set of all possible paths from x to y in G. The graph distance from x to y is defined by $d(x, y) := \inf_{P \in \mathscr{P}_{xy}} \ell(P)$.

- If an edge e does not have a direction, we write $e = (x, y)$.
- If each edge $e = (x, y)$ of G has a weight (normally positive), written as $w_e = w_{xy}$, then G is called a weighted graph. G is said to be unweighted if $w_e = \text{const.}$ for each $e \in E$, and normally w_e is set to 1.
- A path from x to y in a graph G is a subgraph of G consisting of a sequence of distinct vertices starting with x and ending with y such that consecutive vertices are adjacent. A path starting from x that returns to x (but is not a loop) is called a cycle.
- For any two vertices in V, if $\exists$ a path connecting them, then such a graph G is said to be connected. In the case of a digraph, it is said to be strongly connected.
- A tree is a connected graph without cycles, and is often denoted by T instead of G. For a tree T, we have $|E(T)| = |V(T)| - 1$, where $|\cdot|$ denotes a cardinality of a set.
- The length (or cost) $\ell(P)$ of a path P is the sum of its corresponding edge weights, i.e., $\ell(P) := \sum_{e \in E(P)} w_e$. Let $\mathscr{P}_{xy}$ be a set of all possible paths from x to y in G. The graph distance from x to y is defined by $d(x, y) := \inf_{P \in \mathscr{P}_{xy}} \ell(P)$.

- If an edge e does not have a direction, we write $e = (x, y)$.
- If each edge $e = (x, y)$ of G has a weight (normally positive), written as $w_e = w_{xy}$, then G is called a weighted graph. G is said to be unweighted if $w_e = \text{const.}$ for each $e \in E$, and normally w_e is set to 1.
- A path from x to y in a graph G is a subgraph of G consisting of a sequence of distinct vertices starting with x and ending with y such that consecutive vertices are adjacent. A path starting from x that returns to x (but is not a loop) is called a cycle.
- For any two vertices in V, if $\exists$ a path connecting them, then such a graph G is said to be connected. In the case of a digraph, it is said to be strongly connected.
- A tree is a connected graph without cycles, and is often denoted by T instead of G. For a tree T, we have $|E(T)| = |V(T)| - 1$, where $|\cdot|$ denotes a cardinality of a set.
- The length (or cost) $\ell(P)$ of a path P is the sum of its corresponding edge weights, i.e., $\ell(P) := \sum_{e \in E(P)} w_e$. Let $\mathscr{P}_{xy}$ be a set of all possible paths from x to y in G. The graph distance from x to y is defined by $d(x, y) := \inf_{P \in \mathscr{P}_{xy}} \ell(P)$.

- If an edge e does not have a direction, we write $e = (x, y)$.
- If each edge $e = (x, y)$ of G has a weight (normally positive), written as $w_e = w_{xy}$, then G is called a weighted graph. G is said to be unweighted if $w_e = \text{const.}$ for each $e \in E$, and normally w_e is set to 1.
- A path from x to y in a graph G is a subgraph of G consisting of a sequence of distinct vertices starting with x and ending with y such that consecutive vertices are adjacent. A path starting from x that returns to x (but is not a loop) is called a cycle.
- For any two vertices in V, if $\exists$ a path connecting them, then such a graph G is said to be connected. In the case of a digraph, it is said to be strongly connected.
- A tree is a connected graph without cycles, and is often denoted by T instead of G. For a tree T, we have $|E(T)| = |V(T)| - 1$, where $|\cdot|$ denotes a cardinality of a set.
- The length (or cost) $\ell(P)$ of a path P is the sum of its corresponding edge weights, i.e., $\ell(P) := \sum_{e \in E(P)} w_e$. Let $\mathscr{P}_{xy}$ be a set of all possible paths from x to y in G. The graph distance from x to y is defined by $d(x, y) := \inf_{P \in \mathscr{P}_{xy}} \ell(P)$.

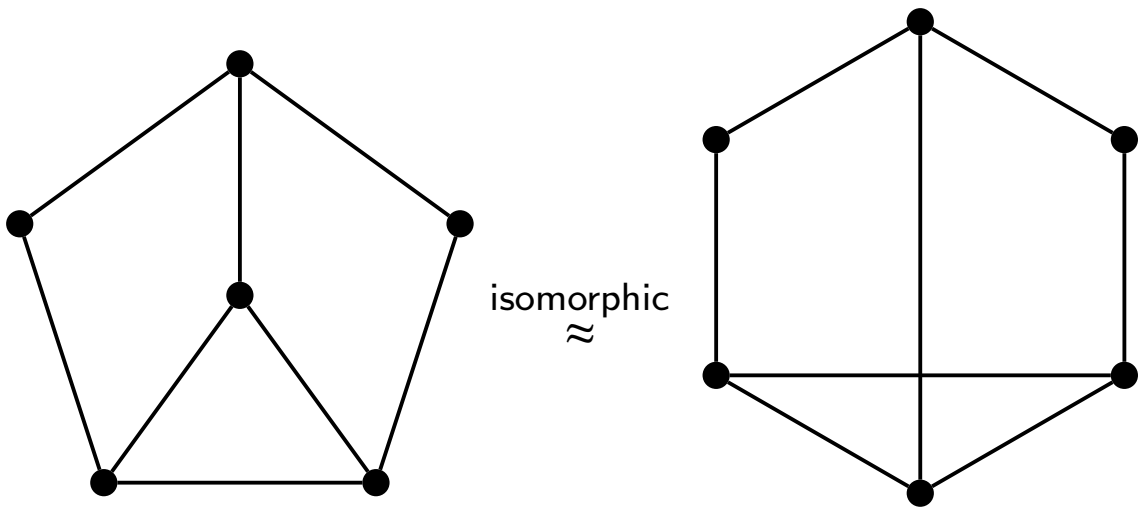
- If an edge e does not have a direction, we write $e = (x, y)$.
- If each edge $e = (x, y)$ of G has a weight (normally positive), written as $w_e = w_{xy}$, then G is called a weighted graph. G is said to be unweighted if $w_e = \text{const.}$ for each $e \in E$, and normally w_e is set to 1.
- A path from x to y in a graph G is a subgraph of G consisting of a sequence of distinct vertices starting with x and ending with y such that consecutive vertices are adjacent. A path starting from x that returns to x (but is not a loop) is called a cycle.
- For any two vertices in V, if $\exists$ a path connecting them, then such a graph G is said to be connected. In the case of a digraph, it is said to be strongly connected.
- A tree is a connected graph without cycles, and is often denoted by T instead of G. For a tree T, we have $|E(T)| = |V(T)| - 1$, where $|\cdot|$ denotes a cardinality of a set.
- The length (or cost) $\ell(P)$ of a path P is the sum of its corresponding edge weights, i.e., $\ell(P) := \sum_{e \in E(P)} w_e$. Let $\mathscr{P}_{xy}$ be a set of all possible paths from x to y in G. The graph distance from x to y is defined by $d(x, y) := \inf_{P \in \mathscr{P}_{xy}} \ell(P)$.

- If an edge e does not have a direction, we write $e = (x, y)$.
- If each edge $e = (x, y)$ of G has a weight (normally positive), written as $w_e = w_{xy}$, then G is called a weighted graph. G is said to be unweighted if $w_e = \text{const.}$ for each $e \in E$, and normally w_e is set to 1.
- A path from x to y in a graph G is a subgraph of G consisting of a sequence of distinct vertices starting with x and ending with y such that consecutive vertices are adjacent. A path starting from x that returns to x (but is not a loop) is called a cycle.
- For any two vertices in V, if $\exists$ a path connecting them, then such a graph G is said to be connected. In the case of a digraph, it is said to be strongly connected.
- A tree is a connected graph without cycles, and is often denoted by T instead of G. For a tree T, we have $|E(T)| = |V(T)| - 1$, where $|\cdot|$ denotes a cardinality of a set.
- The length (or cost) $\ell(P)$ of a path P is the sum of its corresponding edge weights, i.e., $\ell(P) := \sum_{e \in E(P)} w_e$. Let $\mathscr{P}_{xy}$ be a set of all possible paths from x to y in G. The graph distance from x to y is defined by $d(x, y) := \inf_{P \in \mathscr{P}_{xy}} \ell(P)$.

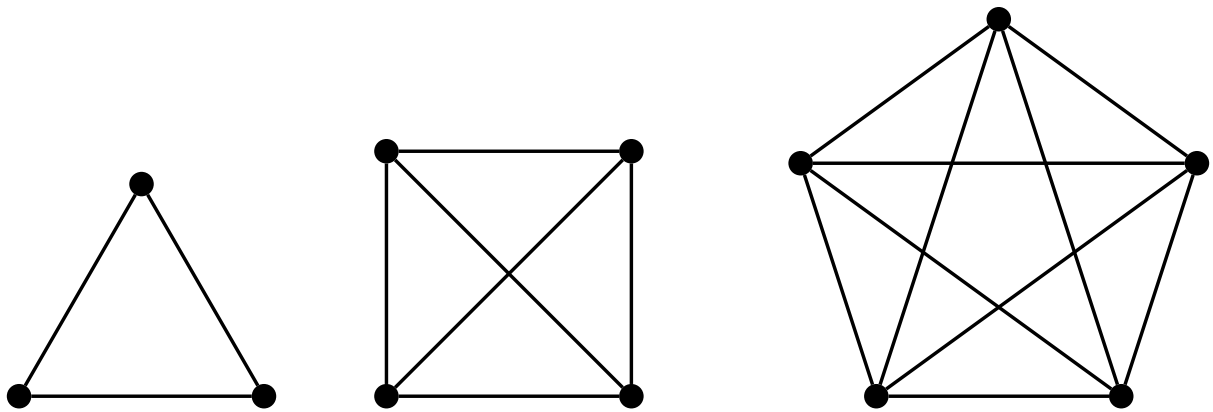
- Clearly, for an undirected graph, we always have $d(x,y) = d(y,x)$, but that is not the case for a directed graph in general.
- $\mathrm{diam}(G) := \sup_{x,y \in V} d(x,y)$ is called the diameter of G. Note that $\mathrm{diam}(G) < \infty \Longleftrightarrow G$ is finite.
- We say two graphs are isomorphic if $\exists$ a one-to-one correspondence between the vertex sets such that if two vertices are joined by an edge in one graph, the corresponding vertices are also joined by an edge in the other graph.

- Clearly, for an undirected graph, we always have $d(x,y) = d(y,x)$, but that is not the case for a directed graph in general.
- $\mathrm{diam}(G) := \sup_{x,y \in V} d(x,y)$ is called the diameter of G. Note that $\mathrm{diam}(G) < \infty \Longleftrightarrow G$ is finite.
- We say two graphs are isomorphic if $\exists$ a one-to-one correspondence between the vertex sets such that if two vertices are joined by an edge in one graph, the corresponding vertices are also joined by an edge in the other graph.

- Clearly, for an undirected graph, we always have $d(x,y) = d(y,x)$, but that is not the case for a directed graph in general.
- $\mathrm{diam}(G) := \sup_{x,y \in V} d(x,y)$ is called the diameter of G. Note that $\mathrm{diam}(G) < \infty \iff G$ is finite.
- We say two graphs are isomorphic if $\exists$ a one-to-one correspondence between the vertex sets such that if two vertices are joined by an edge in one graph, the corresponding vertices are also joined by an edge in the other graph.



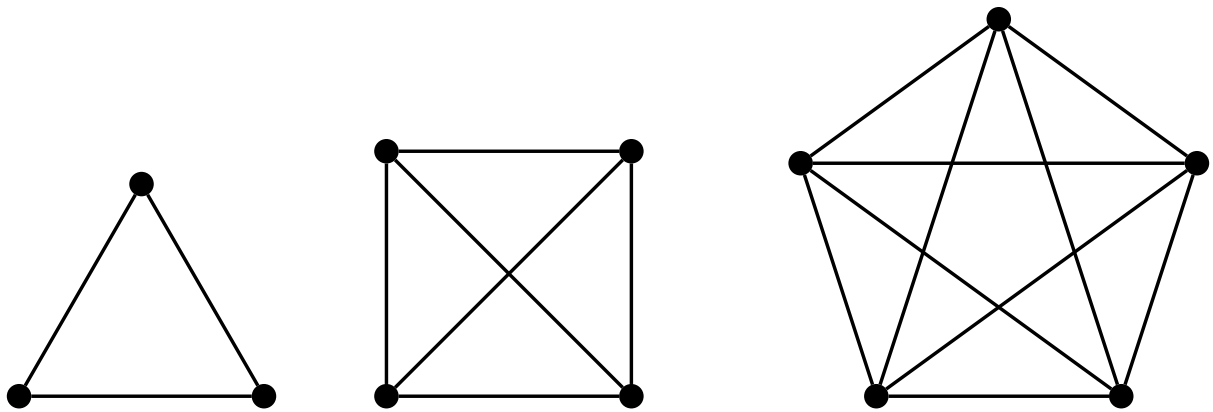
- The complete graph K_n on n vertices is a simple graph that has all possible $\binom{n}{2}$ edges.



K_3 K_4 K_5

- If all the vertices of a graph has the same degree, the graph is called regular. Hence, K_n is regular.

- The complete graph K_n on n vertices is a simple graph that has all possible $\binom{n}{2}$ edges.



K_3 K_4 K_5

- If all the vertices of a graph has the same degree, the graph is called regular. Hence, K_n is regular.

Matrices Associated with a Graph

- The adjacency matrix $A = A(G) = (a_{ij}) \in \mathbb{R}^{n \times n}$, $n = |V|$, for an unweighted graph G consists of the following entries:

$$a_{ij} := \begin{cases} 1 & \text{if } v_i \sim v_j; \\ 0 & \text{otherwise.} \end{cases}$$

- Another typical way to define its entries is based on the similarity of information at v_i and v_j:

$$a_{ij} := \exp(-\mathrm{dist}(v_i, v_j)^2/\epsilon^2)$$

where dist is an appropriate distance measure (i.e., metric) defined in V, and $\epsilon > 0$ is an appropriate scale parameter. This leads to a weighted graph. We will discuss later more about the weighted graphs, how to determine weights, and how to construct a graph from given datasets in general.

Matrices Associated with a Graph

- The adjacency matrix $A = A(G) = (a_{ij}) \in \mathbb{R}^{n \times n}$, $n = |V|$, for an unweighted graph G consists of the following entries:

$$a_{ij} := \begin{cases} 1 & \text{if } v_i \sim v_j; \\ 0 & \text{otherwise.} \end{cases}$$

- Another typical way to define its entries is based on the similarity of information at v_i and v_j:

$$a_{ij} := \exp(-\mathrm{dist}(v_i, v_j)^2/\epsilon^2)$$

where dist is an appropriate distance measure (i.e., metric) defined in V, and $\epsilon > 0$ is an appropriate scale parameter. This leads to a weighted graph. We will discuss later more about the weighted graphs, how to determine weights, and how to construct a graph from given datasets in general.

Matrices Associated with a Graph ...

- The degree matrix $D = D(G) = \mathrm{diag}(d_1, \ldots, d_n) \in \mathbb{R}^{n \times n}$ is a diagonal matrix whose entries are:

$$d_i = d(v_i) = d_{v_i} := \sum_{j=1}^{n} a_{ij}.$$

  Note that the above definition works for both unweighted and weighted graphs.

- The transition matrix $P = P(G) = (p_{ij}) \in \mathbb{R}^{n \times n}$ consists of the following entries:

$$p_{ij} := a_{ij}/d_i \quad \text{if } d_i \neq 0.$$

- p_{ij} represents the probability of a random walk from v_i to v_j in one step: $\sum_j p_{ij} = 1$, i.e., P is row stochastic.
- $A^\top = A$, $P^\top \neq P$, $P = D^{-1}A$.

Matrices Associated with a Graph ...

- The degree matrix $D = D(G) = \mathrm{diag}(d_1, \ldots, d_n) \in \mathbb{R}^{n \times n}$ is a diagonal matrix whose entries are:

$$d_i = d(v_i) = d_{v_i} := \sum_{j=1}^{n} a_{ij}.$$

  Note that the above definition works for both unweighted and weighted graphs.

- The transition matrix $P = P(G) = (p_{ij}) \in \mathbb{R}^{n \times n}$ consists of the following entries:

$$p_{ij} := a_{ij}/d_i \quad \text{if } d_i \neq 0.$$

- p_{ij} represents the probability of a random walk from v_i to v_j in one step: $\sum_j p_{ij} = 1$, i.e., P is row stochastic.
- $A^\top = A$, $P^\top \neq P$, $P = D^{-1}A$.

Matrices Associated with a Graph ...

- The degree matrix $D = D(G) = \mathrm{diag}(d_1, \ldots, d_n) \in \mathbb{R}^{n \times n}$ is a diagonal matrix whose entries are:

$$d_i = d(v_i) = d_{v_i} := \sum_{j=1}^{n} a_{ij}.$$

  Note that the above definition works for both unweighted and weighted graphs.

- The transition matrix $P = P(G) = (p_{ij}) \in \mathbb{R}^{n \times n}$ consists of the following entries:

$$p_{ij} := a_{ij}/d_i \quad \text{if } d_i \neq 0.$$

- p_{ij} represents the probability of a random walk from v_i to v_j in one step: $\sum_j p_{ij} = 1$, i.e., P is row stochastic.
- $A^\top = A$, $P^\top \neq P$, $P = D^{-1}A$.

Matrices Associated with a Graph ...

- The degree matrix $D = D(G) = \mathrm{diag}(d_1, \ldots, d_n) \in \mathbb{R}^{n \times n}$ is a diagonal matrix whose entries are:

$$d_i = d(v_i) = d_{v_i} := \sum_{j=1}^{n} a_{ij}.$$

  Note that the above definition works for both unweighted and weighted graphs.

- The transition matrix $P = P(G) = (p_{ij}) \in \mathbb{R}^{n \times n}$ consists of the following entries:

$$p_{ij} := a_{ij}/d_i \quad \text{if } d_i \neq 0.$$

- p_{ij} represents the probability of a random walk from v_i to v_j in one step: $\sum_j p_{ij} = 1$, i.e., P is row stochastic.
- $A^\mathsf{T} = A$, $P^\mathsf{T} \neq P$, $P = D^{-1}A$.

Matrices Associated with a Graph ...

- Let G be an *undirected* graph. Then, we can define several Laplacian matrices of G:

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

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

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

- The signless Laplacian is defined as follows, but we will not deal with this in this lecture: $Q(G) := D + A$.
- 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 G be an *undirected* graph. Then, we can define several Laplacian matrices of G:

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

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

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

- The signless Laplacian is defined as follows, but we will not deal with this in this lecture: $Q(G) := D + A$.
- 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 G be an *undirected* graph. Then, we can define several Laplacian matrices of G:

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

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

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

- The signless Laplacian is defined as follows, but we will not deal with this in this lecture: $Q(G) := D + A$.
- 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.

Functions Defined on a Graph

$$C(V) := \{\text{all functions defined on } V\}$$

$$C_0(V) := \{f \in C(V) \,|\, \operatorname{supp} f \text{ is a finite subset of } V\}$$

$$\operatorname{supp} f := \{u \in V \,|\, f(u) \neq 0\}$$

$$\langle f, g \rangle := \sum_{u \in V} f(u) g(u)$$

$$\langle f, g \rangle_{\#} := \sum_{u \in V} d(u) f(u) g(u)$$

$$\mathcal{L}^2(V) := \left\{ f \in C(V) \,\middle|\, \|f\|_{\#} := \sqrt{\langle f, f \rangle_{\#}} < \infty \right\}$$

Lemma

$$\langle Pf, g \rangle_{\#} = \langle f, Pg \rangle_{\#} \quad \forall f, g \in \mathcal{L}^2(V);$$

$$\|Pf\|_{\#} \leq \|f\|_{\#} \quad \forall f \in \mathcal{L}^2(V).$$

Functions Defined on a Graph ...

- Let $f \in \mathcal{L}^2(V)$. Then

$$Lf(v_i) = d_i f(v_i) - \sum_{j=1}^{n} a_{ij} f(v_j) = \sum_{j=1}^{n} a_{ij} \big(f(v_i) - f(v_j)\big).$$

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

- On the other hand,

$$L_{\mathrm{rw}} f(v_i) = f(v_i) - \sum_{j=1}^{n} p_{ij} f(v_j) = \frac{1}{d_i} \sum_{j=1}^{n} a_{ij} \big(f(v_i) - f(v_j)\big).$$

$$L_{\mathrm{sym}} f(v_i) = f(v_i) - \frac{1}{\sqrt{d_i}} \sum_{j=1}^{n} \frac{a_{ij}}{\sqrt{d_j}} f(v_j) = \frac{1}{\sqrt{d_i}} \sum_{j=1}^{n} a_{ij} \left(\frac{f(v_i)}{\sqrt{d_i}} - \frac{f(v_j)}{\sqrt{d_j}} \right).$$

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

Functions Defined on a Graph ...

- Let $f \in \mathcal{L}^2(V)$. Then

$$Lf(v_i) = d_i f(v_i) - \sum_{j=1}^{n} a_{ij} f(v_j) = \sum_{j=1}^{n} a_{ij} \big(f(v_i) - f(v_j)\big).$$

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

- On the other hand,

$$L_{\mathrm{rw}} f(v_i) = f(v_i) - \sum_{j=1}^{n} p_{ij} f(v_j) = \frac{1}{d_i} \sum_{j=1}^{n} a_{ij} \big(f(v_i) - f(v_j)\big).$$

$$L_{\mathrm{sym}} f(v_i) = f(v_i) - \frac{1}{\sqrt{d_i}} \sum_{j=1}^{n} \frac{a_{ij}}{\sqrt{d_j}} f(v_j) = \frac{1}{\sqrt{d_i}} \sum_{j=1}^{n} a_{ij} \left(\frac{f(v_i)}{\sqrt{d_i}} - \frac{f(v_j)}{\sqrt{d_j}} \right).$$

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

Functions Defined on a Graph ...

- Let $f \in \mathcal{L}^2(V)$. Then

$$Lf(v_i) = d_i f(v_i) - \sum_{j=1}^{n} a_{ij} f(v_j) = \sum_{j=1}^{n} a_{ij} \big(f(v_i) - f(v_j)\big).$$

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

- On the other hand,

$$L_{\mathrm{rw}} f(v_i) = f(v_i) - \sum_{j=1}^{n} p_{ij} f(v_j) = \frac{1}{d_i} \sum_{j=1}^{n} a_{ij} \big(f(v_i) - f(v_j)\big).$$

$$L_{\mathrm{sym}} f(v_i) = f(v_i) - \frac{1}{\sqrt{d_i}} \sum_{j=1}^{n} \frac{a_{ij}}{\sqrt{d_j}} f(v_j) = \frac{1}{\sqrt{d_i}} \sum_{j=1}^{n} a_{ij} \left(\frac{f(v_i)}{\sqrt{d_i}} - \frac{f(v_j)}{\sqrt{d_j}} \right).$$

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

Functions Defined on a Graph ...

- A function $f \in C(V)$ is called harmonic if

$$Lf = 0, \; L_{\mathrm{rw}} f = 0, \text{ or } \; L_{\mathrm{sym}} f = 0.$$

- A function $f \in C(V)$ is called superharmonic at $x \in V$ if

$$Lf(x) \geq 0, \; L_{\mathrm{rw}} f(x) \geq 0, \text{ or } \; L_{\mathrm{sym}} f(x) \geq 0.$$

- These corresponds to:

$$f(v_i) \geq \frac{1}{d_i} \sum_{j=1}^{n} a_{ij} f(v_j), \; f(v_i) \geq \sum_{j=1}^{n} p_{ij} f(v_j), \text{ or } \; f(v_i) \geq \sum_{j=1}^{n} \frac{a_{ij}}{\sqrt{d_i}\sqrt{d_j}} f(v_j).$$

- One can also generalize various analytic concepts such as Green's functions, Green's identity, analytic functions, Cauchy-Riemann equations, ..., to the graph setting!

Functions Defined on a Graph ...

- A function $f \in C(V)$ is called harmonic if

$$Lf = 0, \; L_{\mathrm{rw}} f = 0, \text{ or } \; L_{\mathrm{sym}} f = 0.$$

- A function $f \in C(V)$ is called superharmonic at $x \in V$ if

$$Lf(x) \geq 0, \; L_{\mathrm{rw}} f(x) \geq 0, \text{ or } \; L_{\mathrm{sym}} f(x) \geq 0.$$

- These corresponds to:

$$f(v_i) \geq \frac{1}{d_i} \sum_{j=1}^{n} a_{ij} f(v_j), \; f(v_i) \geq \sum_{j=1}^{n} p_{ij} f(v_j), \text{ or } \; f(v_i) \geq \sum_{j=1}^{n} \frac{a_{ij}}{\sqrt{d_i}\sqrt{d_j}} f(v_j).$$

- One can also generalize various analytic concepts such as Green's functions, Green's identity, analytic functions, Cauchy-Riemann equations, ..., to the graph setting!

Functions Defined on a Graph ...

- A function $f \in C(V)$ is called harmonic if

$$Lf = 0,\ L_{\mathrm{rw}} f = 0,\ \text{or}\ \ L_{\mathrm{sym}} f = 0.$$

- A function $f \in C(V)$ is called superharmonic at $x \in V$ if

$$Lf(x) \geq 0,\ L_{\mathrm{rw}} f(x) \geq 0,\ \text{or}\ \ L_{\mathrm{sym}} f(x) \geq 0.$$

- These corresponds to:

$$f(v_i) \geq \frac{1}{d_i} \sum_{j=1}^{n} a_{ij} f(v_j),\ f(v_i) \geq \sum_{j=1}^{n} p_{ij} f(v_j),\ \text{or}\ \ f(v_i) \geq \sum_{j=1}^{n} \frac{a_{ij}}{\sqrt{d_i}\sqrt{d_j}} f(v_j).$$

- One can also generalize various analytic concepts such as Green's functions, Green's identity, analytic functions, Cauchy-Riemann equations, ..., to the graph setting!

Functions Defined on a Graph ...

- A function $f \in C(V)$ is called harmonic if

$$Lf = 0,\ L_{\mathrm{rw}} f = 0,\ \text{or}\ \ L_{\mathrm{sym}} f = 0.$$

- A function $f \in C(V)$ is called superharmonic at $x \in V$ if

$$Lf(x) \geq 0,\ L_{\mathrm{rw}} f(x) \geq 0,\ \text{or}\ \ L_{\mathrm{sym}} f(x) \geq 0.$$

- These corresponds to:

$$f(v_i) \geq \frac{1}{d_i} \sum_{j=1}^{n} a_{ij} f(v_j),\ f(v_i) \geq \sum_{j=1}^{n} p_{ij} f(v_j),\ \text{or}\ \ f(v_i) \geq \sum_{j=1}^{n} \frac{a_{ij}}{\sqrt{d_i}\sqrt{d_j}} f(v_j).$$

- One can also generalize various analytic concepts such as Green's functions, Green's identity, analytic functions, Cauchy-Riemann equations, ..., to the graph setting!

Derivatives and Green's Identity

Let $C(\boldsymbol{E}) := \{\varphi \text{ defined on } \boldsymbol{E} \mid \varphi(\bar{e}) = -\varphi(e), e \in \boldsymbol{E}\}$. For $f \in C(V)$, define the derivative $df \in C(\boldsymbol{E})$ of f as

$$df(e) = df([x,y]) := f(y) - f(x).$$

Derivatives and Green's Identity

Let $C(\boldsymbol{E}) := \{\varphi \text{ defined on } \boldsymbol{E} \mid \varphi(\bar{e}) = -\varphi(e), e \in \boldsymbol{E}\}$. For $f \in C(V)$, define the derivative $df \in C(\boldsymbol{E})$ of f as

$$df(e) = df([x,y]) := f(y) - f(x).$$

Theorem (The discrete version of Green's first identity, Dodziuk 1984)

$$\forall f_1, f_2 \in C_0(V), \langle df_1, df_2 \rangle = \langle L_{\mathrm{rw}} f_1, f_2 \rangle_{\#} = \langle L f_1, f_2 \rangle$$

Derivatives and Green's Identity

Let $C(\boldsymbol{E}) := \{\varphi \text{ defined on } \boldsymbol{E} \mid \varphi(\bar{e}) = -\varphi(e), e \in \boldsymbol{E}\}$. For $f \in C(V)$, define the derivative $df \in C(\boldsymbol{E})$ of f as

$$df(e) = df([x,y]) := f(y) - f(x).$$

Theorem (The discrete version of Green's first identity, Dodziuk 1984)

$$\forall f_1, f_2 \in C_0(V),\ \langle df_1, df_2 \rangle = \langle L_{\mathrm{rw}} f_1, f_2 \rangle_{\#} = \langle L f_1, f_2 \rangle$$

Corollary

L, L_{rw}, and L_{sym} are nonnegative operators, e.g.,

$$\langle L_{\mathrm{rw}} f, f \rangle_{\#} = \langle Lf, f \rangle = \langle df, df \rangle \geq 0.$$

The Minimum Principle

Theorem (The discrete version of the minimum principle)

Let $f \in C(V)$ be superharmonic at $x \in V$. If $f(x) \leq \min_{y \sim x} f(y)$, then $f(z) = f(x)$, $\forall z \sim x$.

The Minimum Principle

Theorem (The discrete version of the minimum principle)

Let $f \in C(V)$ be superharmonic at $x \in V$. If $f(x) \leq \min_{y \sim x} f(y)$, then $f(z) = f(x)$, $\forall z \sim x$.

Proof. From the superharmonicity of f at $x \in V$, we have

$$\frac{1}{d_x} \sum_{y \sim x} a_{xy} f(y) \leq f(x).$$

On the other hand, from the condition of this theorem, we have

$$\frac{1}{d_x} \sum_{y \sim x} a_{xy} f(y) \geq \frac{1}{d_x} \sum_{y \sim x} a_{xy} f(x) = f(x).$$

Hence, we must have $\frac{1}{d_x} \sum_{y \sim x} a_{xy} f(y) = f(x)$. But this can happen only if $f(z) = f(x)$, $\forall z \sim x$. □

Why Graph Laplacians?

- We already know that the Laplacian eigenvalues and eigenfunctions are extremely useful for general domains in $\mathbb{R}^d$, e.g., see my paper:
  - N. Saito: "Data analysis and representation using eigenfunctions of Laplacian on a general domain," *Applied & Computational Harmonic Analysis*, vol. 25, no. 1, pp. 68–97, 2008.
- The graph Laplacian *eigenvalues* reflect various intrinsic geometric and topological information about the graph including connectivity or the number of separated components; diameter; mean distance, ...
  - Fan Chung: *Spectral Graph Theory*, Amer. Math. Soc., 1997, says: *"This monograph is an intertwined tale of eigenvalues and their use in unlocking a thousand secrets about graphs."*
- Due to the time limitation, I will *not* be able to discuss the details on how the graph Laplacian *eigenvalues* reveal the geometric and topological information of the graph. For the details, please check the above book, the books listed in the beginning of this section, and
  - R. Merris: "Laplacian matrices of graphs: a survey," *Linear Algebra Appl.*, vol. 197/198, pp. 143–176, 1994.
  - N. Saito & E. Woei: "Analysis of neuronal dendrite patterns using eigenvalues of graph Laplacians," *Japan SIAM Lett.* vol. 1 pp. 13–16, 2009 (Invited paper).

Why Graph Laplacians?

- We already know that the Laplacian eigenvalues and eigenfunctions are extremely useful for general domains in $\mathbb{R}^d$, e.g., see my paper:
  - N. Saito: "Data analysis and representation using eigenfunctions of Laplacian on a general domain," *Applied & Computational Harmonic Analysis*, vol. 25, no. 1, pp. 68–97, 2008.
- The graph Laplacian *eigenvalues* reflect various intrinsic geometric and topological information about the graph including connectivity or the number of separated components; diameter; mean distance, ...
  - Fan Chung: *Spectral Graph Theory*, Amer. Math. Soc., 1997, says: *"This monograph is an intertwined tale of eigenvalues and their use in unlocking a thousand secrets about graphs."*
- Due to the time limitation, I will *not* be able to discuss the details on how the graph Laplacian *eigenvalues* reveal the geometric and topological information of the graph. For the details, please check the above book, the books listed in the beginning of this section, and
  - R. Merris: "Laplacian matrices of graphs: a survey," *Linear Algebra Appl.*, vol. 197/198, pp. 143–176, 1994.
  - N. Saito & E. Woei: "Analysis of neuronal dendrite patterns using eigenvalues of graph Laplacians," *Japan SIAM Lett.* vol. 1 pp. 13–16, 2009 (Invited paper).

Why Graph Laplacians?

- We already know that the Laplacian eigenvalues and eigenfunctions are extremely useful for general domains in $\mathbb{R}^d$, e.g., see my paper:
  - N. Saito: "Data analysis and representation using eigenfunctions of Laplacian on a general domain," *Applied & Computational Harmonic Analysis*, vol. 25, no. 1, pp. 68–97, 2008.
- The graph Laplacian *eigenvalues* reflect various intrinsic geometric and topological information about the graph including connectivity or the number of separated components; diameter; mean distance, ...
  - Fan Chung: *Spectral Graph Theory*, Amer. Math. Soc., 1997, says: *"This monograph is an intertwined tale of eigenvalues and their use in unlocking a thousand secrets about graphs."*
- Due to the time limitation, I will *not* be able to discuss the details on how the graph Laplacian *eigenvalues* reveal the geometric and topological information of the graph. For the details, please check the above book, the books listed in the beginning of this section, and
  - R. Merris: "Laplacian matrices of graphs: a survey," *Linear Algebra Appl.*, vol. 197/198, pp. 143–176, 1994.
  - N. Saito & E. Woei: "Analysis of neuronal dendrite patterns using eigenvalues of graph Laplacians," *Japan SIAM Lett.* vol. 1 pp. 13–16, 2009 (Invited paper).

Why Graph Laplacians? ...

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

Why Graph Laplacians? ...

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

Why Graph Laplacians? ...

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

Why Graph Laplacians? ...

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

Why Graph Laplacians? ...

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

Why Graph Laplacians? ...

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

Why Graph Laplacians? ...

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

A Simple Yet Important Example: A Path Graph

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

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).

- $\lambda_k = 2 - 2\cos(\pi k/n) = 4\sin^2(\pi k/2n),\ k = 0, 1, \ldots, n-1.$
- $\boldsymbol{\phi}_k(\ell) = \cos\left(\pi k\left(\ell + \frac{1}{2}\right)/n\right),\ k, \ell = 0, 1, \ldots, n-1.$
- In this simple case, λ (eigenvalue) is a monotonic function w.r.t. the frequency, which is the eigenvalue index k. However, in general, the notion of frequency is not well defined.

Outline

A Brief Review of Graph Laplacian Eigenvalues

- In this review part, we only consider undirected and unweighted graphs and their unnormalized Laplacians $L(G) = D(G) - A(G)$. Let $|V(G)| = n$, $|E(G)| = m$.
- It is a good exercise to see how the statements change for the *normalized* or *symmetrically-normalized* graph Laplacians.
- Can show that $L(G)$ is positive semi-definite.
- Hence, we can *sort* the eigenvalues of $L(G)$ as $0 = \lambda_0(G) \leq \lambda_1(G) \leq \cdots \leq \lambda_{n-1}(G)$ and denote the set of these eigenvalue by $\Lambda(G)$.
- $m_G(\lambda) :=$ the multiplicity of λ.
- Let $I \subset \mathbb{R}$ be an interval of the real line. Then define $m_G(I) := \#\{\lambda_k(G) \in I\}$.

A Brief Review of Graph Laplacian Eigenvalues

- In this review part, we only consider undirected and unweighted graphs and their unnormalized Laplacians $L(G) = D(G) - A(G)$. Let $|V(G)| = n$, $|E(G)| = m$.
- It is a good exercise to see how the statements change for the *normalized* or *symmetrically-normalized* graph Laplacians.
- Can show that $L(G)$ is positive semi-definite.
- Hence, we can *sort* the eigenvalues of $L(G)$ as $0 = \lambda_0(G) \leq \lambda_1(G) \leq \cdots \leq \lambda_{n-1}(G)$ and denote the set of these eigenvalue by $\Lambda(G)$.
- $m_G(\lambda) :=$ the multiplicity of λ.
- Let $I \subset \mathbb{R}$ be an interval of the real line. Then define $m_G(I) := \#\{\lambda_k(G) \in I\}$.

A Brief Review of Graph Laplacian Eigenvalues

- In this review part, we only consider undirected and unweighted graphs and their unnormalized Laplacians $L(G) = D(G) - A(G)$. Let $|V(G)| = n$, $|E(G)| = m$.
- It is a good exercise to see how the statements change for the *normalized* or *symmetrically-normalized* graph Laplacians.
- Can show that $L(G)$ is positive semi-definite.
- Hence, we can *sort* the eigenvalues of $L(G)$ as $0 = \lambda_0(G) \leq \lambda_1(G) \leq \cdots \leq \lambda_{n-1}(G)$ and denote the set of these eigenvalue by $\Lambda(G)$.
- $m_G(\lambda) :=$ the multiplicity of λ.
- Let $I \subset \mathbb{R}$ be an interval of the real line. Then define $m_G(I) := \#\{\lambda_k(G) \in I\}$.

A Brief Review of Graph Laplacian Eigenvalues

- In this review part, we only consider undirected and unweighted graphs and their unnormalized Laplacians $L(G) = D(G) - A(G)$. Let $|V(G)| = n$, $|E(G)| = m$.
- It is a good exercise to see how the statements change for the *normalized* or *symmetrically-normalized* graph Laplacians.
- Can show that $L(G)$ is positive semi-definite.
- Hence, we can *sort* the eigenvalues of $L(G)$ as $0 = \lambda_0(G) \leq \lambda_1(G) \leq \cdots \leq \lambda_{n-1}(G)$ and denote the set of these eigenvalue by $\Lambda(G)$.
- $m_G(\lambda) :=$ the multiplicity of λ.
- Let $I \subset \mathbb{R}$ be an interval of the real line. Then define $m_G(I) := \#\{\lambda_k(G) \in I\}$.

A Brief Review of Graph Laplacian Eigenvalues

- In this review part, we only consider undirected and unweighted graphs and their unnormalized Laplacians $L(G) = D(G) - A(G)$. Let $|V(G)| = n$, $|E(G)| = m$.
- It is a good exercise to see how the statements change for the *normalized* or *symmetrically-normalized* graph Laplacians.
- Can show that $L(G)$ is positive semi-definite.
- Hence, we can *sort* the eigenvalues of $L(G)$ as $0 = \lambda_0(G) \le \lambda_1(G) \le \cdots \le \lambda_{n-1}(G)$ and denote the set of these eigenvalue by $\Lambda(G)$.
- $m_G(\lambda) :=$ the multiplicity of λ.
- Let $I \subset \mathbb{R}$ be an interval of the real line. Then define $m_G(I) := \#\{\lambda_k(G) \in I\}$.

A Brief Review of Graph Laplacian Eigenvalues

- In this review part, we only consider undirected and unweighted graphs and their unnormalized Laplacians $L(G) = D(G) - A(G)$. Let $|V(G)| = n$, $|E(G)| = m$.
- It is a good exercise to see how the statements change for the *normalized* or *symmetrically-normalized* graph Laplacians.
- Can show that $L(G)$ is positive semi-definite.
- Hence, we can *sort* the eigenvalues of $L(G)$ as $0 = \lambda_0(G) \leq \lambda_1(G) \leq \cdots \leq \lambda_{n-1}(G)$ and denote the set of these eigenvalue by $\Lambda(G)$.
- $m_G(\lambda) :=$ the multiplicity of λ.
- Let $I \subset \mathbb{R}$ be an interval of the real line. Then define $m_G(I) := \#\{\lambda_k(G) \in I\}$.

A Brief Review of Graph Laplacian Eigenvalues ...

- Graph Laplacian matrices of the same graph are permutation-similar. In fact, graphs G_1 and G_2 are *isomorphic* iff there exists a permutation matrix Q such that
$$L(G_2) = Q^{\mathsf{T}} L(G_1) Q.$$
- $\operatorname{rank} L(G) = n - m_G(0)$ where $m_G(0)$ turns out to be the number of connected components of G. Easy to check that $L(G)$ becomes $m_G(0)$ diagonal blocks, and the eigenspace corresponding to the zero eigenvalues 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.
- 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 Eigenvalues ...

- Graph Laplacian matrices of the same graph are permutation-similar. In fact, graphs G_1 and G_2 are *isomorphic* iff there exists a permutation matrix Q such that
$$L(G_2) = Q^\top L(G_1) Q.$$
- $\operatorname{rank} L(G) = n - m_G(0)$ where $m_G(0)$ turns out to be the number of connected components of G. Easy to check that $L(G)$ becomes $m_G(0)$ diagonal blocks, and the eigenspace corresponding to the zero eigenvalues 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.
- 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 Eigenvalues ...

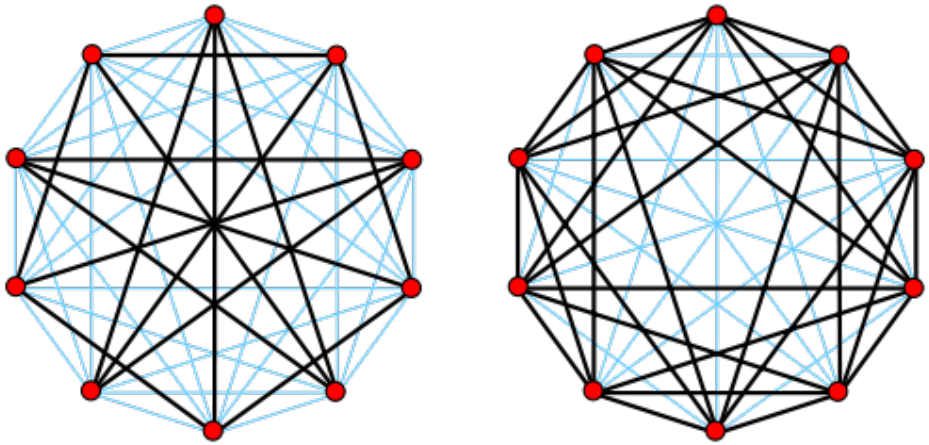
- Graph Laplacian matrices of the same graph are permutation-similar. In fact, graphs G_1 and G_2 are *isomorphic* iff there exists a permutation matrix Q such that
$$L(G_2) = Q^\top L(G_1) Q.$$
- $\operatorname{rank} L(G) = n - m_G(0)$ where $m_G(0)$ turns out to be the number of connected components of G. Easy to check that $L(G)$ becomes $m_G(0)$ diagonal blocks, and the eigenspace corresponding to the zero eigenvalues 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.
- 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 Eigenvalues ...

- Graph Laplacian matrices of the same graph are permutation-similar. In fact, graphs G_1 and G_2 are *isomorphic* iff there exists a permutation matrix Q such that
$$L(G_2) = Q^\top L(G_1) Q.$$
- $\operatorname{rank} L(G) = n - m_G(0)$ where $m_G(0)$ turns out to be the number of connected components of G. Easy to check that $L(G)$ becomes $m_G(0)$ diagonal blocks, and the eigenspace corresponding to the zero eigenvalues 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.
- 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 Eigenvalues . . .

- Denote the complement of G (in K_n) by G^c.



The Petersen graph and its complement in K_{10} (from Wikipedia)

- Then, we have

$$L(G) + L(G^c) = L(K_n) = nI_n - J_n,$$

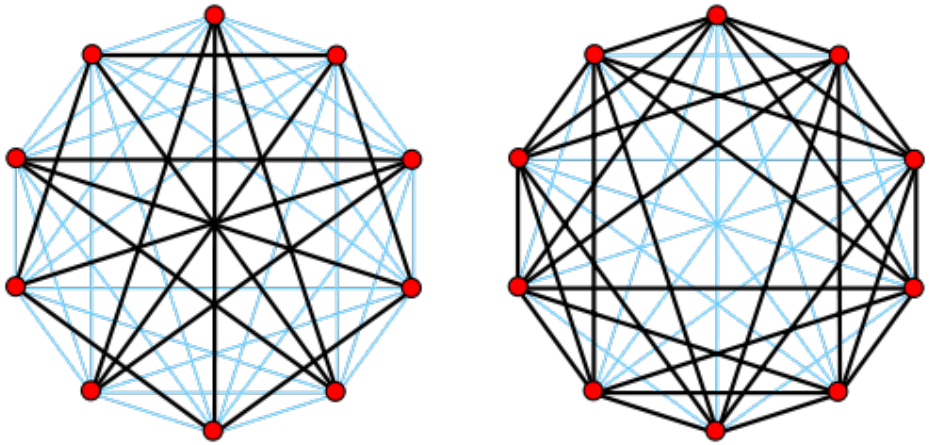
where J_n is the $n \times n$ matrix whose entries are all 1.

- We also have:

$$\Lambda(G^c) = \{0, n - \lambda_{n-1}(G), n - \lambda_{n-2}(G), \ldots, n - \lambda_1(G)\}.$$

A Brief Review of Graph Laplacian Eigenvalues . . .

- Denote the complement of G (in K_n) by G^c.



The Petersen graph and its complement in K_{10} (from Wikipedia)

- Then, we have

$$L(G) + L(G^c) = L(K_n) = nI_n - J_n,$$

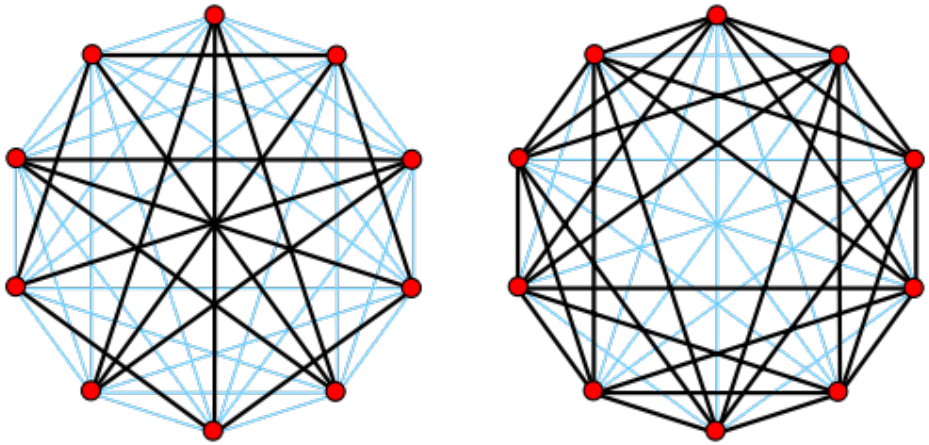
where J_n is the $n \times n$ matrix whose entries are all 1.

- We also have:

$$\Lambda(G^c) = \{0, n - \lambda_{n-1}(G), n - \lambda_{n-2}(G), \ldots, n - \lambda_1(G)\}.$$

A Brief Review of Graph Laplacian Eigenvalues . . .

- Denote the complement of G (in K_n) by G^c.



The Petersen graph and its complement in K_{10} (from Wikipedia)

- Then, we have

$$L(G) + L(G^c) = L(K_n) = nI_n - J_n,$$

where J_n is the $n \times n$ matrix whose entries are all 1.

- We also have:

$$\Lambda(G^c) = \{0, n - \lambda_{n-1}(G), n - \lambda_{n-2}(G), \ldots, n - \lambda_1(G)\}.$$

A Brief Review of Graph Laplacian Eigenvalues ...

- From the above, we can see that

$$\lambda_{\max}(G) = \lambda_{n-1}(G) \leq n,$$

and $m_G(n) = m_{G^c}(0) - 1$.

- On the other hand, Grone and Merris showed in 1994

$$\lambda_{\max}(G) = \lambda_{n-1}(G) \geq \max_{1 \leq j \leq n} d_j + 1.$$

- Let G be a connected graph and suppose $L(G)$ has exactly k distinct eigenvalues. Then

$$\mathrm{diam}(G) \leq k - 1.$$

A Brief Review of Graph Laplacian Eigenvalues ...

- From the above, we can see that

$$\lambda_{\max}(G) = \lambda_{n-1}(G) \leq n,$$

  and $m_G(n) = m_{G^c}(0) - 1$.
- On the other hand, Grone and Merris showed in 1994

$$\lambda_{\max}(G) = \lambda_{n-1}(G) \geq \max_{1 \leq j \leq n} d_j + 1.$$

- Let G be a connected graph and suppose $L(G)$ has exactly k distinct eigenvalues. Then

$$\mathrm{diam}(G) \leq k - 1.$$

A Brief Review of Graph Laplacian Eigenvalues ...

- From the above, we can see that

$$\lambda_{\max}(G) = \lambda_{n-1}(G) \leq n,$$

  and $m_G(n) = m_{G^c}(0) - 1$.
- On the other hand, Grone and Merris showed in 1994

$$\lambda_{\max}(G) = \lambda_{n-1}(G) \geq \max_{1 \leq j \leq n} d_j + 1.$$

- Let G be a connected graph and suppose $L(G)$ has exactly k distinct eigenvalues. Then

$$\mathrm{diam}(G) \leq k - 1.$$

Outline

Basic Properties of GL Eigenfunctions

- If $G = (V, E)$, $|V| = n$, is connected, then $\lambda_0 = 0$, $a(G) = \lambda_1 > 0$.
- We already know that the eigenfunction corresponding to $\lambda_0 = 0$ is $\boldsymbol{\phi}_0 = \mathbf{1}_n$.
- Hence, $\boldsymbol{\phi}_j$ corresponding to $\lambda_j > 0$, $j = 1, \ldots, n-1$, must be orthogonal to $\mathbf{1}_n$: $\sum_{x \in V} \boldsymbol{\phi}_j(x) = 0$, i.e., it must *oscillate*.
- If $\boldsymbol{\phi}(x) = 0$, then $(L\boldsymbol{\phi})(x) = \lambda \boldsymbol{\phi}(x) = 0$. Hence, $\sum_{y \sim x} L_{xy} \boldsymbol{\phi}(y) = 0$.

Theorem (Grover (1990); Gladwell & Zhu (2002))

An eigenfunction of $L(G)$ cannot have a nonnegative local minimum or a nonpositive local maximum.

Proof. Suppose $\boldsymbol{\phi}(x)$ is a local minimum of $\boldsymbol{\phi}$ with $\boldsymbol{\phi}(x) \geq 0$. Then, $\forall y \sim x$, $\boldsymbol{\phi}(x) - \boldsymbol{\phi}(y) < 0$. Now, recall $L\boldsymbol{\phi}(x) = \sum_{y \sim x} a_{xy}(\boldsymbol{\phi}(x) - \boldsymbol{\phi}(y)) = \lambda \boldsymbol{\phi}(x) \geq 0$ where $a_{xy} \geq 0$ is the xy-th entry of the adjacency matrix $A(G)$. These contradicts each other. □

Basic Properties of GL Eigenfunctions

- If $G = (V, E)$, $|V| = n$, is connected, then $\lambda_0 = 0$, $a(G) = \lambda_1 > 0$.
- We already know that the eigenfunction corresponding to $\lambda_0 = 0$ is $\boldsymbol{\phi}_0 = \mathbf{1}_n$.
- Hence, $\boldsymbol{\phi}_j$ corresponding to $\lambda_j > 0$, $j = 1, \ldots, n-1$, must be orthogonal to $\mathbf{1}_n$: $\sum_{x \in V} \boldsymbol{\phi}_j(x) = 0$, i.e., it must *oscillate*.
- If $\boldsymbol{\phi}(x) = 0$, then $(L\boldsymbol{\phi})(x) = \lambda \boldsymbol{\phi}(x) = 0$. Hence, $\sum_{y \sim x} L_{xy} \boldsymbol{\phi}(y) = 0$.

Theorem (Grover (1990); Gladwell & Zhu (2002))

An eigenfunction of $L(G)$ cannot have a nonnegative local minimum or a nonpositive local maximum.

Proof. Suppose $\boldsymbol{\phi}(x)$ is a local minimum of $\boldsymbol{\phi}$ with $\boldsymbol{\phi}(x) \geq 0$. Then, $\forall y \sim x$, $\boldsymbol{\phi}(x) - \boldsymbol{\phi}(y) < 0$. Now, recall $L\boldsymbol{\phi}(x) = \sum_{y \sim x} a_{xy}(\boldsymbol{\phi}(x) - \boldsymbol{\phi}(y)) = \lambda \boldsymbol{\phi}(x) \geq 0$ where $a_{xy} \geq 0$ is the xy-th entry of the adjacency matrix $A(G)$. These contradicts each other. □

Basic Properties of GL Eigenfunctions

- If $G = (V, E)$, $|V| = n$, is connected, then $\lambda_0 = 0$, $a(G) = \lambda_1 > 0$.
- We already know that the eigenfunction corresponding to $\lambda_0 = 0$ is $\boldsymbol{\phi}_0 = \mathbf{1}_n$.
- Hence, $\boldsymbol{\phi}_j$ corresponding to $\lambda_j > 0$, $j = 1, \ldots, n-1$, must be orthogonal to $\mathbf{1}_n$: $\sum_{x \in V} \boldsymbol{\phi}_j(x) = 0$, i.e., it must *oscillate*.
- If $\boldsymbol{\phi}(x) = 0$, then $(L\boldsymbol{\phi})(x) = \lambda \boldsymbol{\phi}(x) = 0$. Hence, $\sum_{y \sim x} L_{xy} \boldsymbol{\phi}(y) = 0$.

Theorem (Grover (1990); Gladwell & Zhu (2002))

An eigenfunction of $L(G)$ cannot have a nonnegative local minimum or a nonpositive local maximum.

Proof. Suppose $\boldsymbol{\phi}(x)$ is a local minimum of $\boldsymbol{\phi}$ with $\boldsymbol{\phi}(x) \geq 0$. Then, $\forall y \sim x$, $\boldsymbol{\phi}(x) - \boldsymbol{\phi}(y) < 0$. Now, recall $L\boldsymbol{\phi}(x) = \sum_{y \sim x} a_{xy}(\boldsymbol{\phi}(x) - \boldsymbol{\phi}(y)) = \lambda \boldsymbol{\phi}(x) \geq 0$ where $a_{xy} \geq 0$ is the xy-th entry of the adjacency matrix $A(G)$. These contradicts each other. □

Basic Properties of GL Eigenfunctions

- If $G = (V, E)$, $|V| = n$, is connected, then $\lambda_0 = 0$, $a(G) = \lambda_1 > 0$.
- We already know that the eigenfunction corresponding to $\lambda_0 = 0$ is $\boldsymbol{\phi}_0 = \mathbf{1}_n$.
- Hence, $\boldsymbol{\phi}_j$ corresponding to $\lambda_j > 0$, $j = 1, \ldots, n-1$, must be orthogonal to $\mathbf{1}_n$: $\sum_{x \in V} \boldsymbol{\phi}_j(x) = 0$, i.e., it must *oscillate*.
- If $\boldsymbol{\phi}(x) = 0$, then $(L\boldsymbol{\phi})(x) = \lambda \boldsymbol{\phi}(x) = 0$. Hence, $\sum_{y \sim x} L_{xy} \boldsymbol{\phi}(y) = 0$.

Theorem (Grover (1990); Gladwell & Zhu (2002))

An eigenfunction of $L(G)$ cannot have a nonnegative local minimum or a nonpositive local maximum.

Proof. Suppose $\boldsymbol{\phi}(x)$ is a local minimum of $\boldsymbol{\phi}$ with $\boldsymbol{\phi}(x) \geq 0$. Then, $\forall y \sim x$, $\boldsymbol{\phi}(x) - \boldsymbol{\phi}(y) < 0$. Now, recall $L\boldsymbol{\phi}(x) = \sum_{y \sim x} a_{xy}(\boldsymbol{\phi}(x) - \boldsymbol{\phi}(y)) = \lambda \boldsymbol{\phi}(x) \geq 0$ where $a_{xy} \geq 0$ is the xy-th entry of the adjacency matrix $A(G)$. These contradicts each other. □

Basic Properties of GL Eigenfunctions

- If $G = (V, E)$, $|V| = n$, is connected, then $\lambda_0 = 0$, $a(G) = \lambda_1 > 0$.
- We already know that the eigenfunction corresponding to $\lambda_0 = 0$ is $\boldsymbol{\phi}_0 = \mathbf{1}_n$.
- Hence, $\boldsymbol{\phi}_j$ corresponding to $\lambda_j > 0$, $j = 1, \ldots, n-1$, must be orthogonal to $\mathbf{1}_n$: $\sum_{x \in V} \boldsymbol{\phi}_j(x) = 0$, i.e., it must *oscillate*.
- If $\boldsymbol{\phi}(x) = 0$, then $(L\boldsymbol{\phi})(x) = \lambda \boldsymbol{\phi}(x) = 0$. Hence, $\sum_{y \sim x} L_{xy} \boldsymbol{\phi}(y) = 0$.

Theorem (Grover (1990); Gladwell & Zhu (2002))

An eigenfunction of $L(G)$ cannot have a nonnegative local minimum or a nonpositive local maximum.

Proof. Suppose $\boldsymbol{\phi}(x)$ is a local minimum of $\boldsymbol{\phi}$ with $\boldsymbol{\phi}(x) \geq 0$. Then, $\forall y \sim x$, $\boldsymbol{\phi}(x) - \boldsymbol{\phi}(y) < 0$. Now, recall $L\boldsymbol{\phi}(x) = \sum_{y \sim x} a_{xy}(\boldsymbol{\phi}(x) - \boldsymbol{\phi}(y)) = \lambda \boldsymbol{\phi}(x) \geq 0$ where $a_{xy} \geq 0$ is the xy-th entry of the adjacency matrix $A(G)$. These contradicts each other. □

Basic Properties of GL Eigenfunctions

- If $G = (V, E)$, $|V| = n$, is connected, then $\lambda_0 = 0$, $a(G) = \lambda_1 > 0$.
- We already know that the eigenfunction corresponding to $\lambda_0 = 0$ is $\boldsymbol{\phi}_0 = \mathbf{1}_n$.
- Hence, $\boldsymbol{\phi}_j$ corresponding to $\lambda_j > 0$, $j = 1, \ldots, n-1$, must be orthogonal to $\mathbf{1}_n$: $\sum_{x \in V} \boldsymbol{\phi}_j(x) = 0$, i.e., it must *oscillate*.
- If $\boldsymbol{\phi}(x) = 0$, then $(L\boldsymbol{\phi})(x) = \lambda \boldsymbol{\phi}(x) = 0$. Hence, $\sum_{y \sim x} L_{xy} \boldsymbol{\phi}(y) = 0$.

Theorem (Grover (1990); Gladwell & Zhu (2002))

An eigenfunction of $L(G)$ cannot have a nonnegative local minimum or a nonpositive local maximum.

Proof. Suppose $\boldsymbol{\phi}(x)$ is a local minimum of $\boldsymbol{\phi}$ with $\boldsymbol{\phi}(x) \geq 0$. Then, $\forall y \sim x$, $\boldsymbol{\phi}(x) - \boldsymbol{\phi}(y) < 0$. Now, recall $L\boldsymbol{\phi}(x) = \sum_{y \sim x} a_{xy}(\boldsymbol{\phi}(x) - \boldsymbol{\phi}(y)) = \lambda \boldsymbol{\phi}(x) \geq 0$ where $a_{xy} \geq 0$ is the xy-th entry of the adjacency matrix $A(G)$. These contradicts each other. □

Basic Properties of *Unweighted* GL Eigenfunctions

Theorem (Merris (1998))

If $0 \lneqq \lambda < n$ is an eigenvalue of $L(G)$, then any eigenfunction affording λ takes the value 0 on every vertex of degree $n-1$.

Proof. Let $v \in V$ be a vertex with $d(v) = n-1$. Then, $L\boldsymbol{\phi}(v) = (n-1)\boldsymbol{\phi}(v) - \sum_{u \neq v} \boldsymbol{\phi}(u) = \lambda \boldsymbol{\phi}(v)$. But, $\boldsymbol{\phi} \perp \mathbf{1}_n$, so $\sum_{u \neq v} \boldsymbol{\phi}(u) = -\boldsymbol{\phi}(v)$. This leads to: $n\boldsymbol{\phi}(v) = \lambda \boldsymbol{\phi}(v)$. Since $0 \lneqq \lambda \lneqq n$, we must have $\boldsymbol{\phi}(v) = 0$. □

Theorem (Merris (1998))

Let $(\lambda, \boldsymbol{\phi})$ be an eigenpair of $L(G)$. If $\boldsymbol{\phi}(u) = \boldsymbol{\phi}(v)$, then $(\lambda, \boldsymbol{\phi})$ is also an eigenpair of $L(G')$ where G' is the graph obtained from G by either deleting or adding the edge $e = (u, v)$ depending on whether or not $e \in E(G)$.

Basic Properties of *Unweighted* GL Eigenfunctions

Theorem (Merris (1998))

If $0 \lneqq \lambda < n$ is an eigenvalue of $L(G)$, then any eigenfunction affording λ takes the value 0 on every vertex of degree $n-1$.

Proof. Let $v \in V$ be a vertex with $d(v) = n-1$. Then, $L\boldsymbol{\phi}(v) = (n-1)\boldsymbol{\phi}(v) - \sum_{u \neq v} \boldsymbol{\phi}(u) = \lambda \boldsymbol{\phi}(v)$. But, $\boldsymbol{\phi} \perp \mathbf{1}_n$, so $\sum_{u \neq v} \boldsymbol{\phi}(u) = -\boldsymbol{\phi}(v)$. This leads to: $n\boldsymbol{\phi}(v) = \lambda \boldsymbol{\phi}(v)$. Since $0 \lneqq \lambda \lneqq n$, we must have $\boldsymbol{\phi}(v) = 0$. □

Theorem (Merris (1998))

Let $(\lambda, \boldsymbol{\phi})$ be an eigenpair of $L(G)$. If $\boldsymbol{\phi}(u) = \boldsymbol{\phi}(v)$, then $(\lambda, \boldsymbol{\phi})$ is also an eigenpair of $L(G')$ where G' is the graph obtained from G by either deleting or adding the edge $e = (u, v)$ depending on whether or not $e \in E(G)$.

Basic Properties of *Unweighted* GL Eigenfunctions

Theorem (Merris (1998))

If $0 \lneqq \lambda < n$ is an eigenvalue of $L(G)$, then any eigenfunction affording λ takes the value 0 on every vertex of degree $n-1$.

Proof. Let $v \in V$ be a vertex with $d(v) = n-1$. Then, $L\boldsymbol{\phi}(v) = (n-1)\boldsymbol{\phi}(v) - \sum_{u \neq v} \boldsymbol{\phi}(u) = \lambda \boldsymbol{\phi}(v)$. But, $\boldsymbol{\phi} \perp \mathbf{1}_n$, so $\sum_{u \neq v} \boldsymbol{\phi}(u) = -\boldsymbol{\phi}(v)$. This leads to: $n\boldsymbol{\phi}(v) = \lambda \boldsymbol{\phi}(v)$. Since $0 \lneqq \lambda \lneqq n$, we must have $\boldsymbol{\phi}(v) = 0$. □

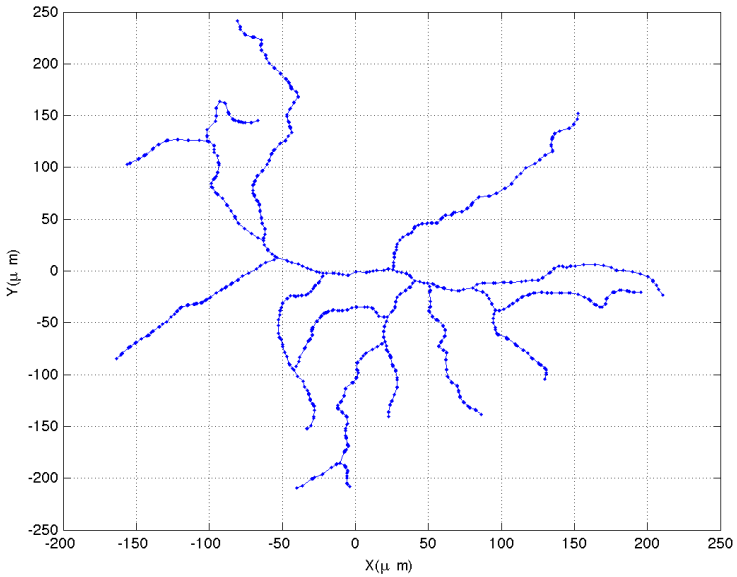
Theorem (Merris (1998))

Let $(\lambda, \boldsymbol{\phi})$ be an eigenpair of $L(G)$. If $\boldsymbol{\phi}(u) = \boldsymbol{\phi}(v)$, then $(\lambda, \boldsymbol{\phi})$ is also an eigenpair of $L(G')$ where G' is the graph obtained from G by either deleting or adding the edge $e = (u, v)$ depending on whether or not $e \in E(G)$.

Outline

A Peculiar Phase Transition Phenomenon

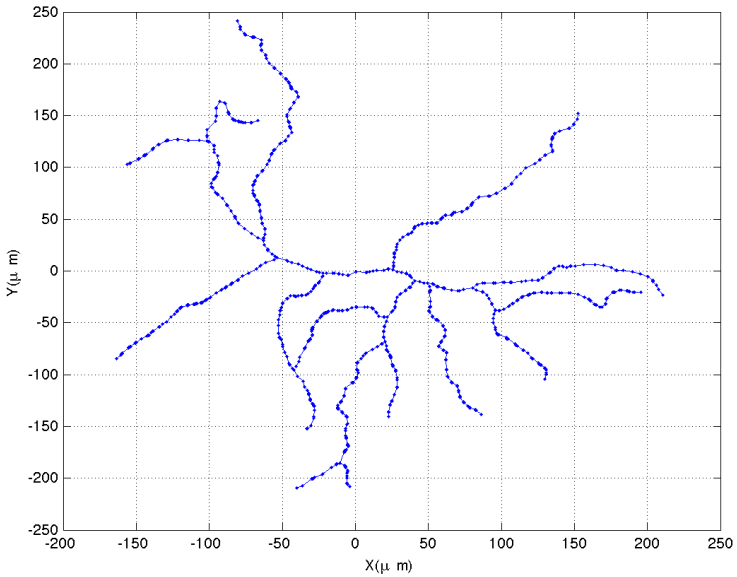
We observed an interesting phase transition phenomenon on the behavior of the eigenvalues of *graph Laplacians* defined on dendritic trees.



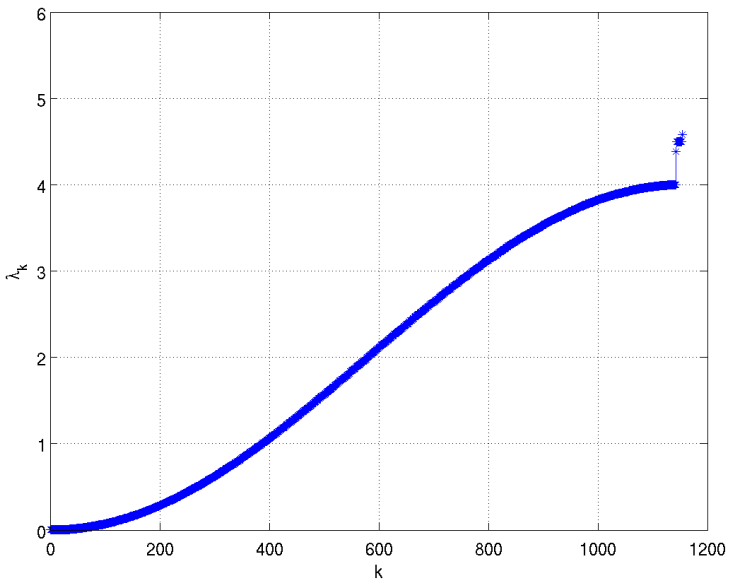
(a) RGC #100

A Peculiar Phase Transition Phenomenon

We observed an interesting phase transition phenomenon on the behavior of the eigenvalues of *graph Laplacians* defined on dendritic trees.



(a) RGC #100



(b) Eigenvalues of RGC #100

A Peculiar Phase Transition Phenomenon ...

We have observed that this value 4 is critical since:

- the eigenfunctions corresponding to the eigenvalues below 4 are *semi-global oscillations* (like *Fourier cosines/sines*) over the entire dendrites or one of the dendrite arbors;
- those corresponding to the eigenvalues above 4 are much more *localized* (like *wavelets*) around *junctions/bifurcation vertices*.

A Peculiar Phase Transition Phenomenon ...

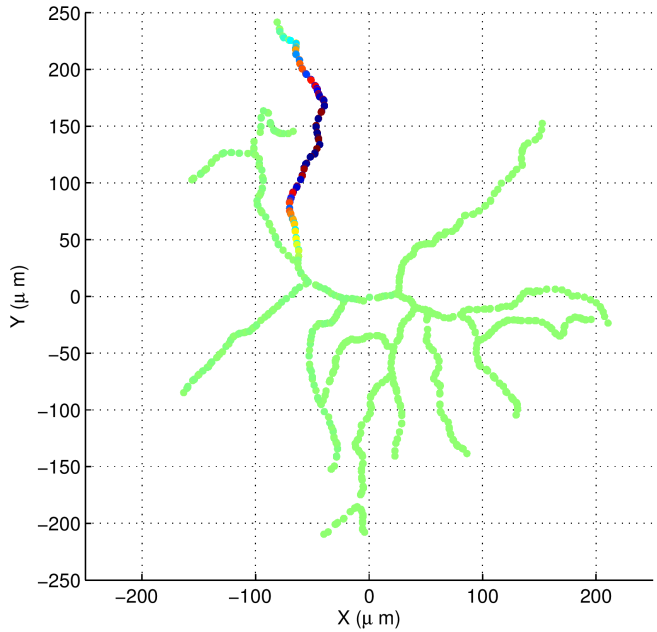
We have observed that this value 4 is critical since:

- the eigenfunctions corresponding to the eigenvalues below 4 are *semi-global oscillations* (like *Fourier cosines/sines*) over the entire dendrites or one of the dendrite arbors;
- those corresponding to the eigenvalues above 4 are much more *localized* (like *wavelets*) around *junctions/bifurcation vertices*.

A Peculiar Phase Transition Phenomenon ...

We have observed that this value 4 is critical since:

- the eigenfunctions corresponding to the eigenvalues below 4 are *semi-global oscillations* (like *Fourier cosines/sines*) over the entire dendrites or one of the dendrite arbors;
- those corresponding to the eigenvalues above 4 are much more *localized* (like *wavelets*) around *junctions/bifurcation vertices*.

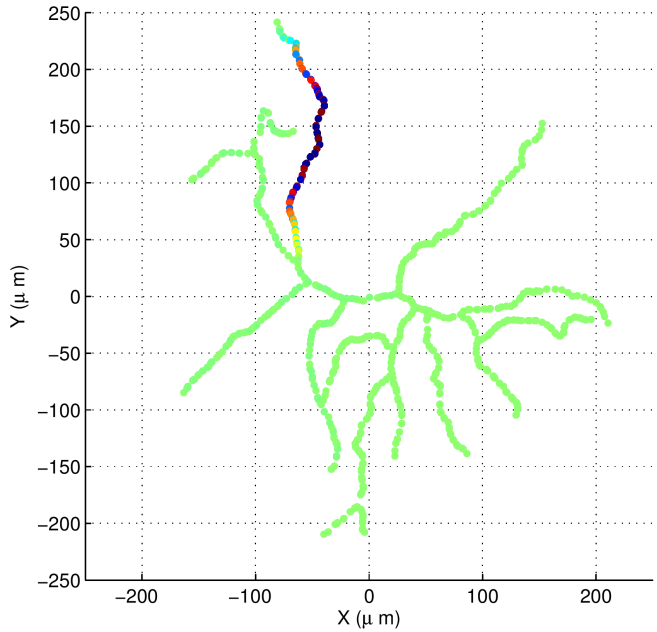


(a) RGC #100; $\lambda_{1141} = 3.9994$

A Peculiar Phase Transition Phenomenon ...

We have observed that this value 4 is critical since:

- the eigenfunctions corresponding to the eigenvalues below 4 are *semi-global oscillations* (like *Fourier cosines/sines*) over the entire dendrites or one of the dendrite arbors;
- those corresponding to the eigenvalues above 4 are much more *localized* (like *wavelets*) around *junctions/bifurcation vertices*.

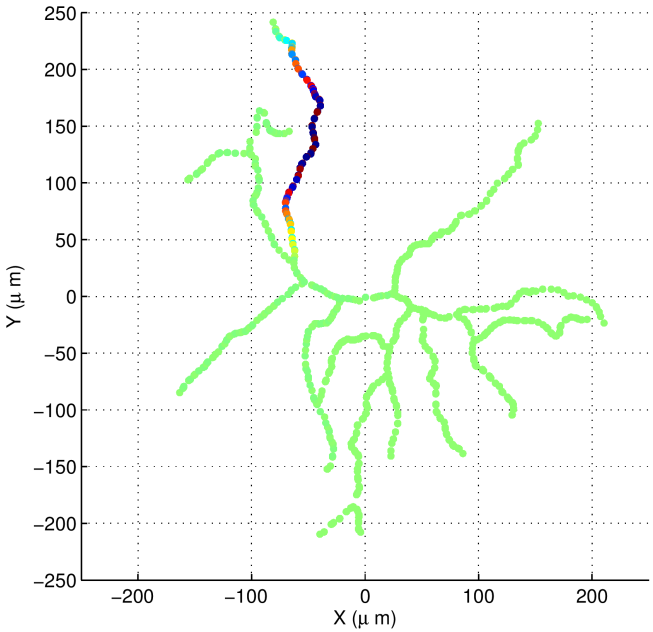


(a) RGC #100; $\lambda_{1141} = 3.9994$

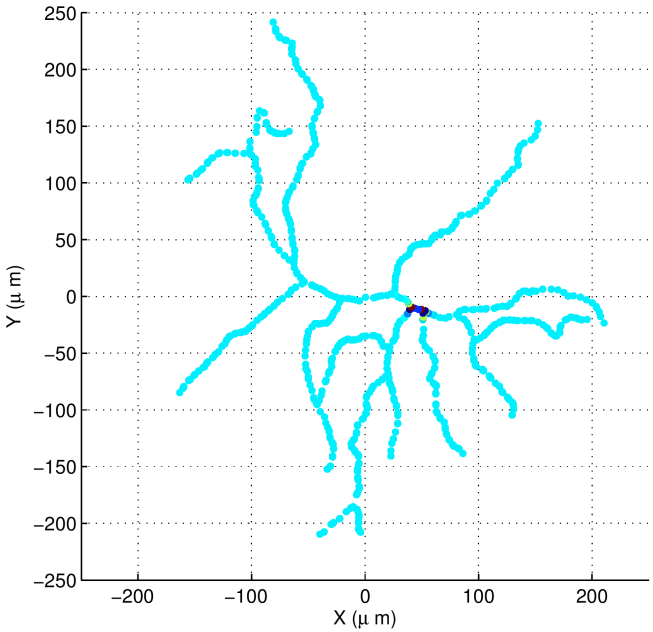
A Peculiar Phase Transition Phenomenon ...

We have observed that this value 4 is critical since:

- the eigenfunctions corresponding to the eigenvalues below 4 are *semi-global oscillations* (like *Fourier cosines/sines*) over the entire dendrites or one of the dendrite arbors;
- those corresponding to the eigenvalues above 4 are much more *localized* (like *wavelets*) around *junctions/bifurcation vertices*.



(a) RGC #100; $\lambda_{1141} = 3.9994$



(b) RGC #100; $\lambda_{1142} = 4.3829$

- We know why such localization/phase transition occurs $\Longrightarrow$ See our article for the detail: Y. Nakatsukasa, N. Saito, & E. Woei: "Mysteries around graph Laplacian eigenvalue 4," *Linear Algebra & Its Applications*, vol. 438, no. 8, pp. 3231–3246, 2013.
- Any physiological consequence? Importance of branching vertices?
- Many such eigenvector localization phenomena have been reported: Anderson localization, scars in quantum chaos, ...
- See also an interesting related work for more general setting and for application in numerical linear algebra: I. Krishtal, T. Strohmer, & T. Wertz: "Localization of matrix factorizations," *Foundations of Comp. Math.*, to appear, 2014.
- Our point is that *eigenvectors corresponding to high eigenvalues are quite sensitive to topology and geometry of the underlying domain and cannot really be viewed as high frequency oscillations unless the underlying graph is a simple unweighted path.*

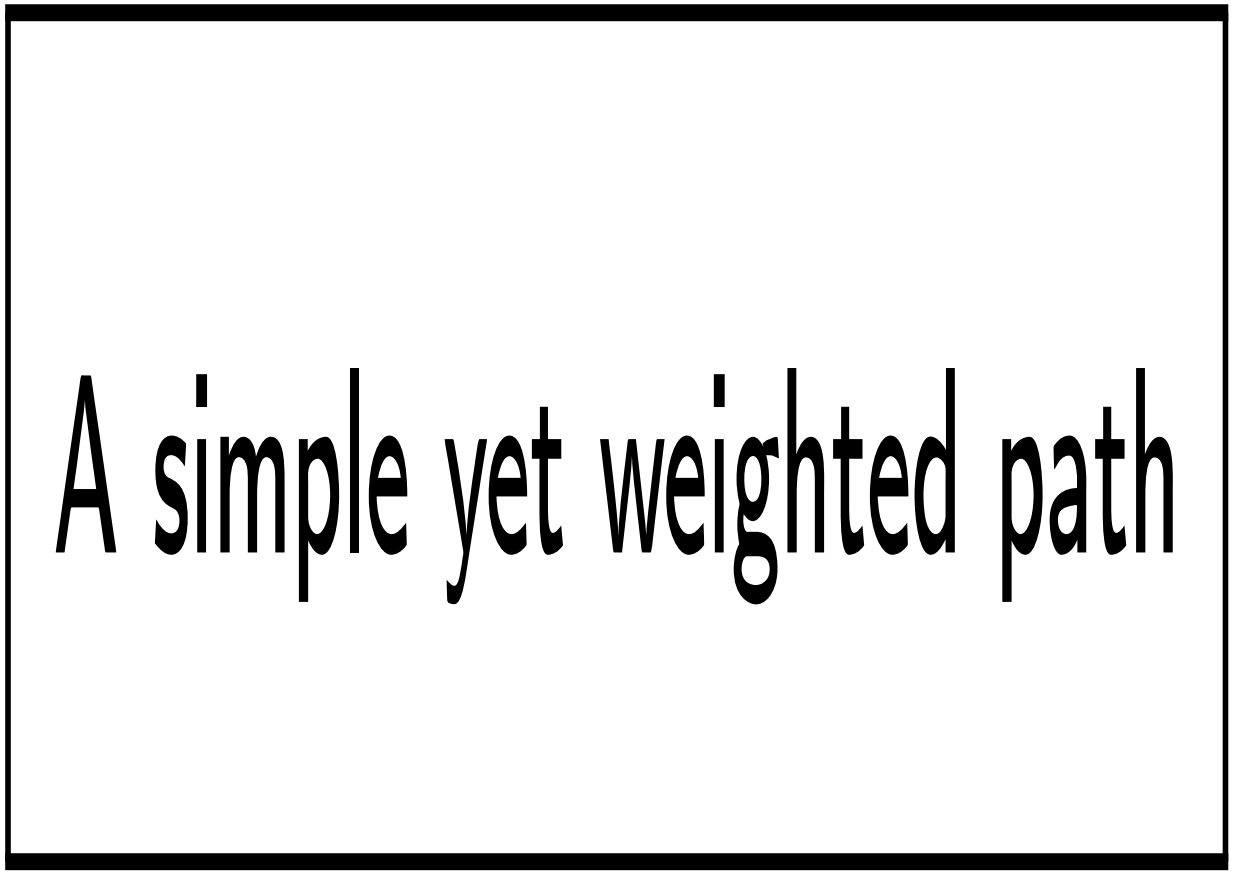
- We know why such localization/phase transition occurs $\Longrightarrow$ See our article for the detail: Y. Nakatsukasa, N. Saito, & E. Woei: "Mysteries around graph Laplacian eigenvalue 4," *Linear Algebra & Its Applications*, vol. 438, no. 8, pp. 3231–3246, 2013.
- Any physiological consequence? Importance of branching vertices?
- Many such eigenvector localization phenomena have been reported: Anderson localization, scars in quantum chaos, ...
- See also an interesting related work for more general setting and for application in numerical linear algebra: I. Krishtal, T. Strohmer, & T. Wertz: "Localization of matrix factorizations," *Foundations of Comp. Math.*, to appear, 2014.
- Our point is that *eigenvectors corresponding to high eigenvalues are quite sensitive to topology and geometry of the underlying domain and cannot really be viewed as high frequency oscillations unless the underlying graph is a simple unweighted path.*

- We know why such localization/phase transition occurs $\Longrightarrow$ See our article for the detail: Y. Nakatsukasa, N. Saito, & E. Woei: "Mysteries around graph Laplacian eigenvalue 4," *Linear Algebra & Its Applications*, vol. 438, no. 8, pp. 3231–3246, 2013.
- Any physiological consequence? Importance of branching vertices?
- Many such eigenvector localization phenomena have been reported: Anderson localization, scars in quantum chaos, ...
- See also an interesting related work for more general setting and for application in numerical linear algebra: I. Krishtal, T. Strohmer, & T. Wertz: "Localization of matrix factorizations," *Foundations of Comp. Math.*, to appear, 2014.
- Our point is that *eigenvectors corresponding to high eigenvalues are quite sensitive to topology and geometry of the underlying domain and cannot really be viewed as high frequency oscillations unless the underlying graph is a simple unweighted path.*

- We know why such localization/phase transition occurs $\Longrightarrow$ See our article for the detail: Y. Nakatsukasa, N. Saito, & E. Woei: "Mysteries around graph Laplacian eigenvalue 4," *Linear Algebra & Its Applications*, vol. 438, no. 8, pp. 3231–3246, 2013.
- Any physiological consequence? Importance of branching vertices?
- Many such eigenvector localization phenomena have been reported: Anderson localization, scars in quantum chaos, ...
- See also an interesting related work for more general setting and for application in numerical linear algebra: I. Krishtal, T. Strohmer, & T. Wertz: "Localization of matrix factorizations," *Foundations of Comp. Math.*, to appear, 2014.
- Our point is that *eigenvectors corresponding to high eigenvalues are quite sensitive to topology and geometry of the underlying domain and cannot really be viewed as high frequency oscillations unless the underlying graph is a simple unweighted path.*

- We know why such localization/phase transition occurs $\Longrightarrow$ See our article for the detail: Y. Nakatsukasa, N. Saito, & E. Woei: "Mysteries around graph Laplacian eigenvalue 4," *Linear Algebra & Its Applications*, vol. 438, no. 8, pp. 3231–3246, 2013.
- Any physiological consequence? Importance of branching vertices?
- Many such eigenvector localization phenomena have been reported: Anderson localization, scars in quantum chaos, ...
- See also an interesting related work for more general setting and for application in numerical linear algebra: I. Krishtal, T. Strohmer, & T. Wertz: "Localization of matrix factorizations," *Foundations of Comp. Math.*, to appear, 2014.
- Our point is that *eigenvectors corresponding to high eigenvalues are quite sensitive to topology and geometry of the underlying domain and cannot really be viewed as high frequency oscillations unless the underlying graph is a simple unweighted path.*

- Even a simple path, if edges are weighted, localization tends to occur.



Play/Pause

- We want to control such eigenvector localizations by ourselves rather than dictated by the topology and geometry of the graphs!
- This leads us to the development of the *multiscale basis dictionaries* on graphs.

Outline

Graph Partitioning via Spectral Clustering

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

Plan: minimize the RatioCut function[1],

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

where

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

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

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

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

Graph Partitioning via Spectral Clustering

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

Plan: minimize the RatioCut function[1],

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

where

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

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

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

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

Graph Partitioning via Spectral Clustering

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

Plan: minimize the RatioCut function[1],

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

where

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

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

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

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

Graph Partitioning via Spectral Clustering

Let us reformulate the RatioCut minimization problem.

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

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

2. The RatioCut problem can be reformulated as

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

Graph Partitioning via Spectral Clustering

Let us reformulate the RatioCut minimization problem.

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

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

2. The RatioCut problem can be reformulated as

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

Graph Partitioning via Spectral Clustering

Let us reformulate the RatioCut minimization problem.

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

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

2. The RatioCut problem can be reformulated as

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

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

Graph Partitioning via Spectral Clustering

Let us reformulate the RatioCut minimization problem.

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

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

2. The RatioCut problem can be reformulated as

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

Unfortunately, this problem is *NP hard* (i.e., at least as hard as solving any Nondeterministic Polynomial time problem) ...

Graph Partitioning via Spectral Clustering

Let us reformulate the RatioCut minimization problem.

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

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

2. The RatioCut problem can be reformulated as

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

Unfortunately, this problem is *NP hard* (i.e., at least as hard as solving any Nondeterministic Polynomial time problem) . . . **Relax!**

Graph Partitioning via Spectral Clustering

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

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

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

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

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

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

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

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

Graph Partitioning via Spectral Clustering

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

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

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

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

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

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

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

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

Graph Partitioning via Spectral Clustering

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

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

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

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

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

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

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

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

Graph Partitioning via Spectral Clustering

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

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

- By the Rayleigh-Ritz Theorem, the solution is given by $\boldsymbol{\phi}_1$ (scaled as necessary), where $\boldsymbol{\phi}_1$ is the eigenvector corresponding to the second smallest eigenvalue of L.
- $\boldsymbol{\phi}_1$ is known as the Fiedler vector and is often used to partition a graph into two subsets.
- von Luxburg recommends the use of the *random-walk* version of the Laplacian matrix, $L_{\text{rw}} := I - D^{-1}W$, over the usual Laplacian matrix L, which leads to the *NCut* and the generalized eigenvalue problem: $L\boldsymbol{\phi} = \lambda D \boldsymbol{\phi}$.

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

Graph Partitioning via Spectral Clustering

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

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

- By the Rayleigh-Ritz Theorem, the solution is given by $\boldsymbol{\phi}_1$ (scaled as necessary), where $\boldsymbol{\phi}_1$ is the eigenvector corresponding to the second smallest eigenvalue of L.
- $\boldsymbol{\phi}_1$ is known as the Fiedler vector and is often used to partition a graph into two subsets.
- von Luxburg recommends the use of the *random-walk* version of the Laplacian matrix, $L_{\text{rw}} := I - D^{-1}W$, over the usual Laplacian matrix L, which leads to the *NCut* and the generalized eigenvalue problem: $L\boldsymbol{\phi} = \lambda D \boldsymbol{\phi}$.

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

Graph Partitioning via Spectral Clustering

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

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

- By the Rayleigh-Ritz Theorem, the solution is given by $\boldsymbol{\phi}_1$ (scaled as necessary), where $\boldsymbol{\phi}_1$ is the eigenvector corresponding to the second smallest eigenvalue of L.
- $\boldsymbol{\phi}_1$ is known as the Fiedler vector and is often used to partition a graph into two subsets.
- von Luxburg recommends the use of the *random-walk* version of the Laplacian matrix, $L_{\text{rw}} := I - D^{-1}W$, over the usual Laplacian matrix L, which leads to the *NCut* and the generalized eigenvalue problem: $L\boldsymbol{\phi} = \lambda D \boldsymbol{\phi}$.

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

Graph Partitioning via Spectral Clustering

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

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

- By the Rayleigh-Ritz Theorem, the solution is given by $\boldsymbol{\phi}_1$ (scaled as necessary), where $\boldsymbol{\phi}_1$ is the eigenvector corresponding to the second smallest eigenvalue of L.
- $\boldsymbol{\phi}_1$ is known as the Fiedler vector and is often used to partition a graph into two subsets.
- von Luxburg recommends the use of the *random-walk* version of the Laplacian matrix, $L_{\text{rw}} := I - D^{-1}W$, over the usual Laplacian matrix L, which leads to the *NCut* and the generalized eigenvalue problem: $L\boldsymbol{\phi} = \lambda D \boldsymbol{\phi}$.

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

Graph Partitioning via Spectral Clustering

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

Graph Partitioning via Spectral Clustering

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

Definition (Weak Nodal Domain)

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

Graph Partitioning via Spectral Clustering

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

Definition (Weak Nodal Domain)

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

Corollary (Fiedler (1975))

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

Example of Graph Partitioning

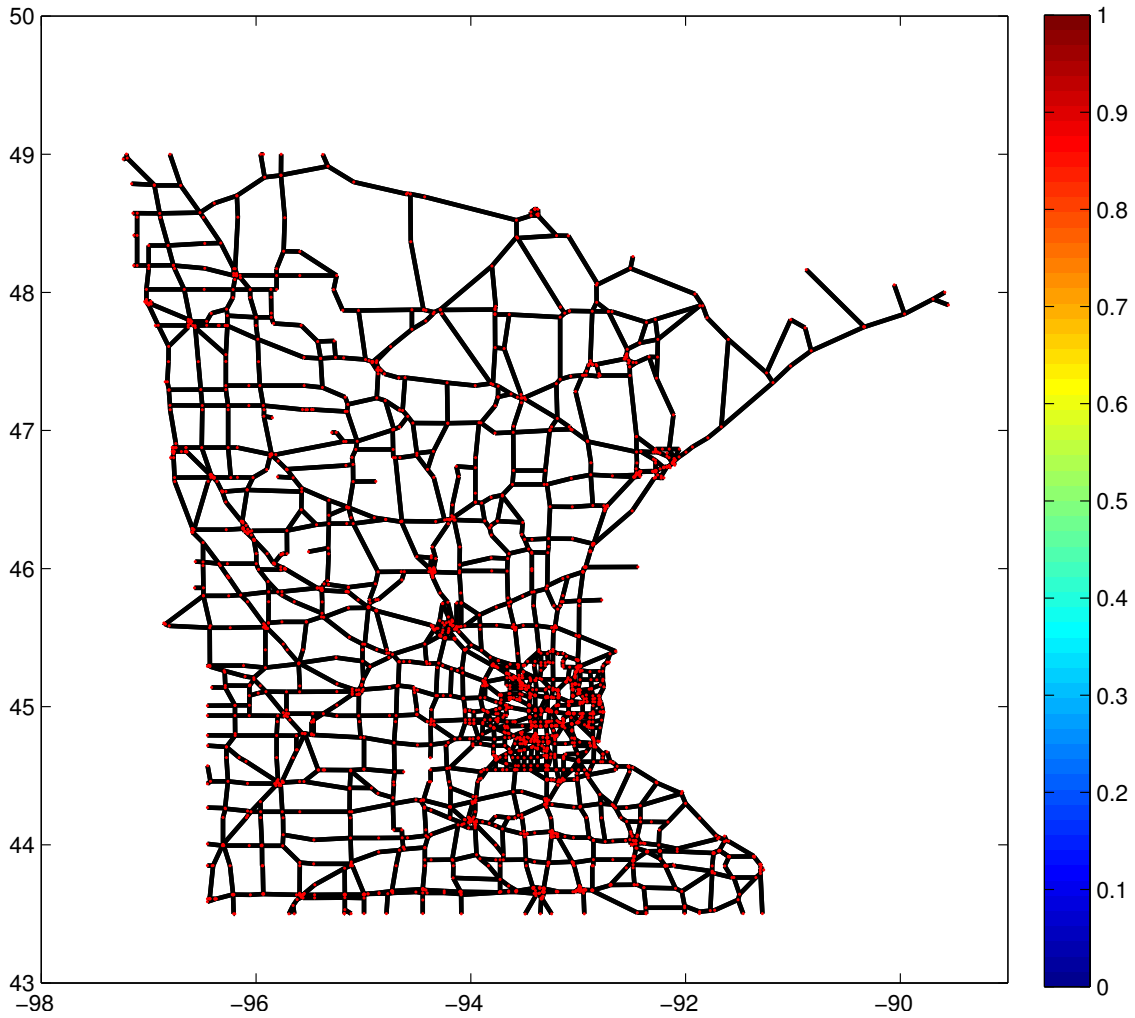


Figure: The MN road network

Example of Graph Partitioning

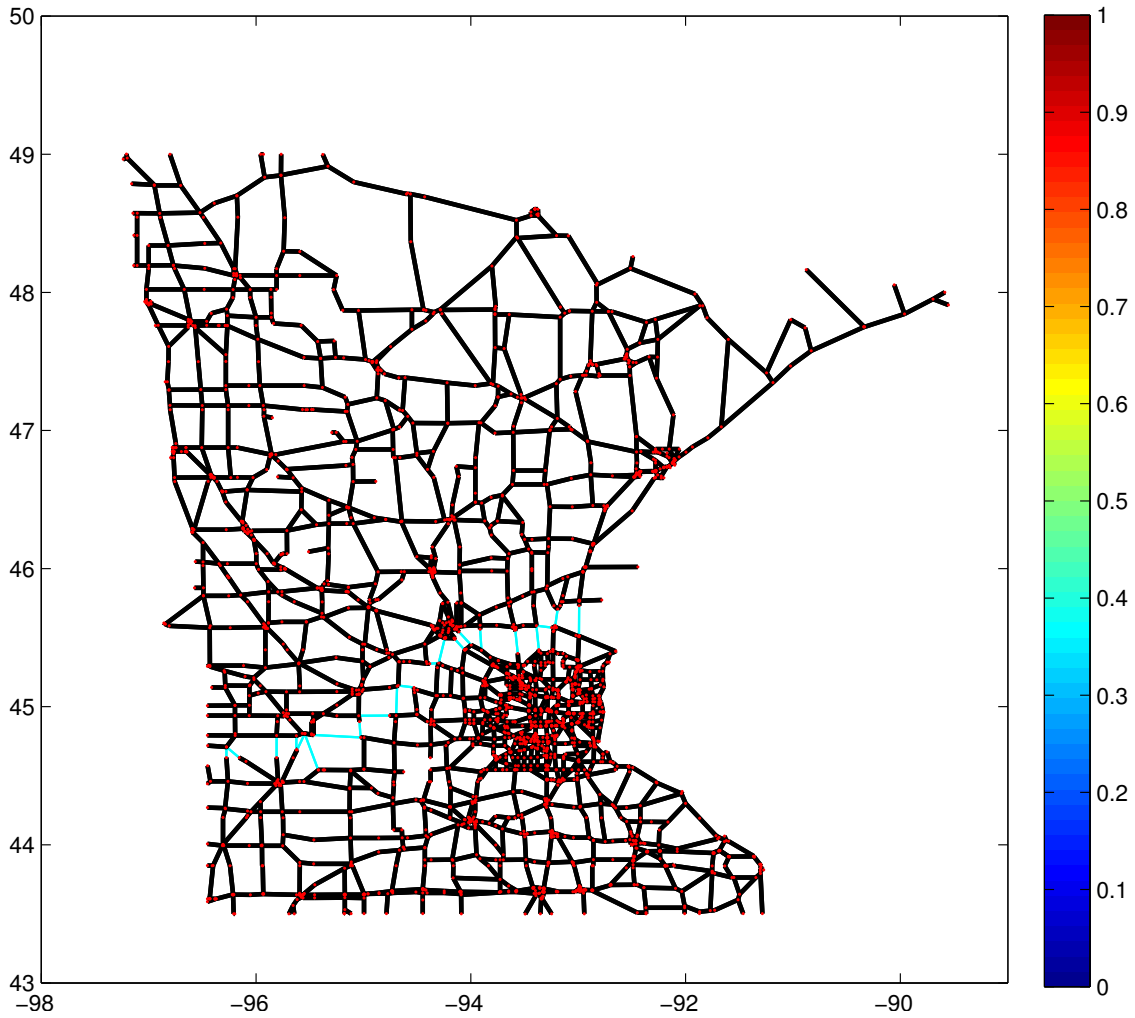
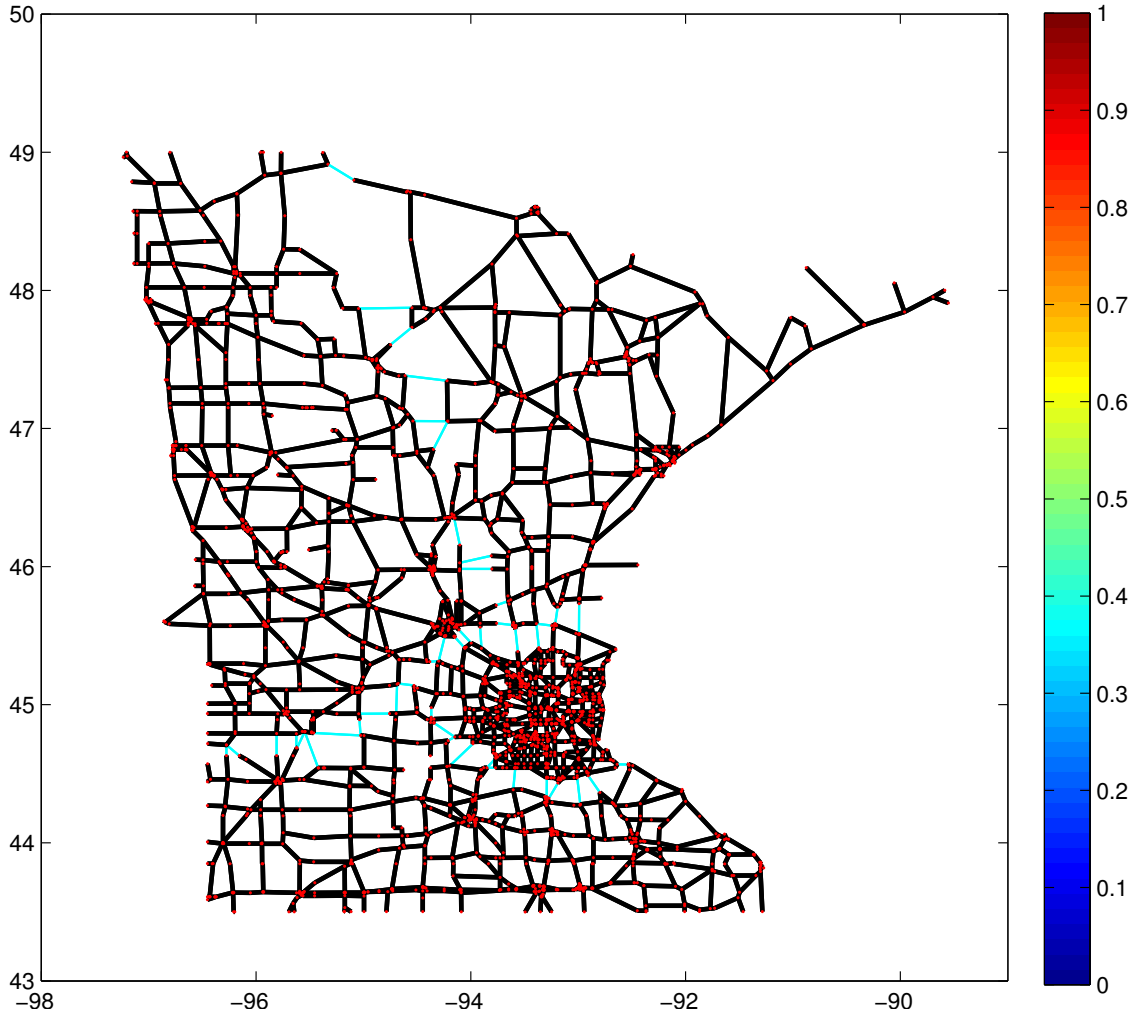


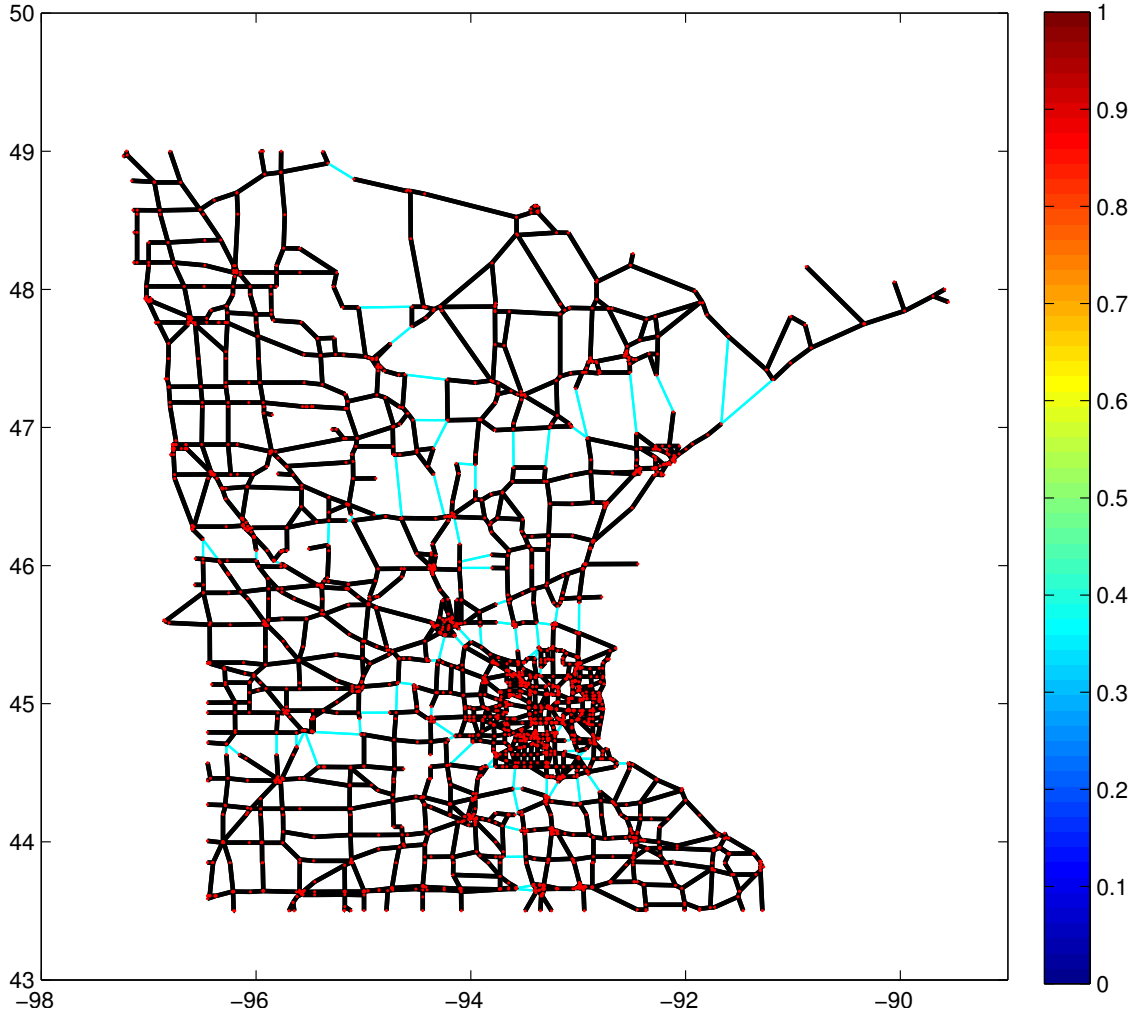
Figure: The MN road network partitioned via the Fiedler vector of L_{rw}

One Can Do This Recursively!



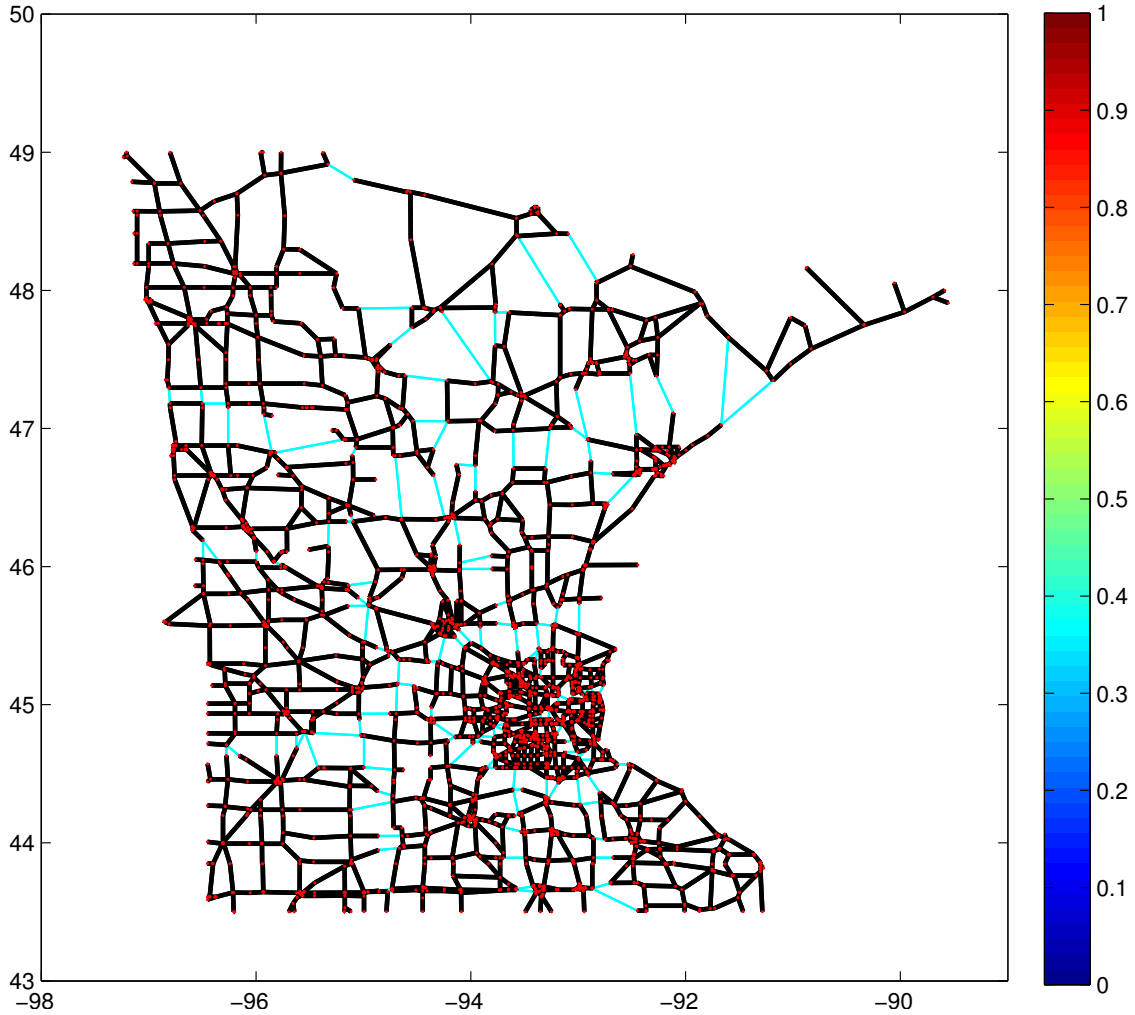
The MN road network recursively partitioned via the Fiedler vectors of L_{rw}'s of subgraphs: $j = 2$

One Can Do This Recursively!



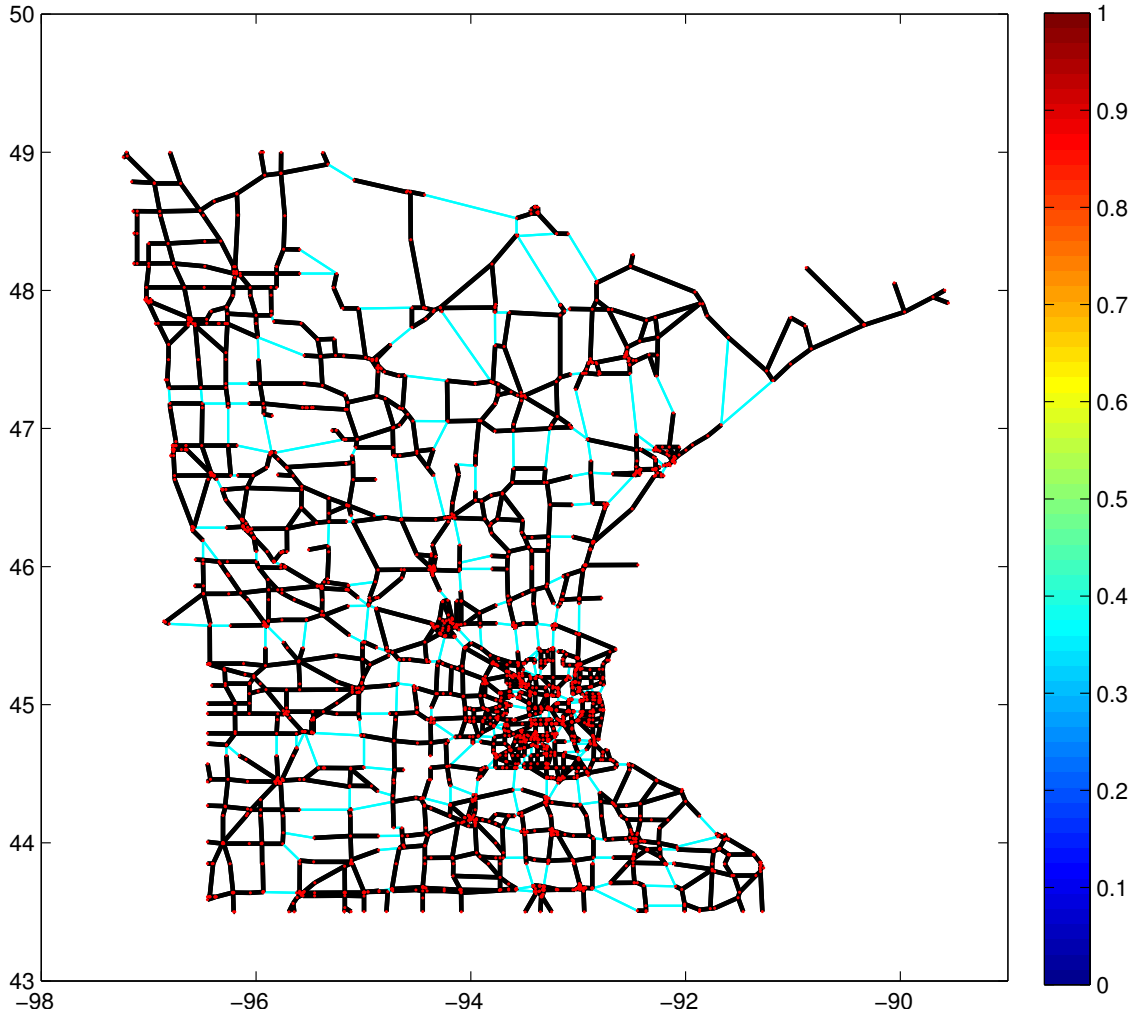
The MN road network recursively partitioned via the Fiedler vectors of L_{rw}'s of subgraphs: $j = 3$

One Can Do This Recursively!



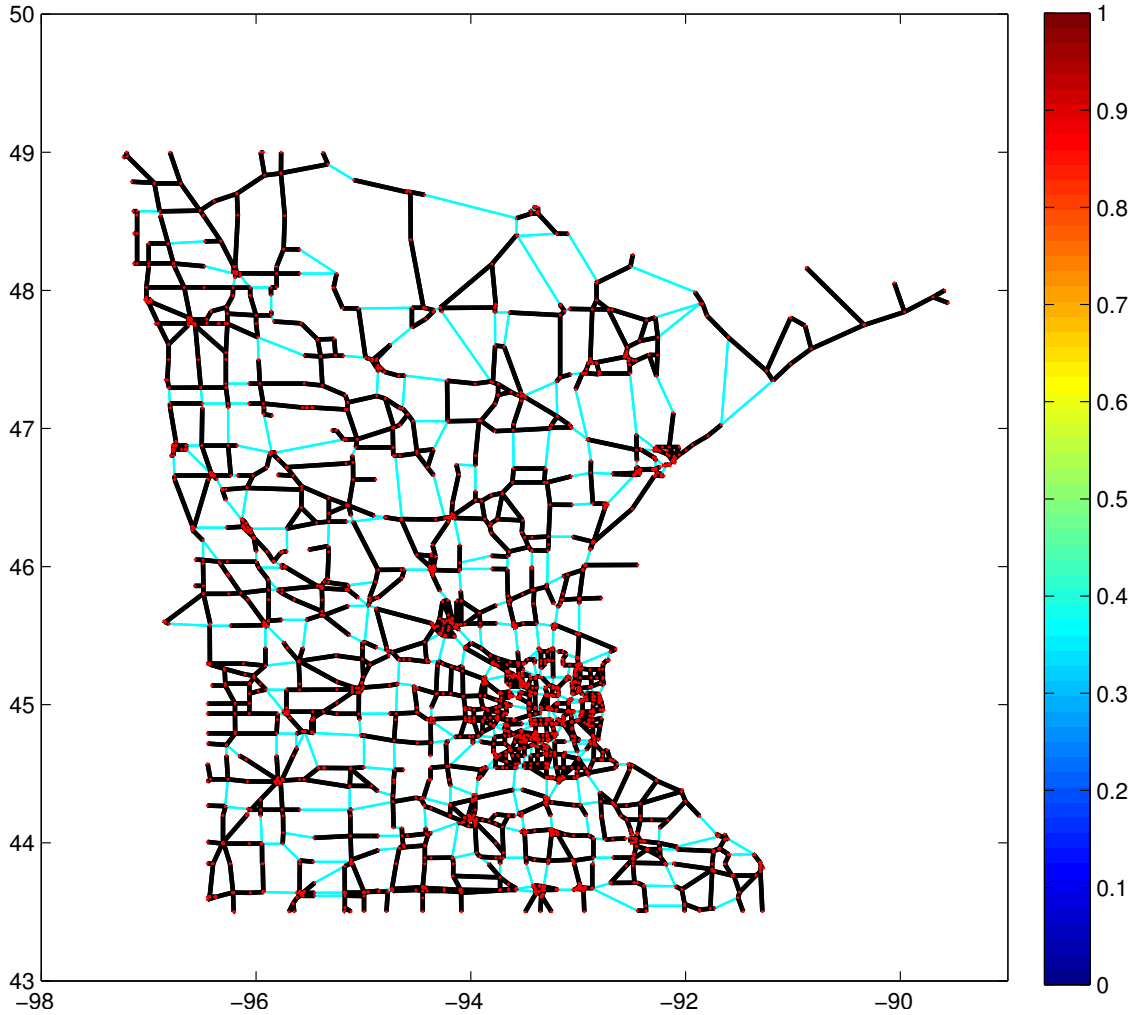
The MN road network recursively partitioned via the Fiedler vectors of L_{rw}'s of subgraphs: $j = 4$

One Can Do This Recursively!



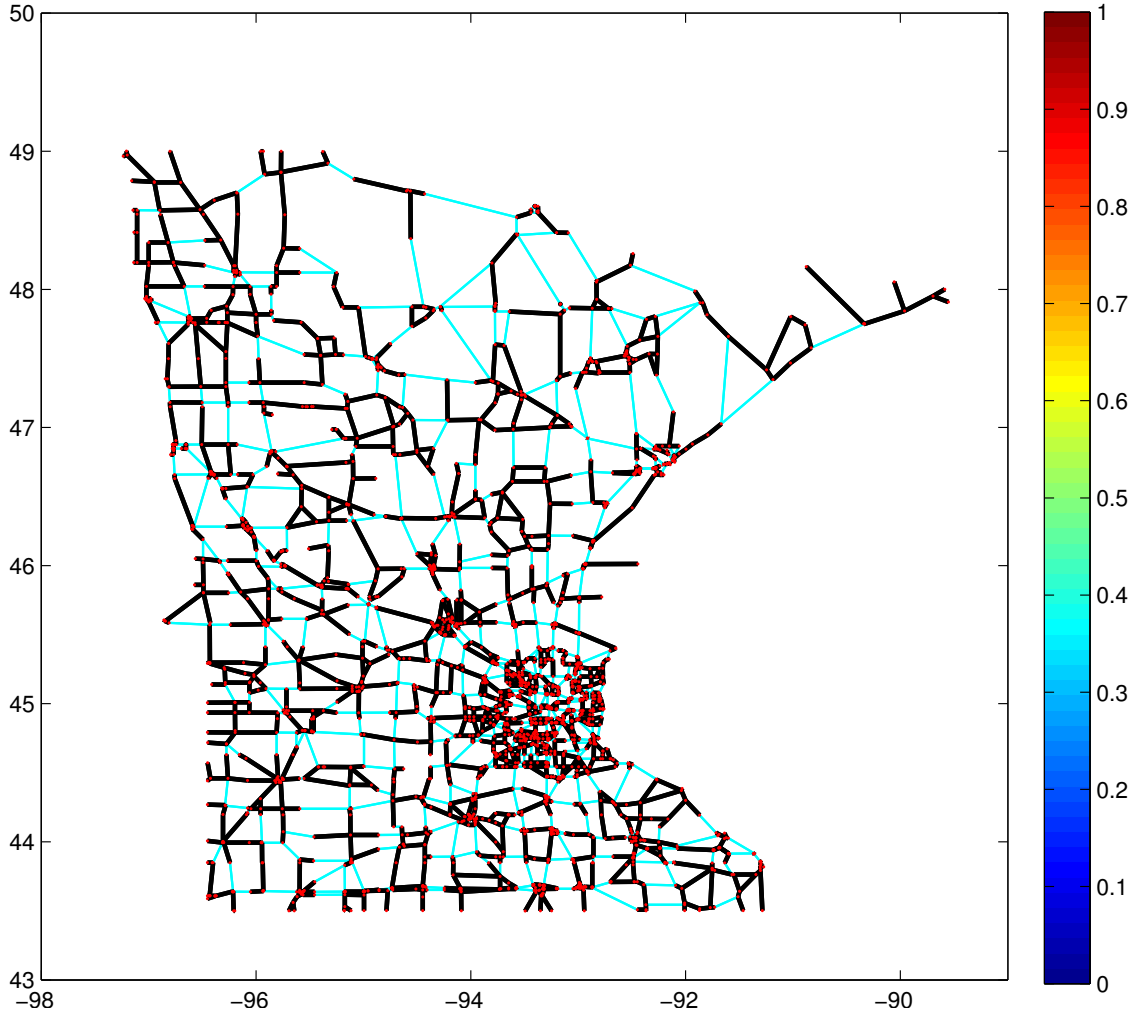
The MN road network recursively partitioned via the Fiedler vectors of L_{rw}'s of subgraphs: $j = 5$

One Can Do This Recursively!



The MN road network recursively partitioned via the Fiedler vectors of L_{rw}'s of subgraphs: $j = 6$

One Can Do This Recursively!



The MN road network recursively partitioned via the Fiedler vectors of L_{rw}'s of subgraphs: $j = 7$

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)
  - ...
- Key difficulties:
  - The notion of *frequency* is ill-defined on graphs and the Fourier transform is not properly defined on graphs
  - Hence, the use of graph Laplacian eigenvectors, which can be viewed as "cosines" on graphs, has been quite popular
  - However, they exhibit peculiar behaviors depending on *topology* and *structure* of given graphs!

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)
  - ...
- Key difficulties:
  - The notion of *frequency* is ill-defined on graphs and the Fourier transform is not properly defined on graphs
  - Hence, the use of graph Laplacian eigenvectors, which can be viewed as "cosines" on graphs, has been quite popular
  - However, they exhibit peculiar behaviors depending on *topology* and *structure* of given graphs!

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)
  - ...
- **Key difficulties:**
  - The notion of *frequency* is ill-defined on graphs and the Fourier transform is not properly defined on graphs
  - Hence, the use of graph Laplacian eigenvectors, which can be viewed as "cosines" on graphs, has been quite popular
  - However, they exhibit peculiar behaviors depending on *topology* and *structure* of given graphs!

Our transforms involve 2 main steps:

1. Recursively partition the graph

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

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

Our transforms involve 2 main steps:

1. Recursively partition the graph

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

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

Outline

Hierarchical Graph Laplacian Eigen Transform (HGLET)

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

The algorithm proceeds as follows...

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

$$\left[\begin{array}{cccc} \boldsymbol{\phi}_{0,0}^{0} & \boldsymbol{\phi}_{0,1}^{0} & \boldsymbol{\phi}_{0,2}^{0} & \cdots & \boldsymbol{\phi}_{0,N_0^0-1}^{0} \end{array} \right]$$

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

$$\left[\begin{array}{cccc} \boldsymbol{\phi}_{0,0}^{0} & \boldsymbol{\phi}_{0,1}^{0} & \boldsymbol{\phi}_{0,2}^{0} & \cdots & \boldsymbol{\phi}_{0,N_0^0-1}^{0} \end{array} \right]$$

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

$$\left[\begin{array}{cccc} \boldsymbol{\phi}_{0,0}^{0} & \boldsymbol{\phi}_{0,1}^{0} & \boldsymbol{\phi}_{0,2}^{0} & \cdots & \boldsymbol{\phi}_{0,N_0^0-1}^{0} \end{array} \right]$$

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

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

$$\left[\begin{array}{cccc} \boldsymbol{\phi}_{0,0}^{0} & \boldsymbol{\phi}_{0,1}^{0} & \boldsymbol{\phi}_{0,2}^{0} & \cdots & \boldsymbol{\phi}_{0,N_0^0-1}^{0} \end{array} \right]$$

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

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

$$\left[\begin{array}{cccc} \boldsymbol{\phi}_{0,0}^{0} & \boldsymbol{\phi}_{0,1}^{0} & \boldsymbol{\phi}_{0,2}^{0} & \cdots & \boldsymbol{\phi}_{0,N_0^0-1}^{0} \end{array} \right]$$

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

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

$$\left[\begin{array}{ccccc} \boldsymbol{\phi}_{0,0}^{0} & \boldsymbol{\phi}_{0,1}^{0} & \boldsymbol{\phi}_{0,2}^{0} & \cdots & \boldsymbol{\phi}_{0,N_0^0-1}^{0} \end{array}\right]$$

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

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

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

$$\left[\begin{array}{ccccc} \boldsymbol{\phi}_{0,0}^{0} & \boldsymbol{\phi}_{0,1}^{0} & \boldsymbol{\phi}_{0,2}^{0} & \cdots & \boldsymbol{\phi}_{0,N_0^0-1}^{0} \end{array}\right]$$

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

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

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

$$\left[\begin{array}{ccccc} \boldsymbol{\phi}_{0,0}^{0} & \boldsymbol{\phi}_{0,1}^{0} & \boldsymbol{\phi}_{0,2}^{0} & \cdots & \boldsymbol{\phi}_{0,N_0^0-1}^{0} \end{array}\right]$$

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

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

$$\vdots$$

Remarks

- For an unweighted path graph, this exactly yields a *dictionary of the multiscale BDCT-II*
- 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, ...
  - A union of bases on disjoint subsets is obviously orthonormal

Remarks

- For an unweighted path graph, this exactly yields a *dictionary of the multiscale BDCT-II*
- 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, ...
  - A union of bases on disjoint subsets is obviously orthonormal

Remarks

- For an unweighted path graph, this exactly yields a *dictionary of the multiscale BDCT-II*
- 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, ...
  - A union of bases on disjoint subsets is obviously orthonormal

Remarks

- For an unweighted path graph, this exactly yields a *dictionary of the multiscale BDCT-II*
- 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, ...
  - A union of bases on disjoint subsets is obviously orthonormal

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

$$\left[\begin{array}{cccc} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,N^1_0-1} \end{array}\right] \left[\begin{array}{cccc} \boldsymbol{\phi}^1_{1,0} & \boldsymbol{\phi}^1_{1,1} & \boldsymbol{\phi}^1_{1,2} & \cdots & \boldsymbol{\phi}^1_{1,N^1_1-1} \end{array}\right]$$

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

Remarks

- For an unweighted path graph, this exactly yields a *dictionary of the multiscale BDCT-II*
- 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, ...
  - A union of bases on disjoint subsets is obviously orthonormal

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

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

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

Remarks

- For an unweighted path graph, this exactly yields a *dictionary of the multiscale BDCT-II*
- 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, ...
  - A union of bases on disjoint subsets is obviously orthonormal

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

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

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

Remarks

- For an unweighted path graph, this exactly yields a *dictionary of the multiscale BDCT-II*
- 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, ...
  - A union of bases on disjoint subsets is obviously orthonormal

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

$$\left[\begin{array}{cccc} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,N^1_0-1} \end{array}\right] \left[\begin{array}{cccc} \boldsymbol{\phi}^1_{1,0} & \boldsymbol{\phi}^1_{1,1} & \boldsymbol{\phi}^1_{1,2} & \cdots & \boldsymbol{\phi}^1_{1,N^1_1-1} \end{array}\right]$$

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

Remarks

- For an unweighted path graph, this exactly yields a *dictionary of the multiscale BDCT-II*
- 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, ...
  - A union of bases on disjoint subsets is obviously orthonormal

$$\left[\begin{array}{cccc} \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,N_0^0-1} \end{array}\right]$$

$$\left[\begin{array}{cccccc} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,N_0^1-1} \end{array}\right] \left[\begin{array}{cccccc} \boldsymbol{\phi}^1_{1,0} & \boldsymbol{\phi}^1_{1,1} & \boldsymbol{\phi}^1_{1,2} & \cdots & \boldsymbol{\phi}^1_{1,N_1^1-1} \end{array}\right]$$

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

Remarks

- For an unweighted path graph, this exactly yields a *dictionary of the multiscale BDCT-II*
- 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, ...
  - A union of bases on disjoint subsets is obviously orthonormal

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

$$\left[\begin{array}{cccc} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,N^1_0-1} \end{array}\right] \left[\begin{array}{cccc} \boldsymbol{\phi}^1_{1,0} & \boldsymbol{\phi}^1_{1,1} & \boldsymbol{\phi}^1_{1,2} & \cdots & \boldsymbol{\phi}^1_{1,N^1_1-1} \end{array}\right]$$

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

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 proceeds as follows...

Generalized Haar-Walsh Transform (GHWT)

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

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

The algorithm proceeds as follows...

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

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

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

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

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

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

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

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

$$\vdots$$

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

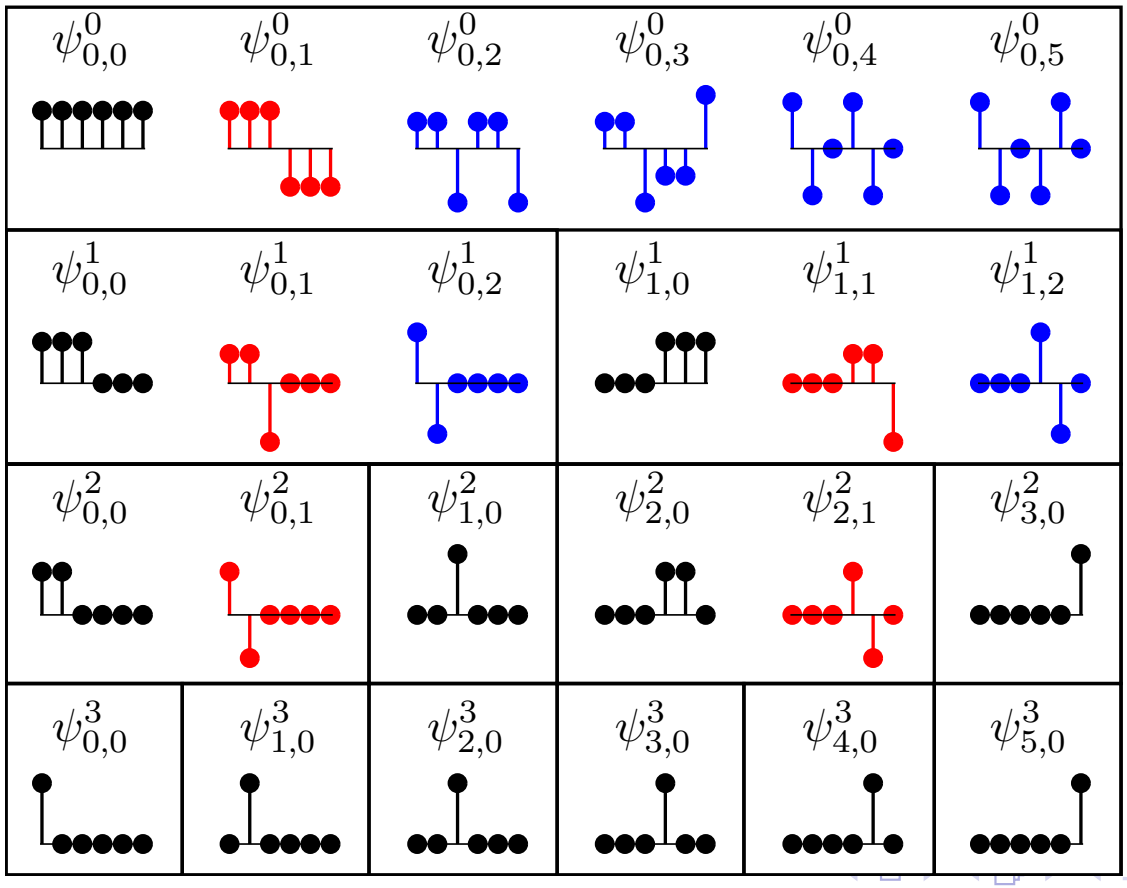
$$\left[\ \boldsymbol{\psi}^{j_{\max}}_{0,0}\ \right] \left[\ \boldsymbol{\psi}^{j_{\max}}_{1,0}\ \right] \left[\ \boldsymbol{\psi}^{j_{\max}}_{2,0}\ \right] \left[\ \boldsymbol{\psi}^{j_{\max}}_{3,0}\ \right] \cdots \left[\ \boldsymbol{\psi}^{j_{\max}}_{K^{j_{\max}}-2,0}\ \right] \left[\ \boldsymbol{\psi}^{j_{\max}}_{K^{j_{\max}}-1,0}\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

Remarks

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

Remarks

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



Remarks

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

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**)

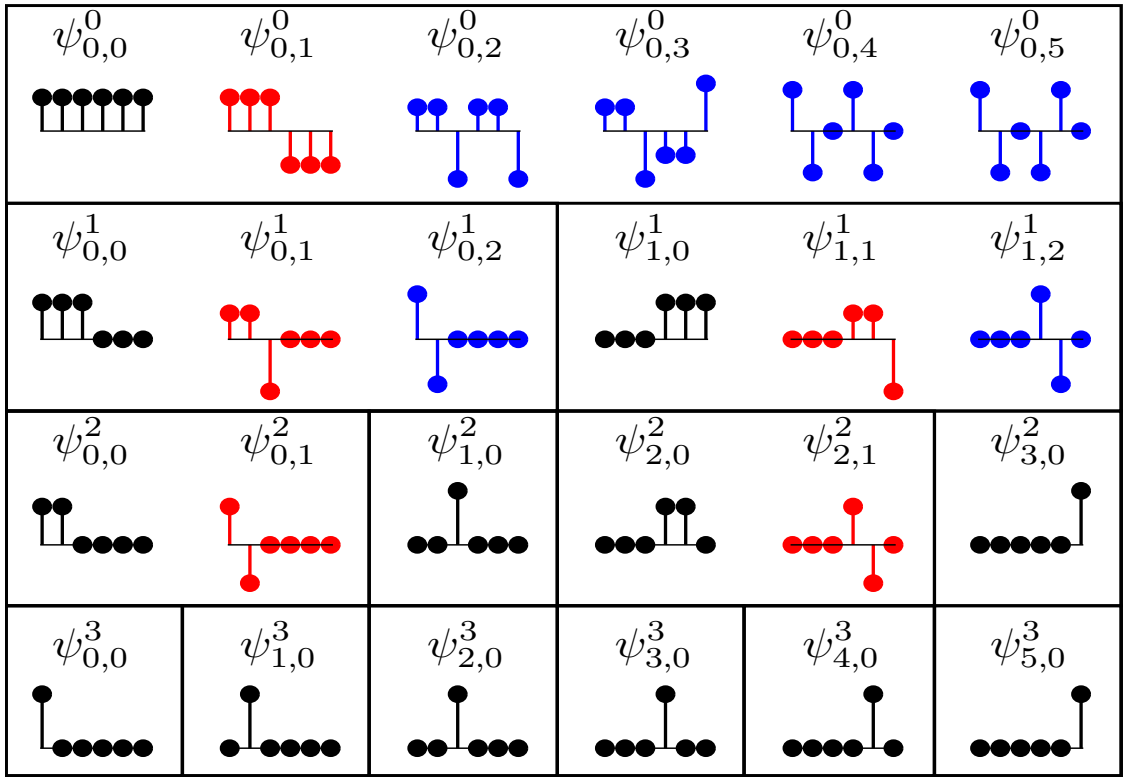


Figure: Default dictionary; i.e., coarse-to-fine

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**)

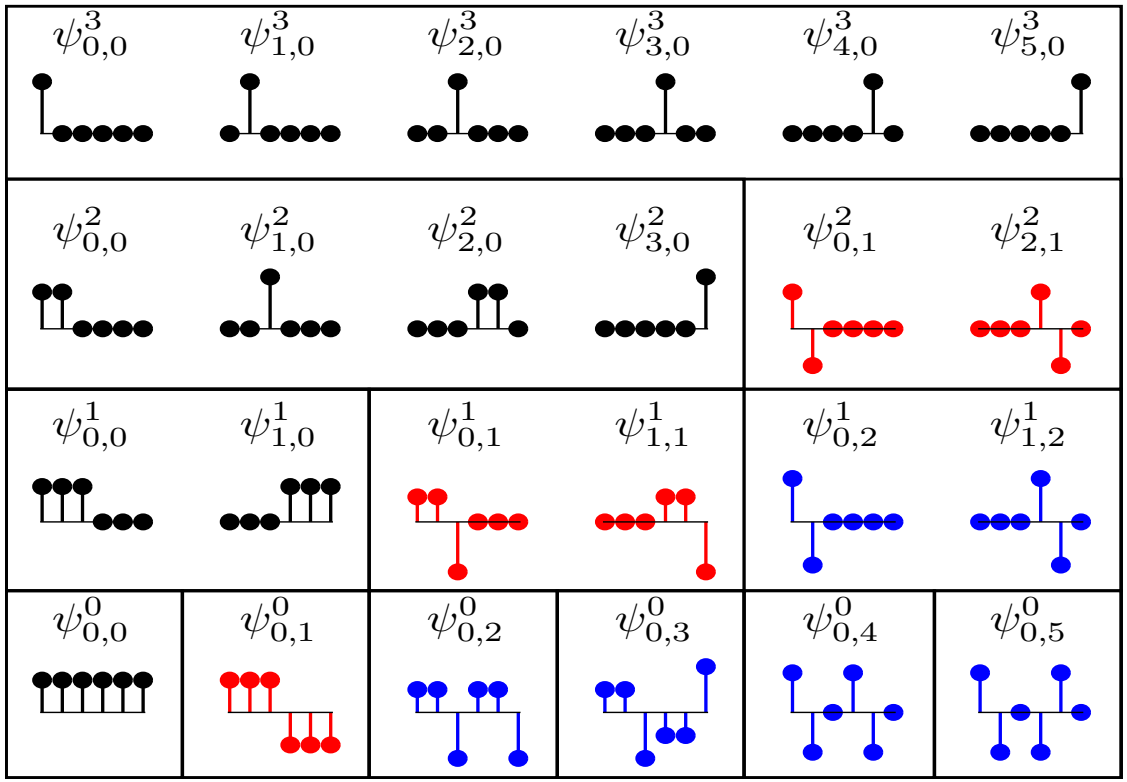


Figure: Reordered & regrouped dictionary; i.e., fine-to-coarse

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**)

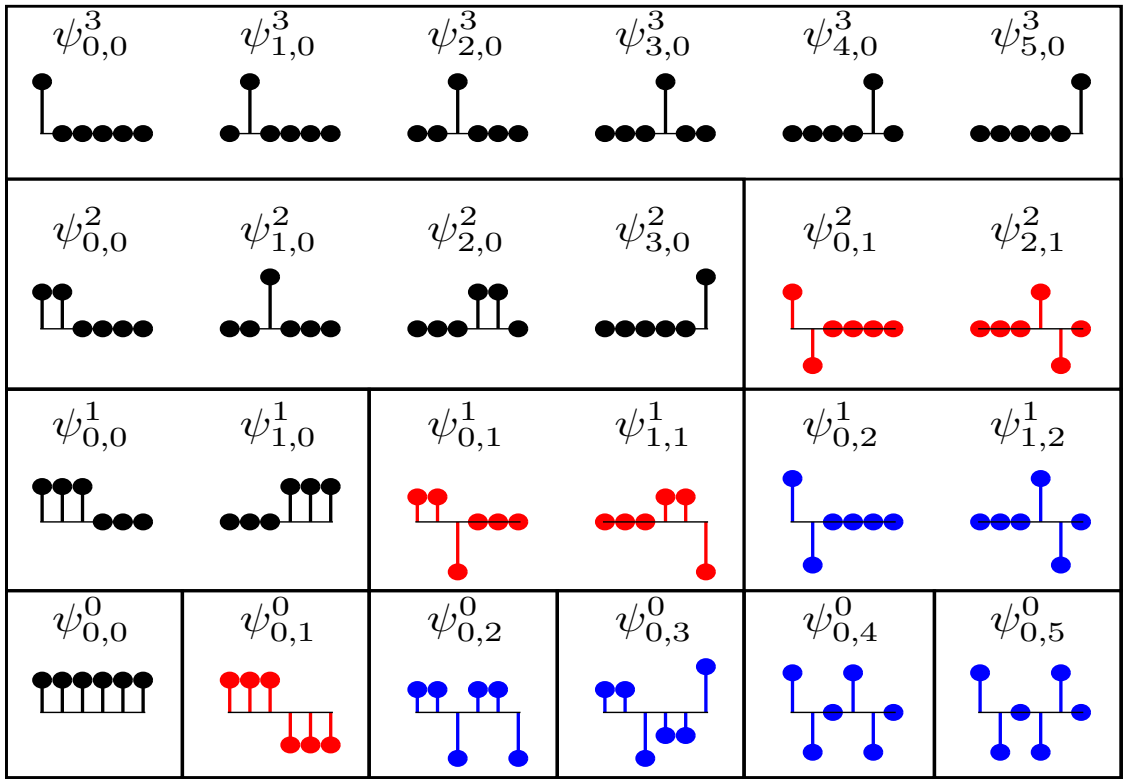


Figure: Reordered & regrouped dictionary; i.e., fine-to-coarse

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

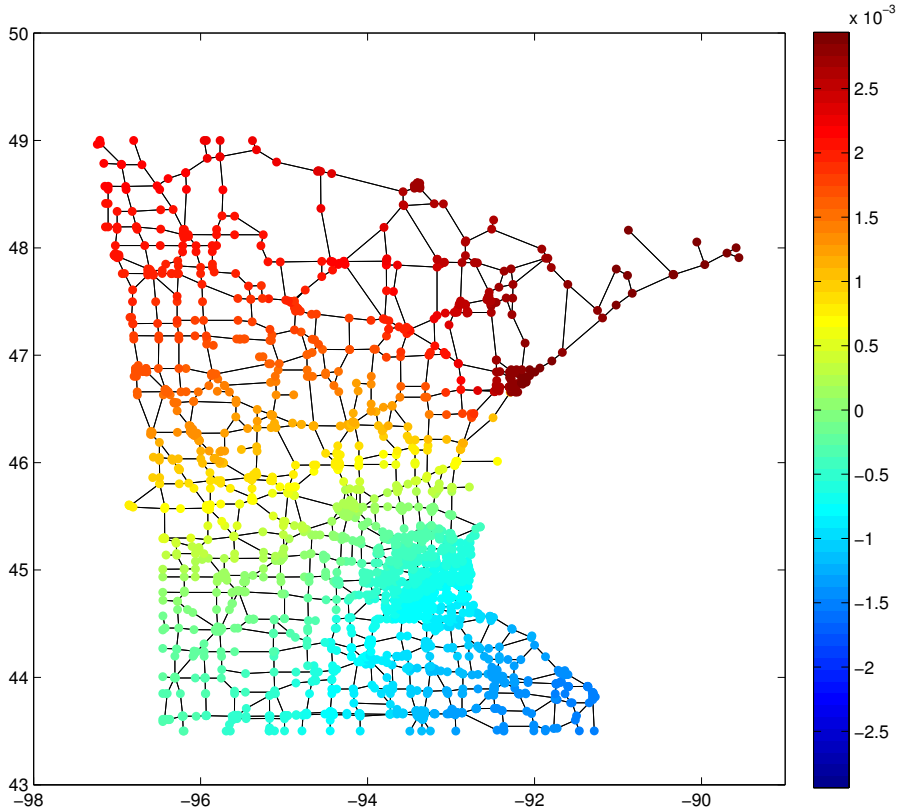
HGLET vs. GHWT

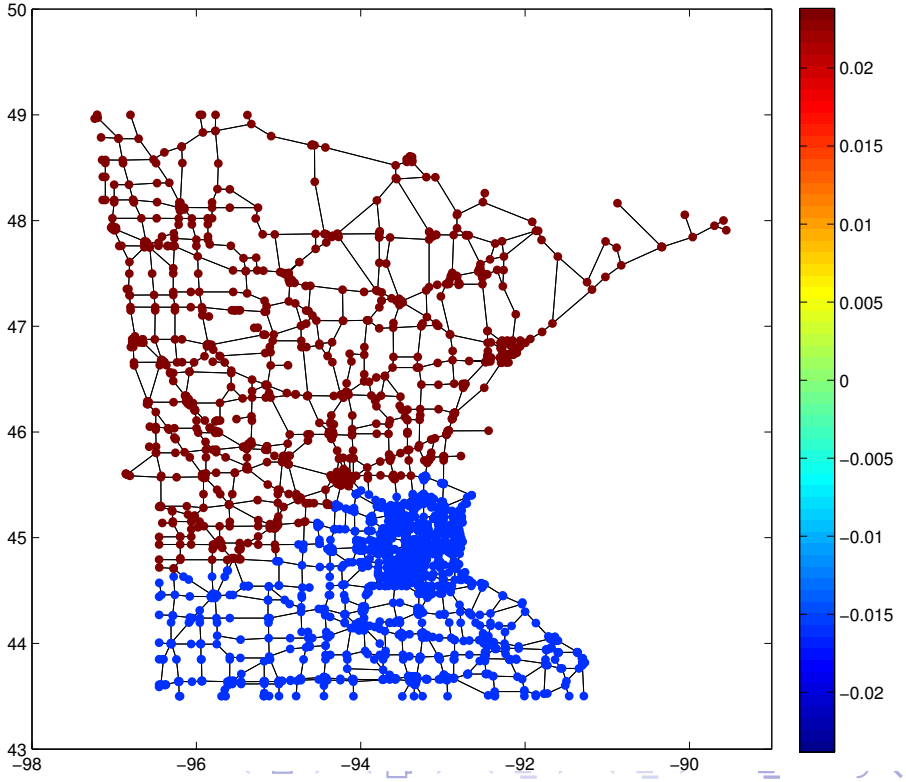
Here we display some of the basis vectors generated by our HGLET (left) and GHWT (right) schemes on the MN road network. (Note: $j = 0$ is the coarsest scale, $j = 14$ is the finest.)

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 $k = 0$, $l = 1$

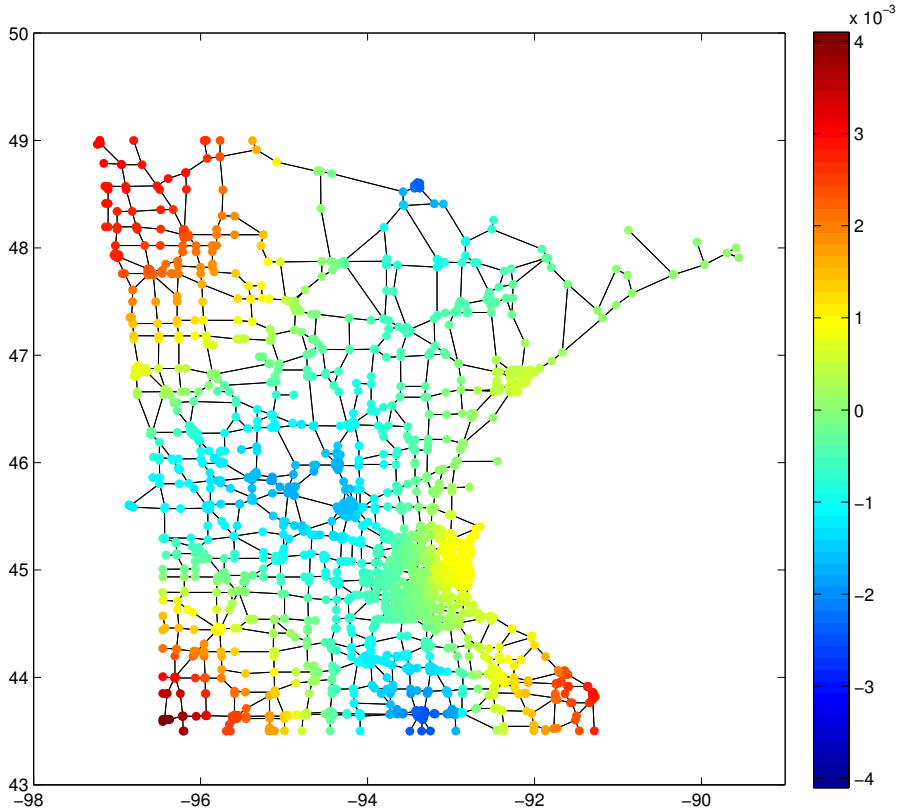


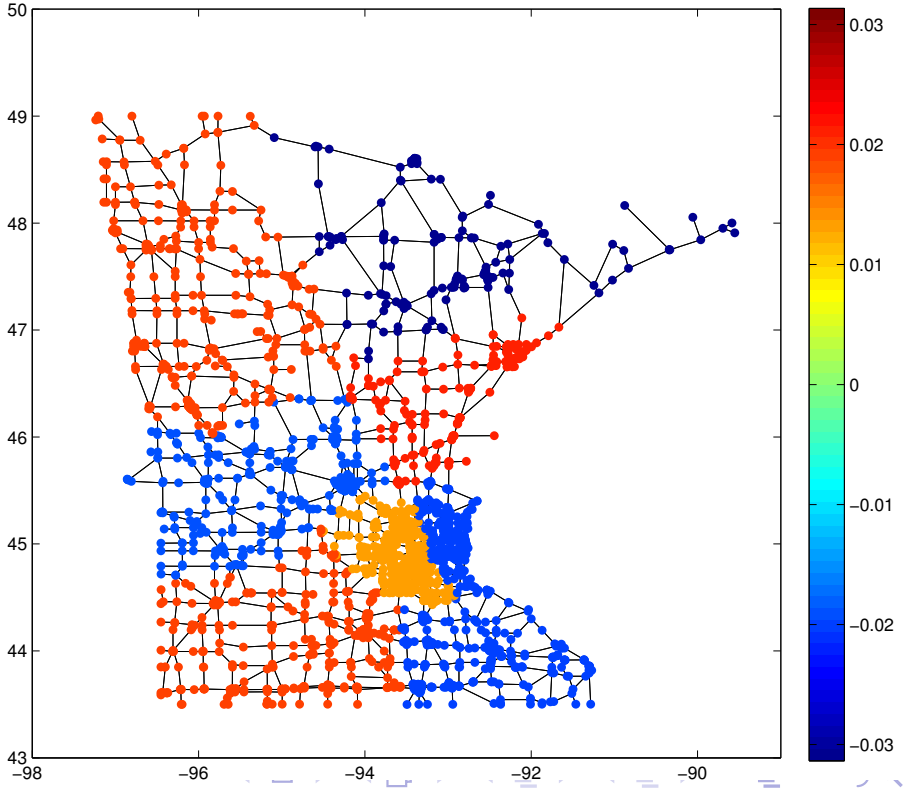


HGLET vs. GHWT

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 $k = 0$, $l = 7$

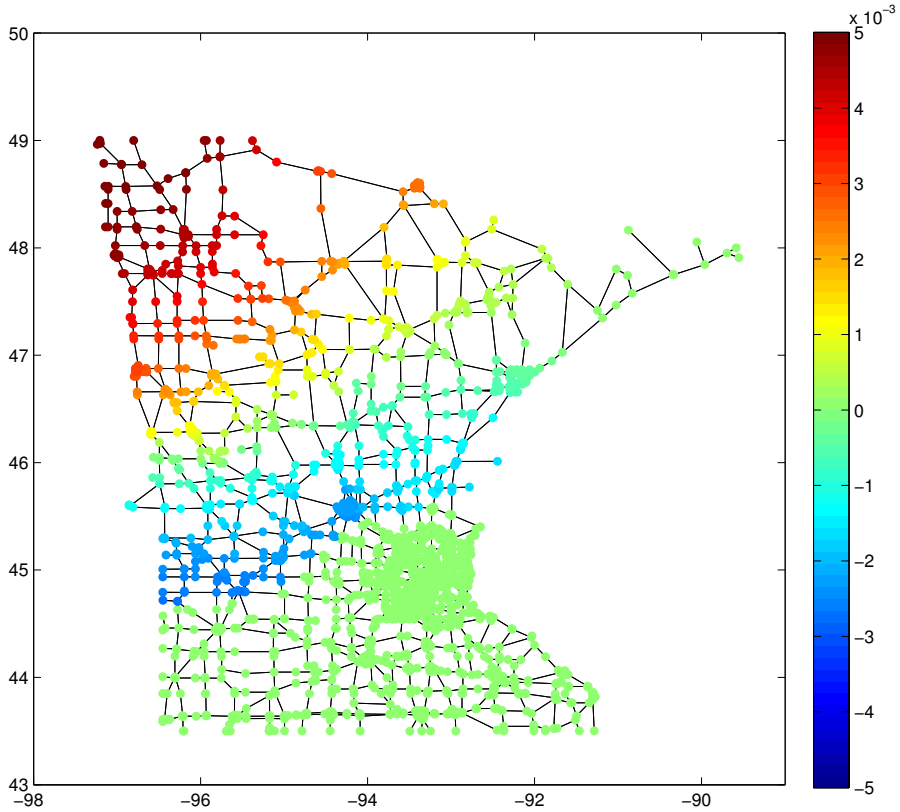


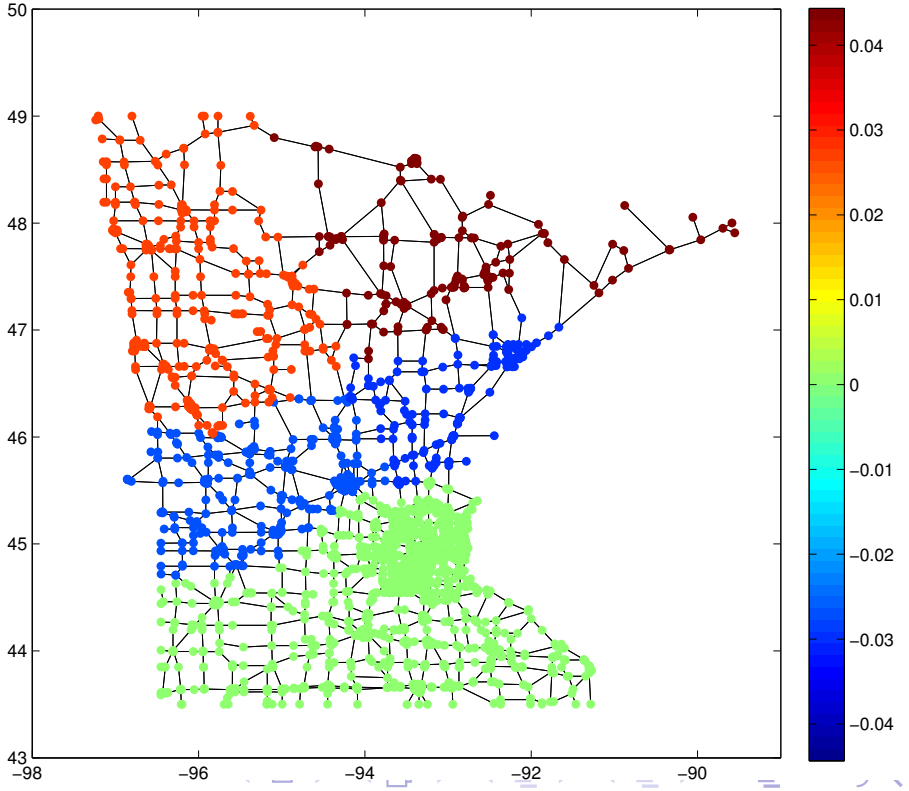


HGLET vs. GHWT

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 $k = 0$, $l = 2$

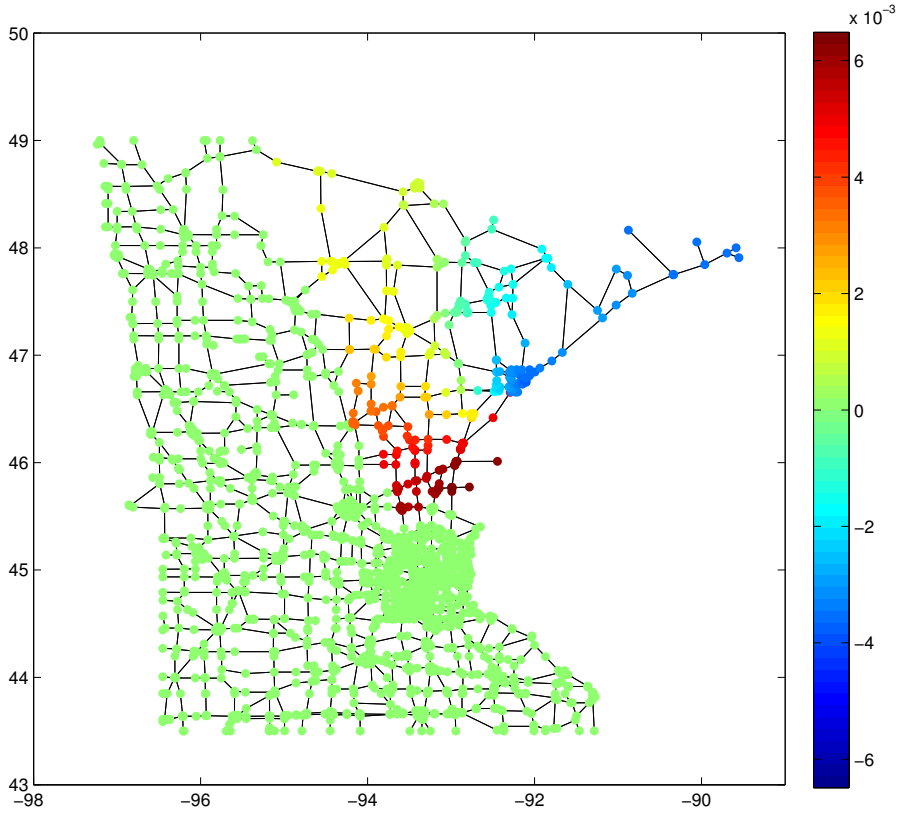


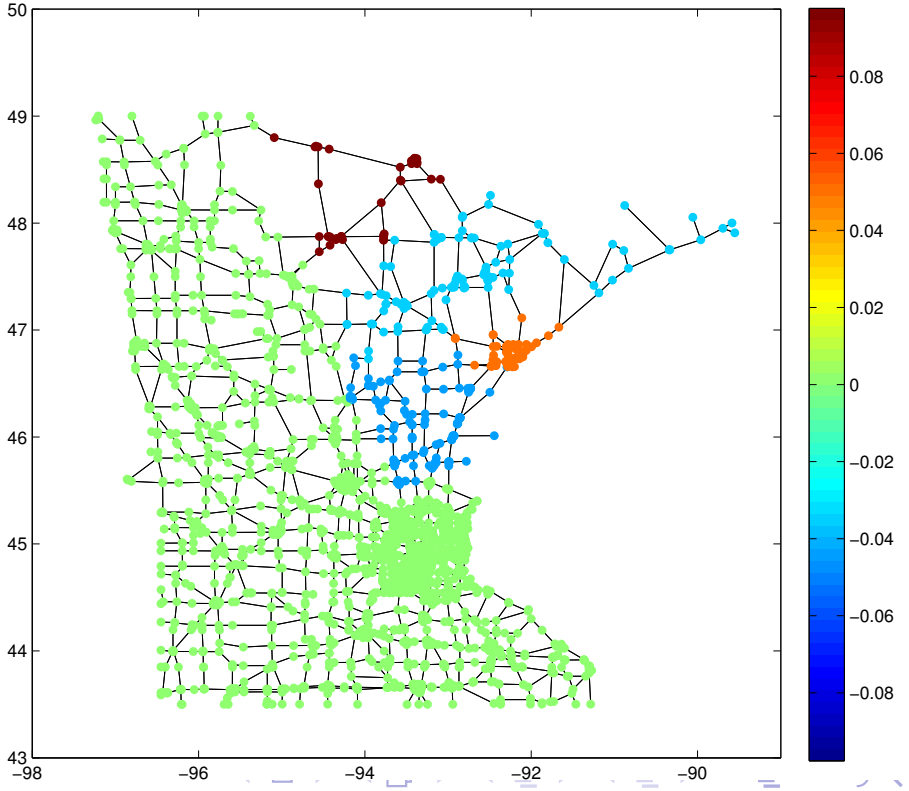


HGLET vs. GHWT

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 $k = 1$, $l = 2$

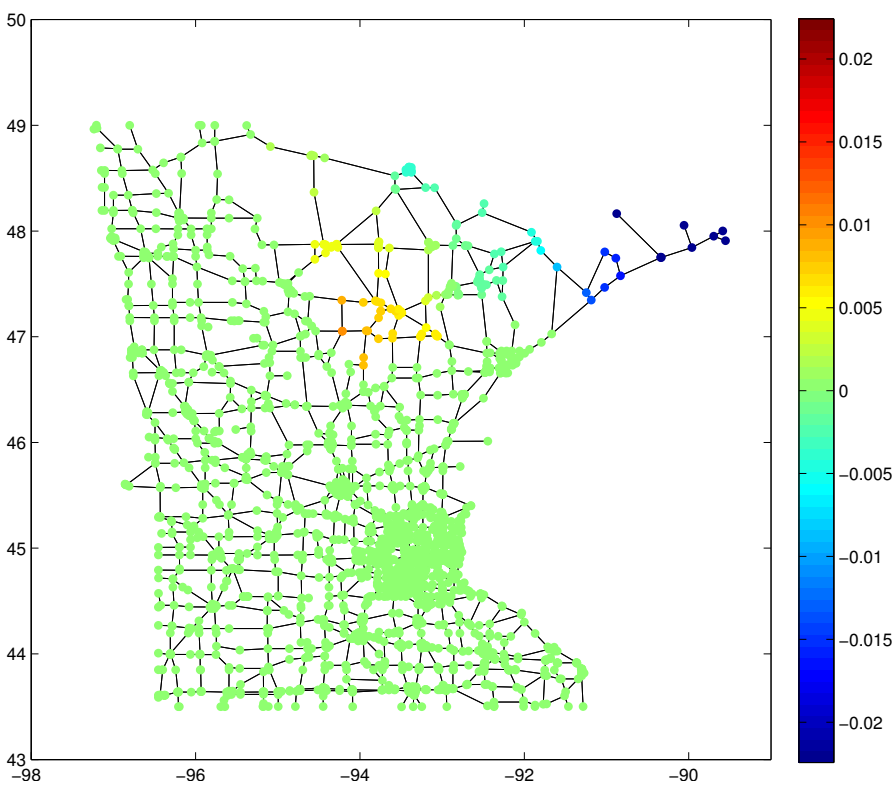


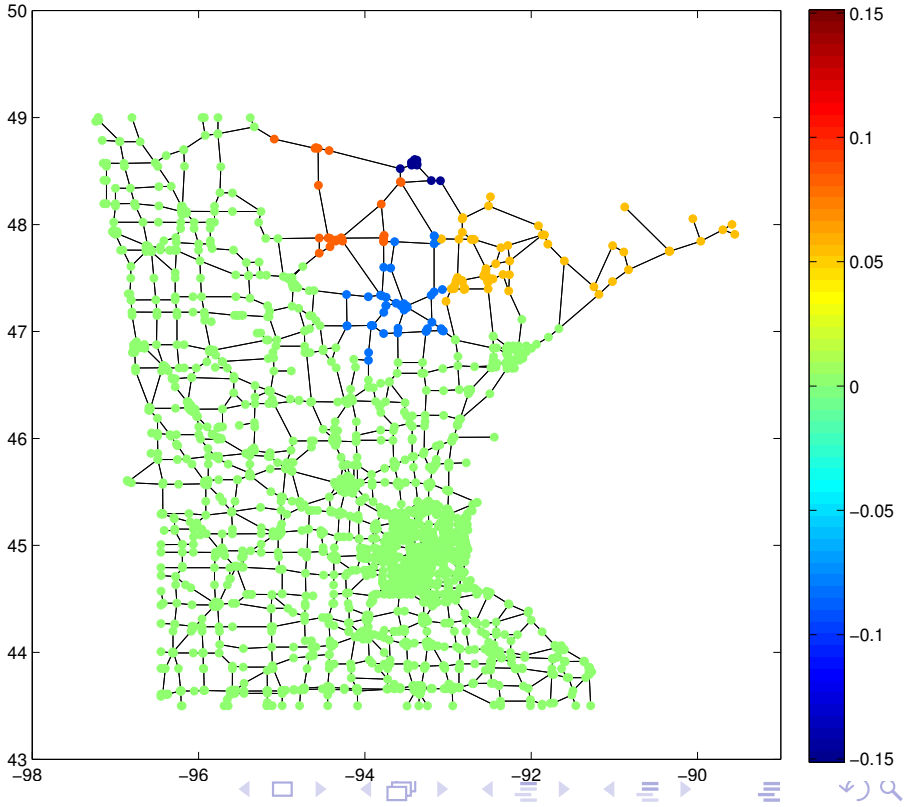


HGLET vs. GHWT

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

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

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 *not the Walsh-Hadamard transform* on them:

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

Outline

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\begin{matrix} c^0_{0,0} & c^0_{0,1} & c^0_{0,2} & \cdots & c^0_{0,N^0_0-1} \end{matrix}$$

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

$$\begin{matrix} c^1_{0,0} & c^1_{0,1} & c^1_{0,2} & \cdots & c^1_{0,N^1_0-1} \end{matrix} \quad \begin{matrix} c^1_{1,0} & c^1_{1,1} & c^1_{1,2} & \cdots & c^1_{1,N^1_1-1} \end{matrix}$$

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

$$\begin{matrix} c^2_{0,0} & c^2_{0,1} \cdots & c^2_{0,N^2_0-1} \end{matrix} \quad \begin{matrix} c^2_{1,0} & c^2_{1,1} \cdots & c^2_{1,N^2_1-1} \end{matrix} \quad \begin{matrix} c^2_{2,0} & c^2_{2,1} \cdots & c^2_{2,N^2_2-1} \end{matrix} \quad \begin{matrix} c^2_{3,0} & c^2_{3,1} \cdots & c^2_{3,N^2_3-1} \end{matrix}$$

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

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

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

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

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

$$\begin{matrix} c^0_{0,0} & c^0_{0,1} & c^0_{0,2} & \cdots & c^0_{0,N^0_0-1} \end{matrix}$$

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

$$\begin{matrix} c^1_{0,0} & c^1_{0,1} & c^1_{0,2} & \cdots & c^1_{0,N^1_0-1} \end{matrix} \quad \begin{matrix} c^1_{1,0} & c^1_{1,1} & c^1_{1,2} & \cdots & c^1_{1,N^1_1-1} \end{matrix}$$

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

$$\begin{matrix} c^2_{2,0} & c^2_{2,1} \cdots & c^2_{2,N^2_2-1} \end{matrix} \quad \begin{matrix} c^2_{3,0} & c^2_{3,1} \cdots & c^2_{3,N^2_3-1} \end{matrix}$$

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

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

$$\begin{matrix} c^0_{0,0} & c^0_{0,1} & c^0_{0,2} & \cdots & c^0_{0,N^0_0-1} \end{matrix}$$

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

$$\begin{matrix} c^1_{0,0} & c^1_{0,1} & c^1_{0,2} & \cdots & c^1_{0,N^1_0-1} \end{matrix} \quad \begin{matrix} c^1_{1,0} & c^1_{1,1} & c^1_{1,2} & \cdots & c^1_{1,N^1_1-1} \end{matrix}$$

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

$$\begin{matrix} c^2_{2,0} & c^2_{2,1} \cdots & c^2_{2,N^2_2-1} \end{matrix} \quad \begin{matrix} c^2_{3,0} & c^2_{3,1} \cdots & c^2_{3,N^2_3-1} \end{matrix}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

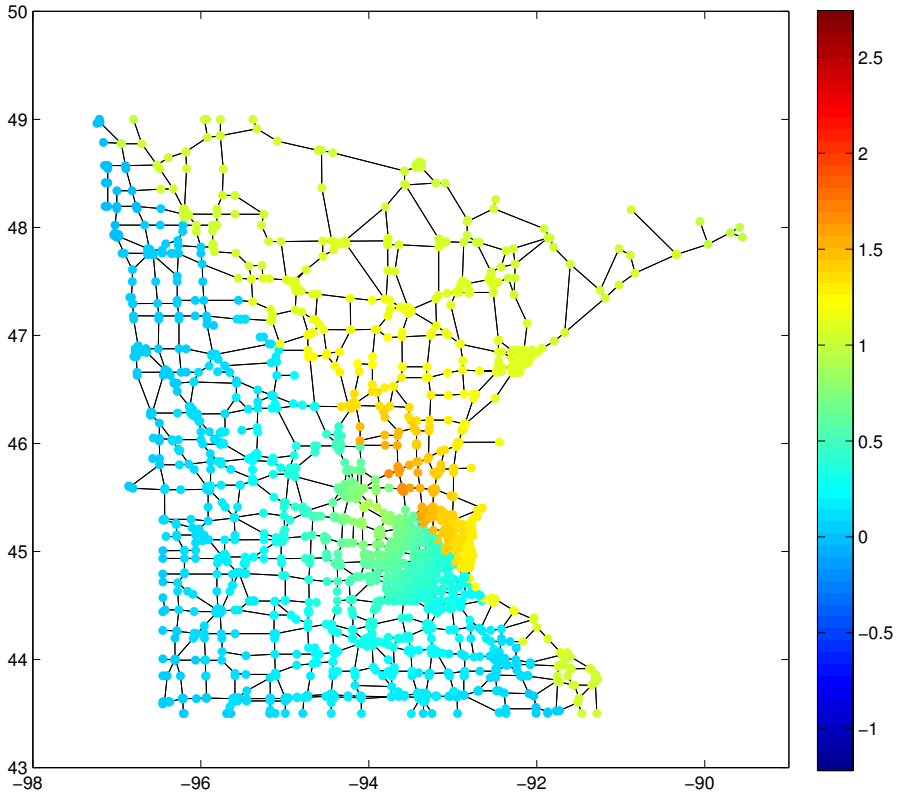
$$\begin{bmatrix} \boldsymbol{\phi}^2_{2,0}\, \boldsymbol{\phi}^2_{2,1} \cdots \boldsymbol{\phi}^2_{2,N^2_2-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^2_{3,0}\, \boldsymbol{\phi}^2_{3,1} \cdots \boldsymbol{\phi}^2_{3,N^2_3-1} \end{bmatrix}$$
$$\begin{matrix} c^2_{2,0}\ c^2_{2,1} \cdots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

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

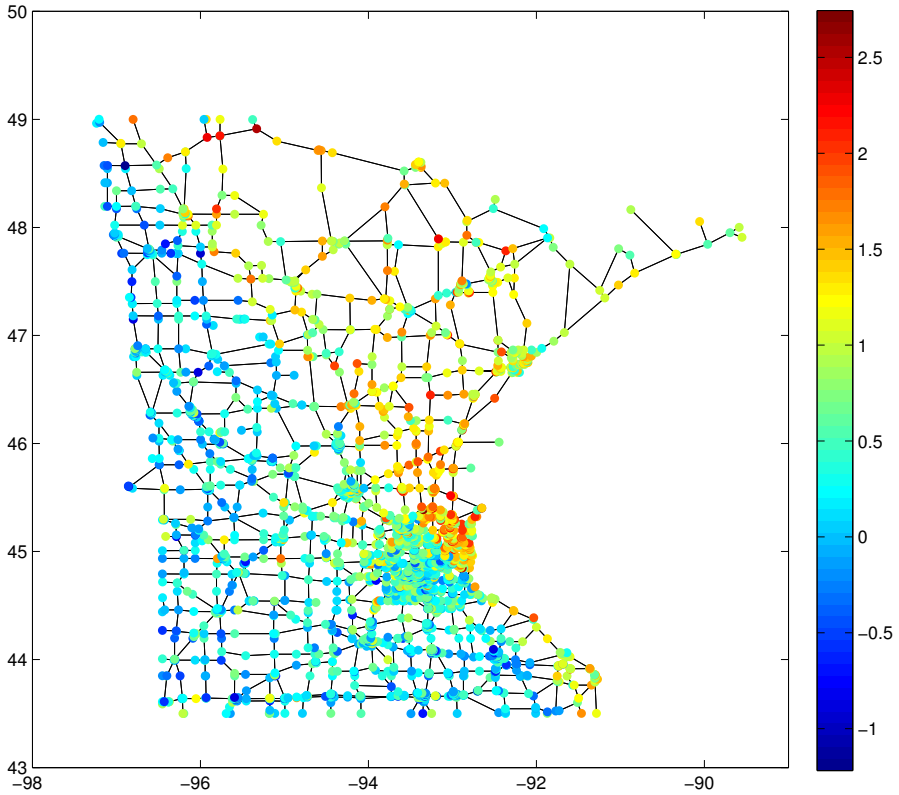
According to cost functional $\mathscr{J}$, this is the best basis for approximation. With the GHWT bases, we can run the best-basis algorithm on both the default (*coarse-to-fine*) dictionary and the reorganized (*fine-to-coarse*) dictionary and then compare the results.

Outline

Original Signal vs. Noisy Signal



(a) Original signal: mutilated Gaussian



(b) Noisy signal: SNR = 5dB, i.e., Noise energy $\approx 0.3162 \times$ Signal energy

Denoising Algorithm

1. Construct the HGLET / GHWT dictionaries on the noisy signal
2. Choose a particular basis either automatically (e.g., the best basis) or manually (e.g., a basis at the fixed scale)
3. Soft-threshold the expansion coefficients, i.e.,
   - Sort the expansion coefficients in non-increasing order of magnitude
   - Specify a magnitude threshold, T, via the "elbow" selection algorithm
   - Soft-threshold the coefficients c:

$$d_{ST}(i) = \begin{cases} \mathrm{sign}(c(i)) \cdot (|c(i)| - T) & \text{if } |c(i)| > T \\ 0 & \text{otherwise} \end{cases}$$

   - Note: keep all scaling coefficients intact

Denoising Algorithm

1. Construct the HGLET / GHWT dictionaries on the noisy signal
2. Choose a particular basis either automatically (e.g., the best basis) or manually (e.g., a basis at the fixed scale)
3. Soft-threshold the expansion coefficients, i.e.,
   - Sort the expansion coefficients in non-increasing order of magnitude
   - Specify a magnitude threshold, T, via the "elbow" selection algorithm
   - Soft-threshold the coefficients c:

$$d_{ST}(i) = \begin{cases} \mathrm{sign}(c(i)) \cdot (|c(i)| - T) & \text{if } |c(i)| > T \\ 0 & \text{otherwise} \end{cases}$$

   - Note: keep all scaling coefficients intact

Denoising Algorithm

1. Construct the HGLET / GHWT dictionaries on the noisy signal
2. Choose a particular basis either automatically (e.g., the best basis) or manually (e.g., a basis at the fixed scale)
3. Soft-threshold the expansion coefficients, i.e.,
   - Sort the expansion coefficients in non-increasing order of magnitude
   - Specify a magnitude threshold, T, via the "elbow" selection algorithm
   - Soft-threshold the coefficients c:

$$d_{\mathrm{ST}}(i) = \begin{cases} \mathrm{sign}(c(i)) \cdot (|c(i)| - T) & \text{if } |c(i)| > T \\ 0 & \text{otherwise} \end{cases}$$

   - Note: keep all scaling coefficients intact

Denoising Algorithm

1. Construct the HGLET / GHWT dictionaries on the noisy signal
2. Choose a particular basis either automatically (e.g., the best basis) or manually (e.g., a basis at the fixed scale)
3. Soft-threshold the expansion coefficients, i.e.,
   - Sort the expansion coefficients in non-increasing order of magnitude
   - Specify a magnitude threshold, T, via the "elbow" selection algorithm
   - Soft-threshold the coefficients c:

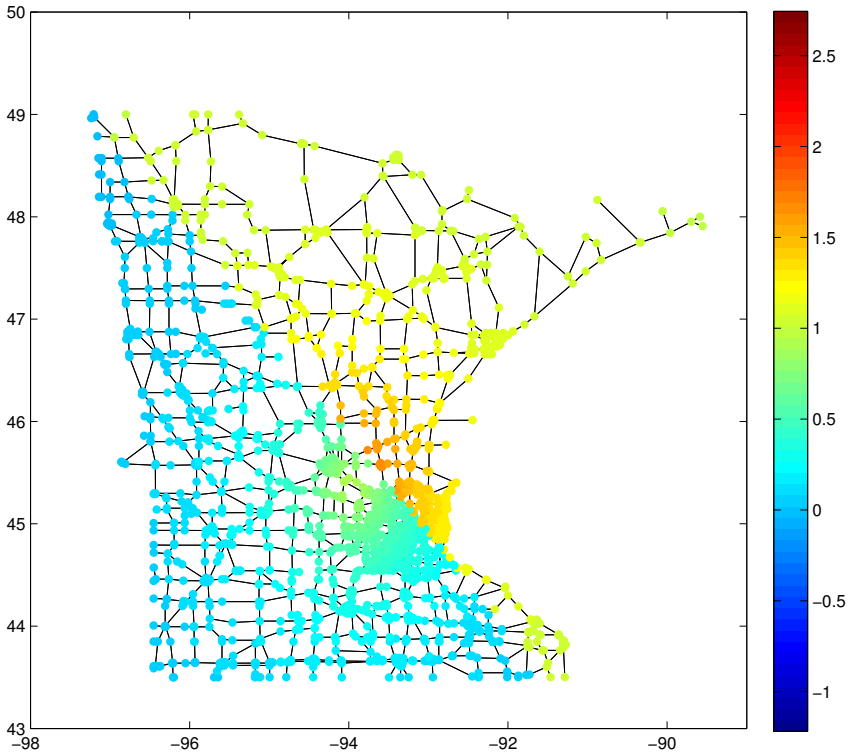
$$d_{\mathrm{ST}}(i) = \begin{cases} \mathrm{sign}(c(i)) \cdot (|c(i)| - T) & \text{if } |c(i)| > T \\ 0 & \text{otherwise} \end{cases}$$

   - Note: keep all scaling coefficients intact

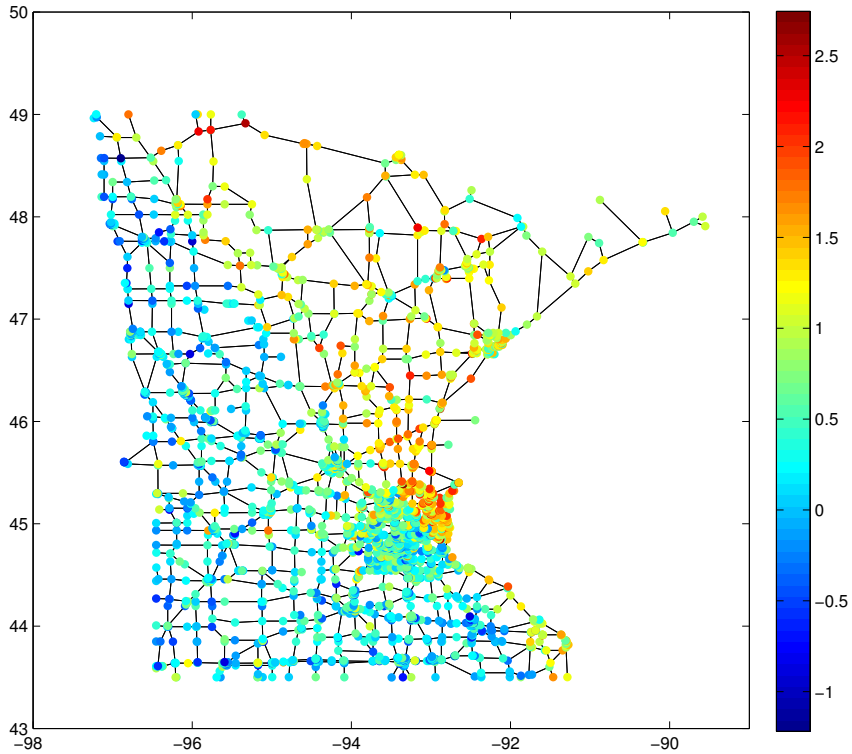
Preliminary Results (L_{rw}'s for Recursive Partitioning)

Transform	SNR (dB)	Coefficients Kept (%)
GHWT c2f BB	10.20	24.43
GHWT f2c BB	10.09	1.40
GHWT $j = 6$	**13.47**	3.87
GHWT $j = 0$ (= Walsh)	9.81	1.63
Haar $0 \le j \le j_{\max} = 14$	10.99	1.63
Haar $0 \le j \le 6$	12.03	2.43
HGLET BB (L)	10.15	24.32
HGLET $j = 6$ (L)	**14.01**	3.49
HGLET $j = 0$ (L)	11.06	1.33
HGLET BB (L_{rw})	4.85	95.33
HGLET $j = 6$ (L_{rw})	11.79	4.48
HGLET $j = 0$ (L_{rw})	11.18	2.69
HGLET BB (L_{sym})	5.65	30.84
HGLET $j = 6$ (L_{sym})	6.40	5.54
HGLET $j = 0$ (L_{sym})	5.60	3.15

Preliminary Results (L_{rw}'s for Recursive Partitioning) ...

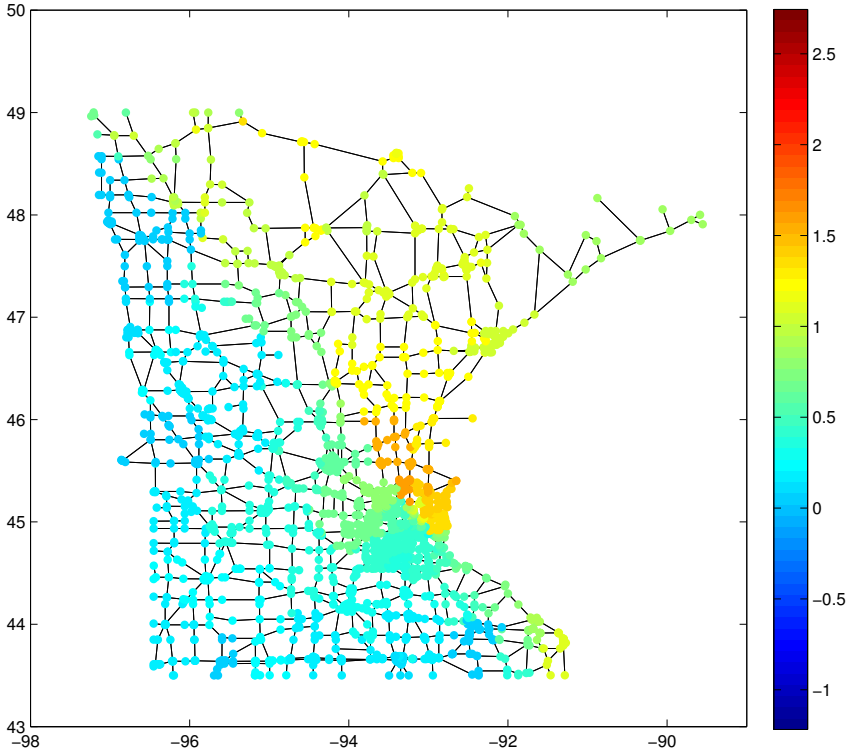


(a) Original: SNR = ∞

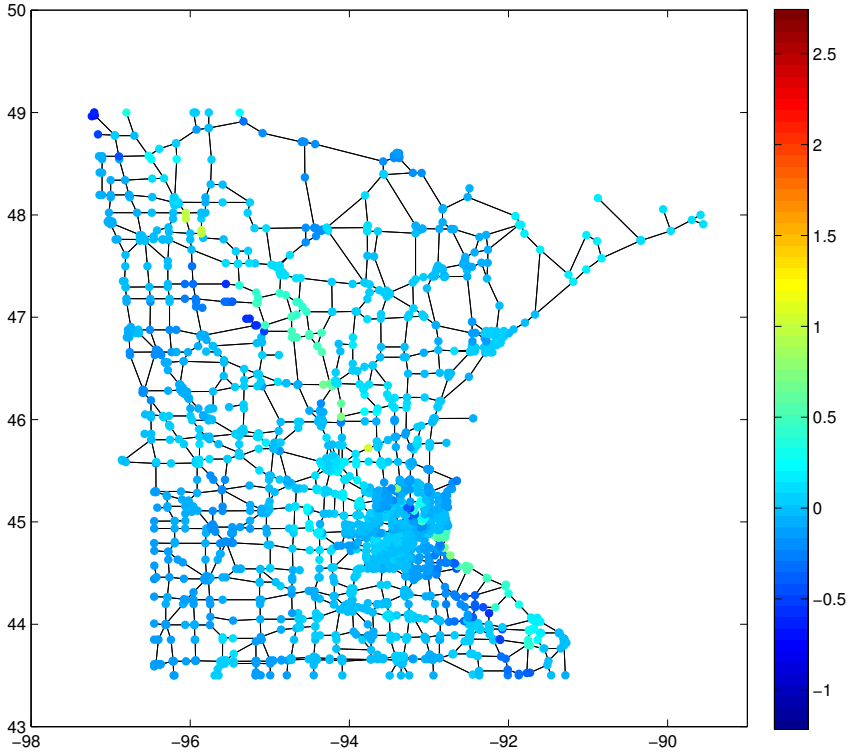


(b) Noisy signal: SNR = 5 dB

Preliminary Results (L_{rw}'s for Recursive Partitioning) ...

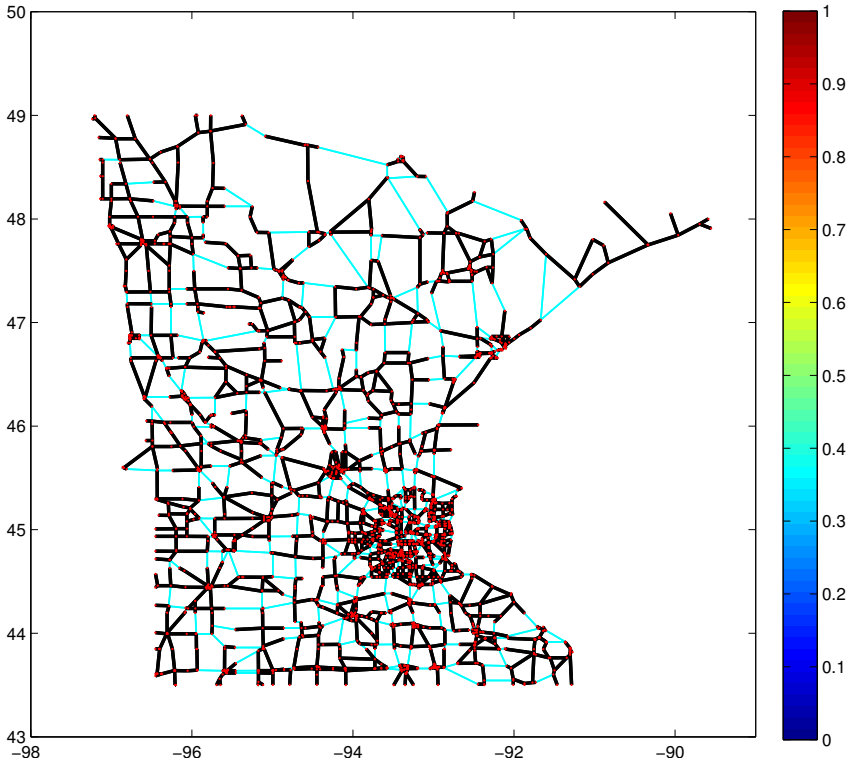


(a) HGLET $j = 6$ (L): SNR = 14.01 dB

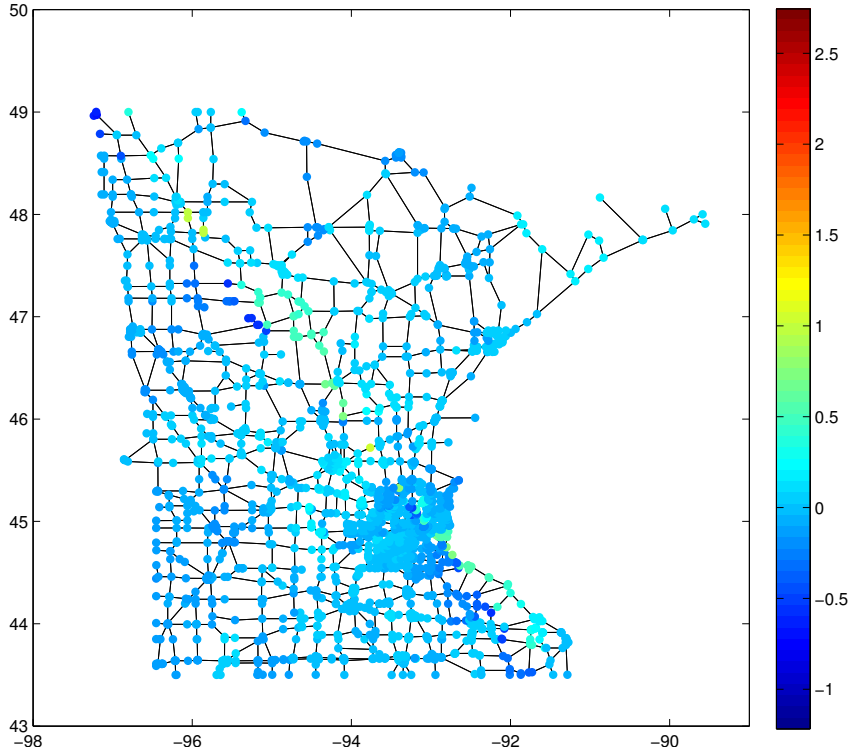


(b) Residual of HGLET $j = 6$ (L): SNR = 14.01 dB

Preliminary Results (L_{rw}'s for Recursive Partitioning) ...

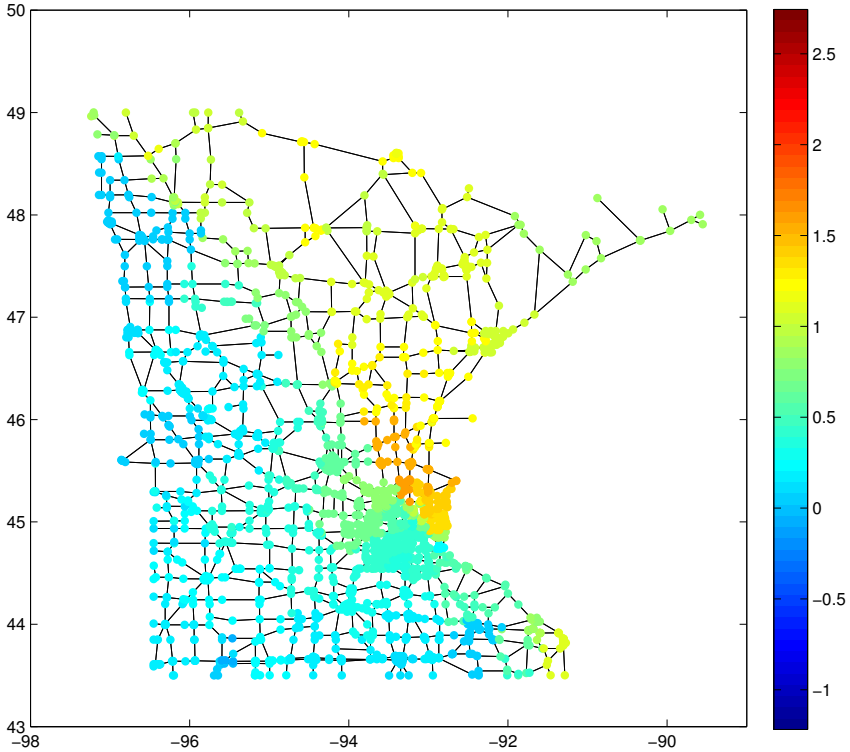


(a) Partition at $j = 6$

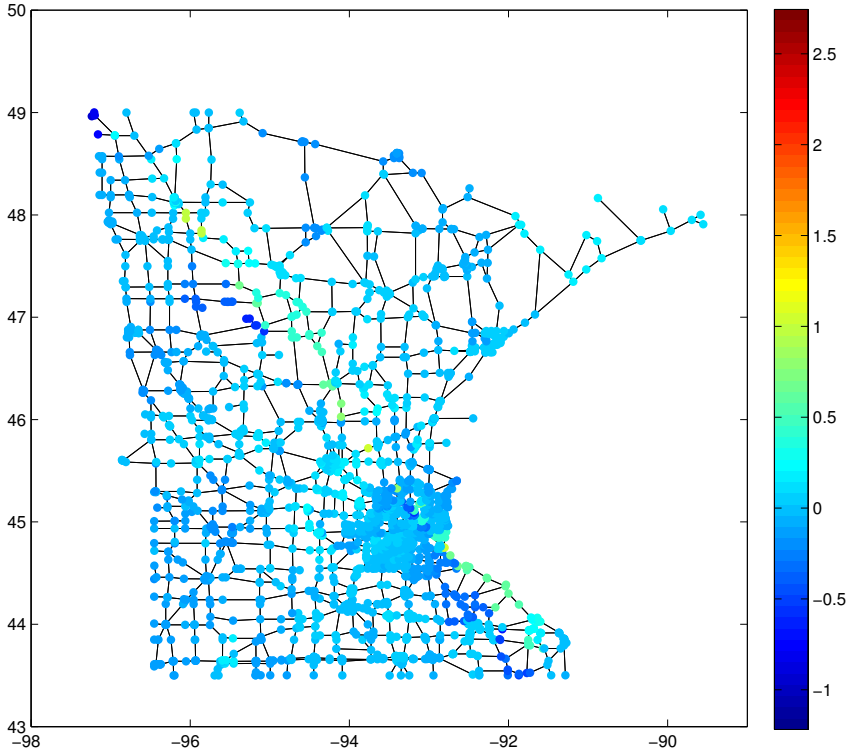


(b) Residual of HGLET $j = 6$ (L): SNR = 14.01 dB

Preliminary Results (L_{rw}'s for Recursive Partitioning) ...

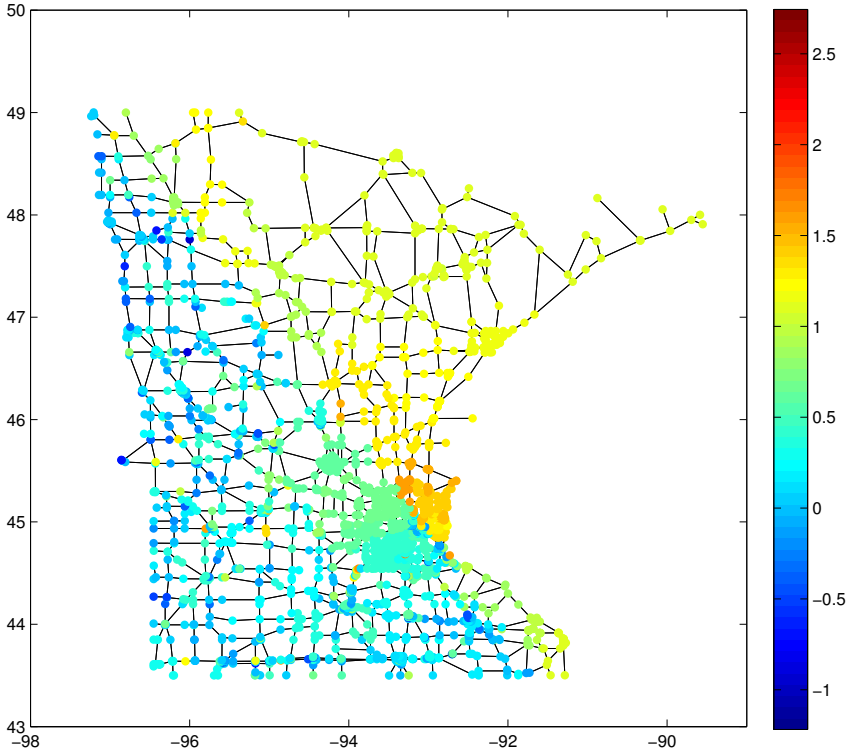


(a) GHWT $j = 6$: SNR = 13.47 dB

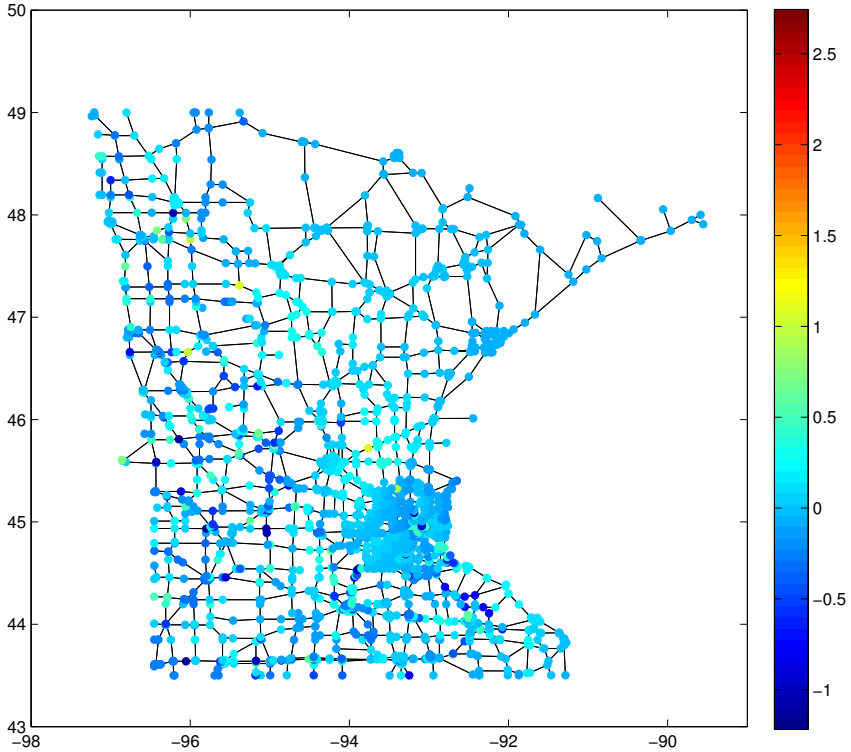


(b) Residual of GHWT $j = 6$: SNR = 13.47 dB

Preliminary Results (L_{rw}'s for Recursive Partitioning) ...

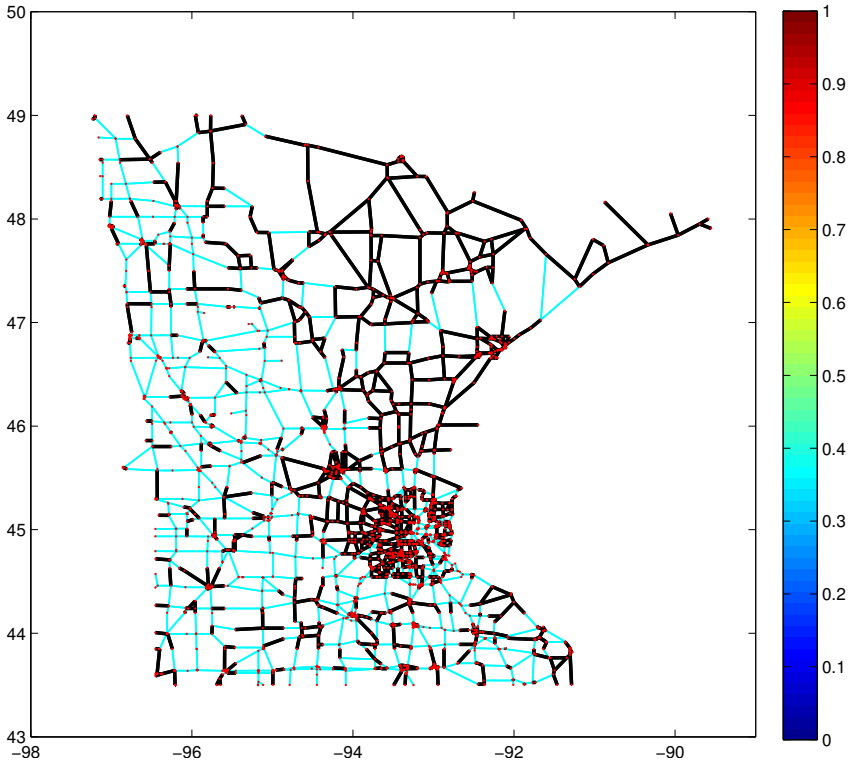


(a) HGLET BB (L): SNR = 10.15 dB

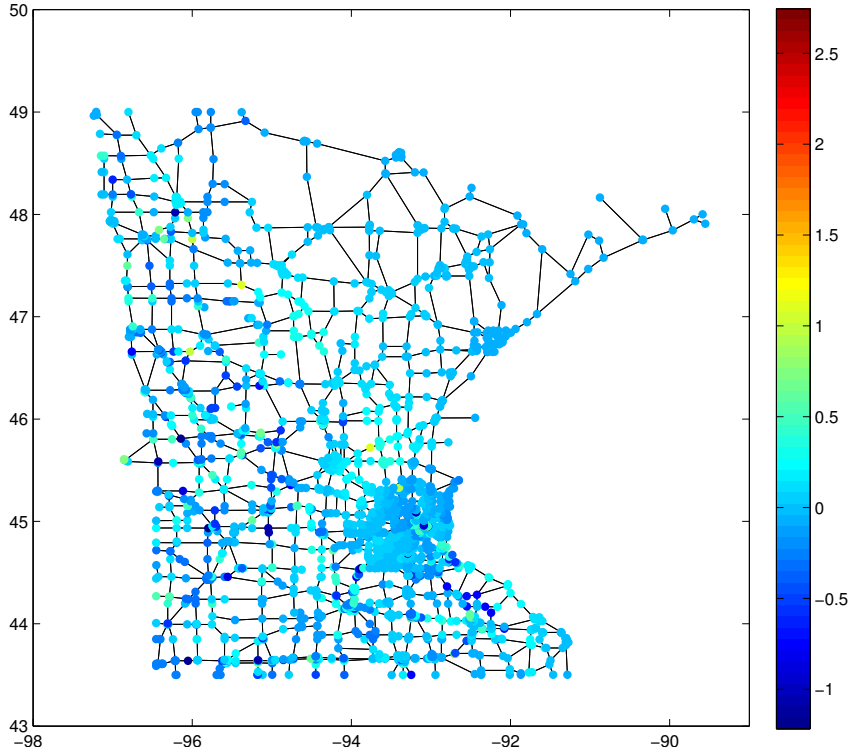


(b) Residual of HGLET BB (L): SNR = 10.15 dB

Preliminary Results (L_{rw}'s for Recursive Partitioning) ...

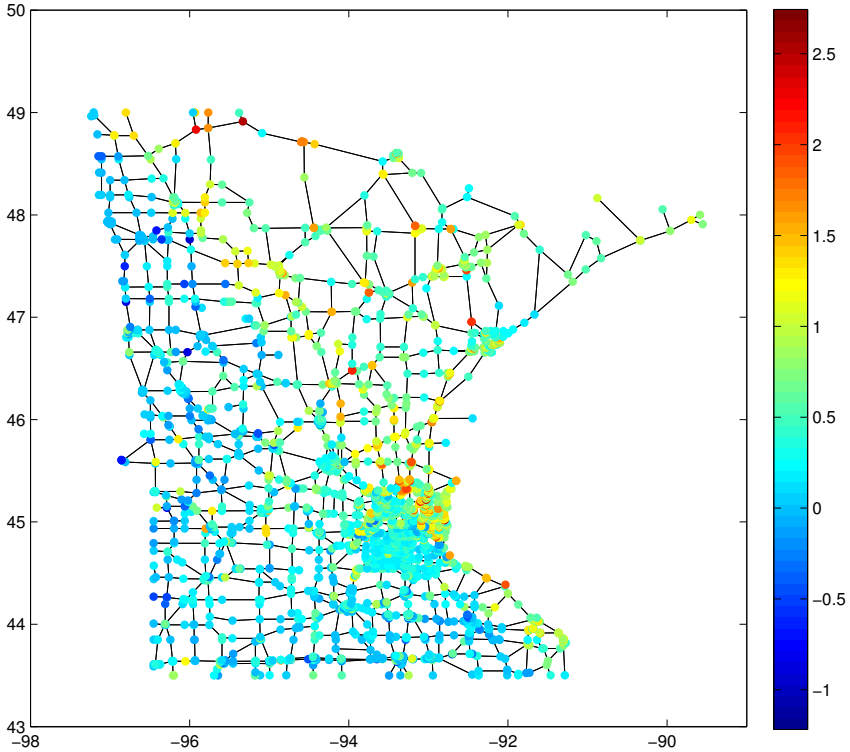


(a) HGLET BB (L) Partition

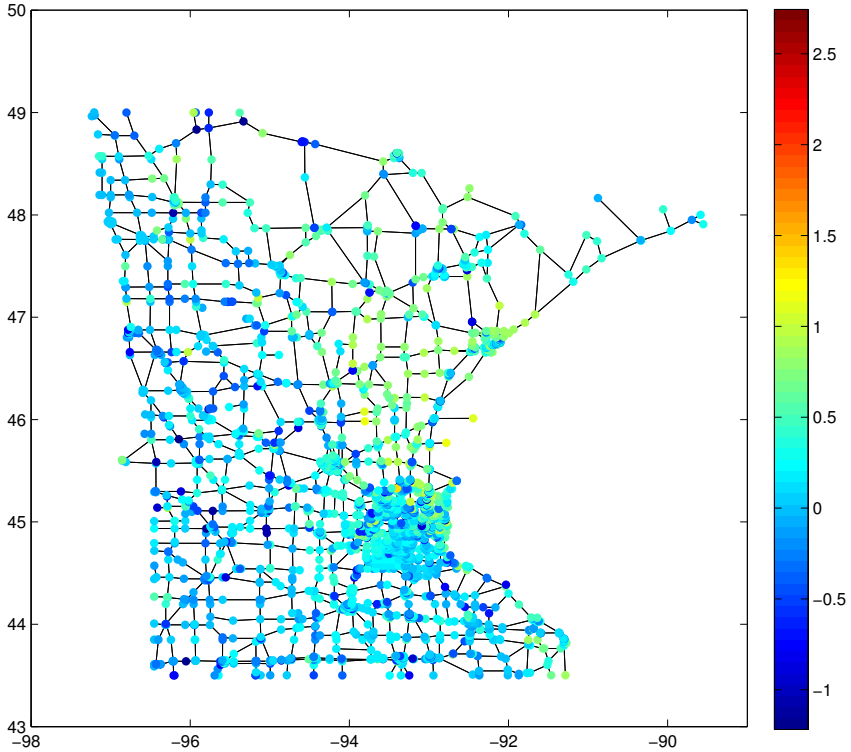


(b) Residual of HGLET BB (L): SNR = 10.15 dB

Preliminary Results (L_{rw}'s for Recursive Partitioning) ...

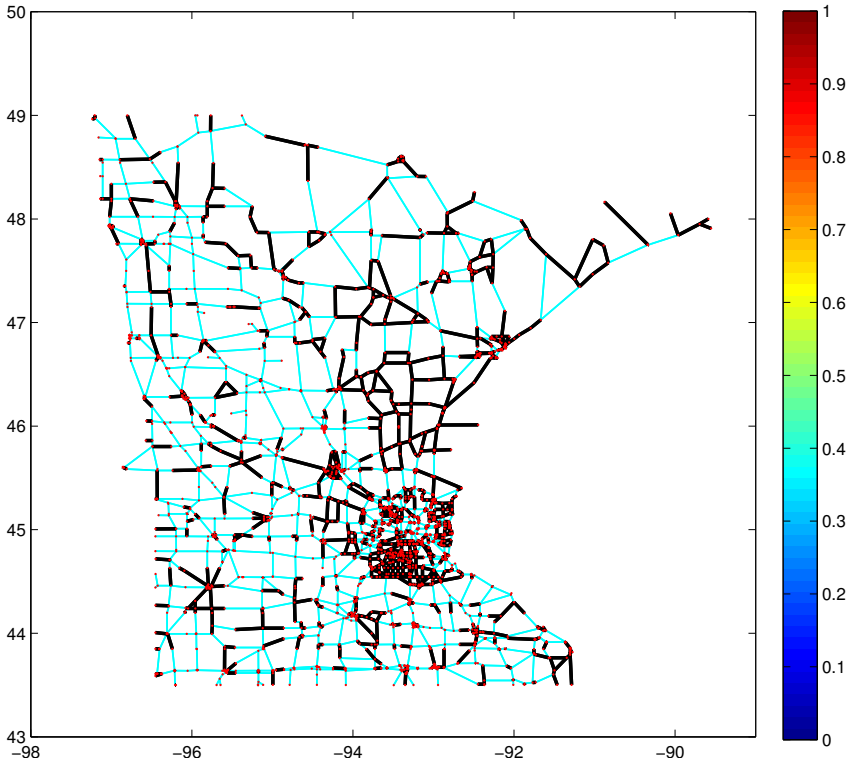


(a) HGLET BB (L_{sym}): SNR = 5.65 dB

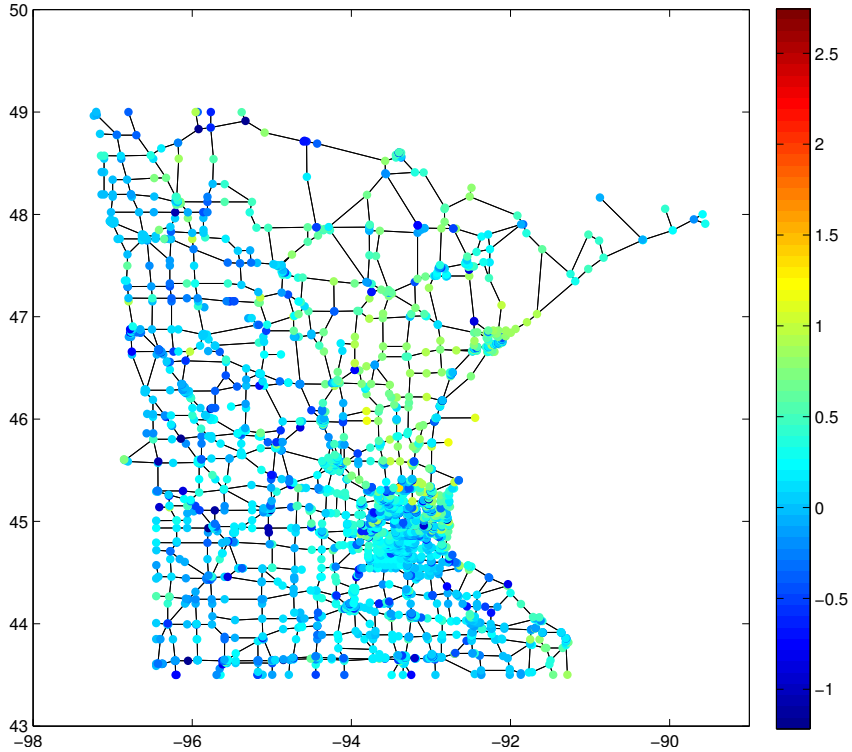


(b) Residual of HGLET BB (L_{sym}): SNR = 5.65 dB

Preliminary Results (L_{rw}'s for Recursive Partitioning) ...

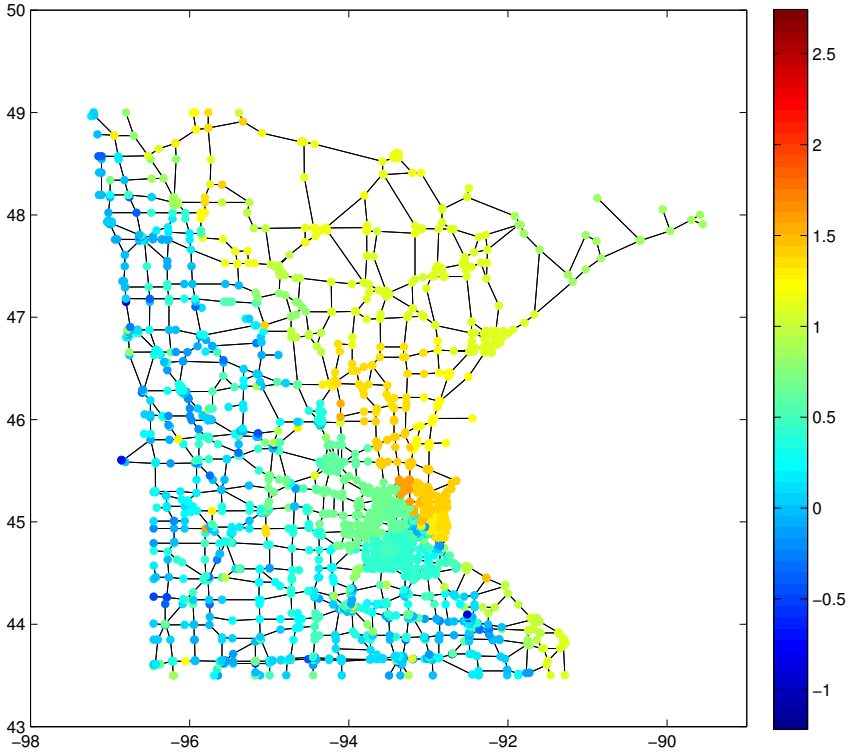


(a) HGLET BB (L_{sym}) Partition

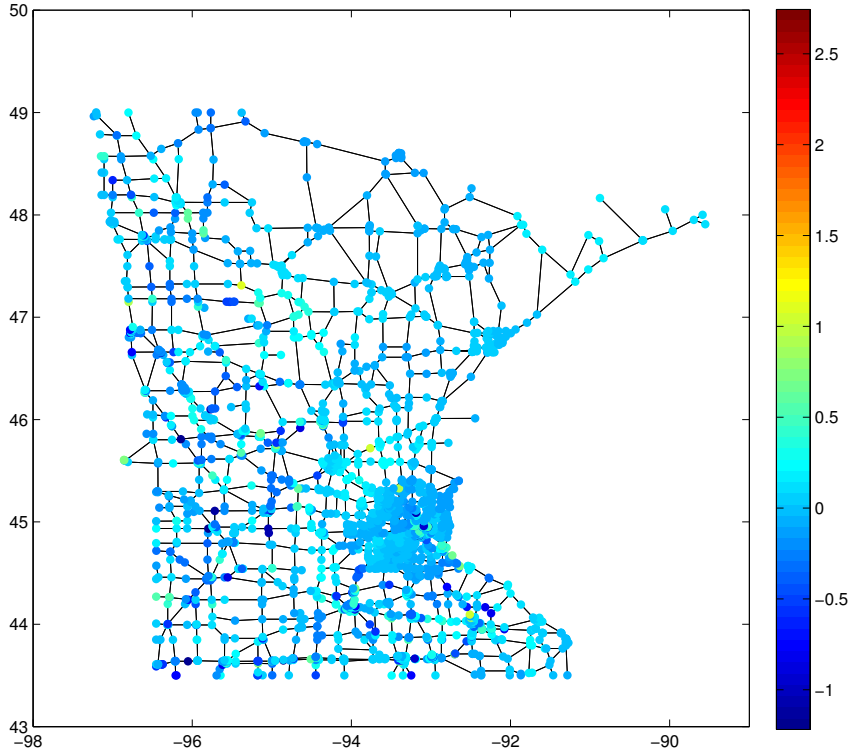


(b) Residual of HGLET BB (L_{sym}): SNR = 5.65 dB

Preliminary Results (L_{rw}'s for Recursive Partitioning) ...

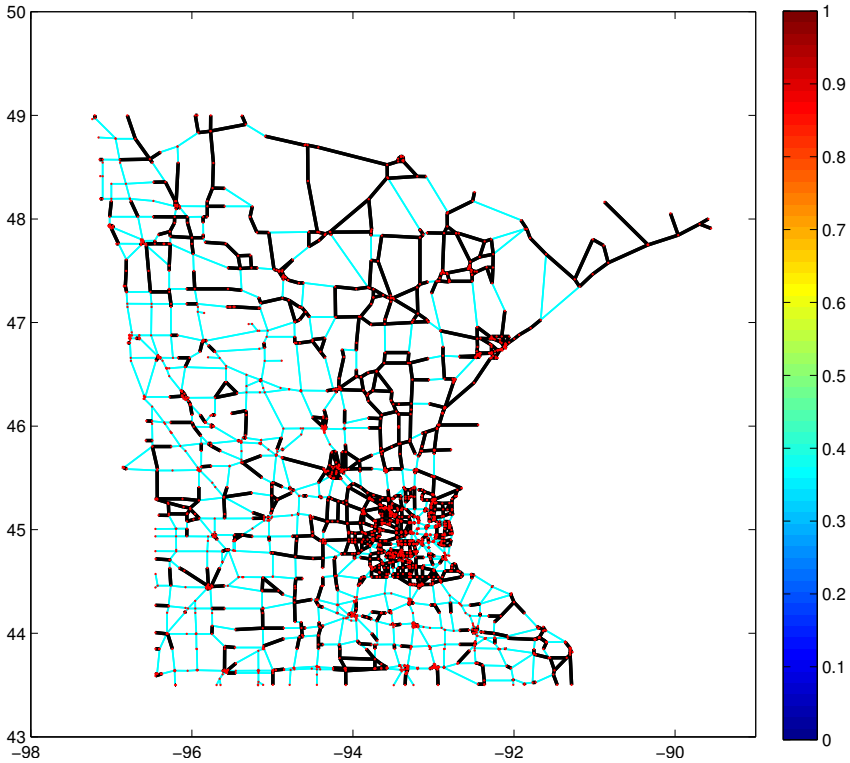


(a) GHWT c2f BB: SNR = 10.20 dB

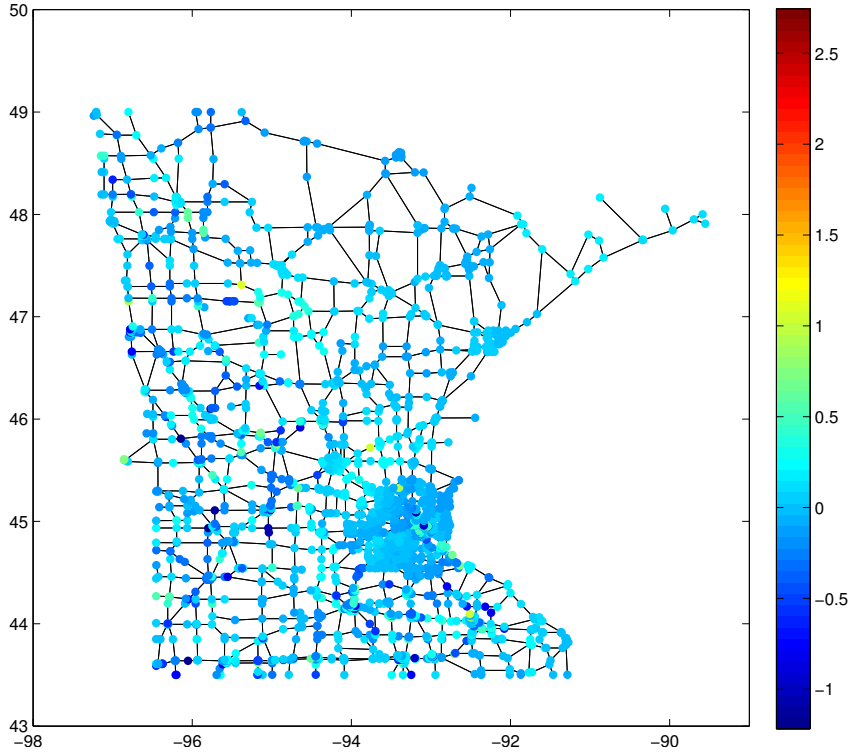


(b) Residual of GHWT c2f BB: SNR = 10.20 dB

Preliminary Results (L_{rw}'s for Recursive Partitioning) ...

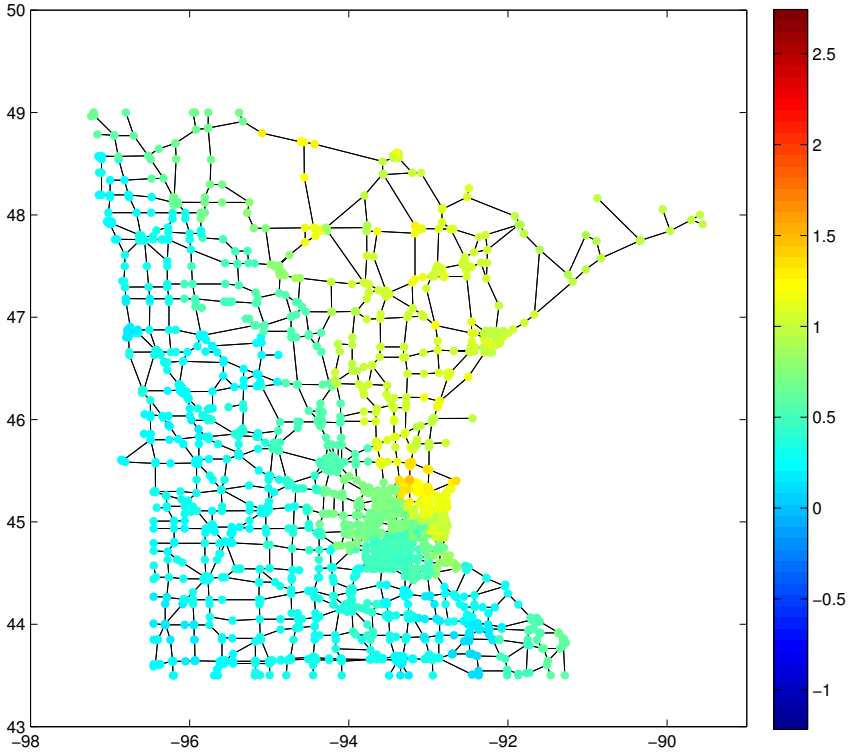


(a) GHWT c2f BB Partition

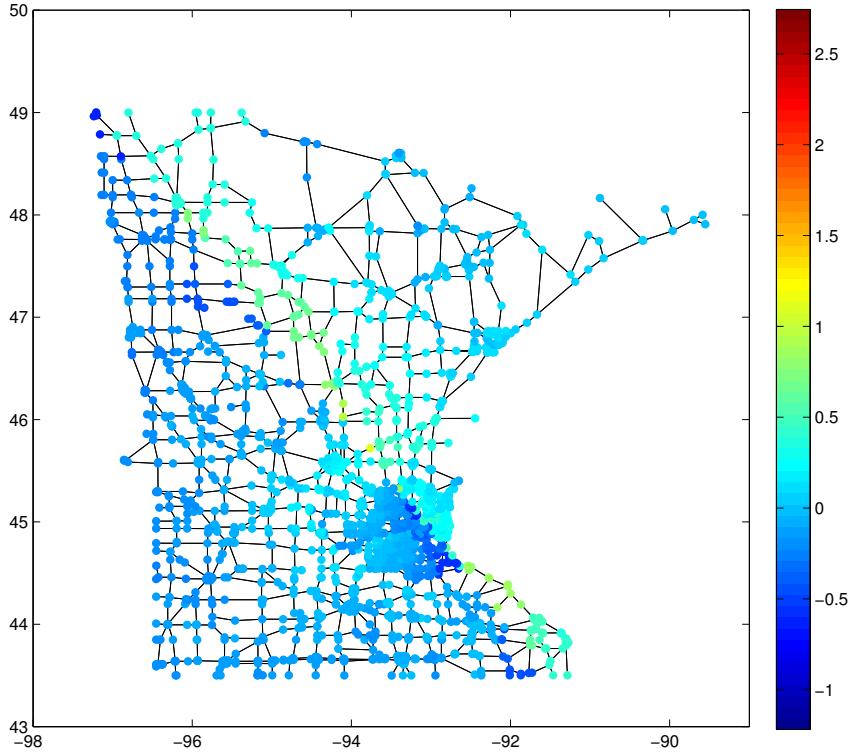


(b) Residual of GHWT c2f BB: SNR = 10.20 dB

Preliminary Results (L_{rw}'s for Recursive Partitioning) ...



(a) GHWT f2c BB: SNR = 10.09 dB



(b) Residual of GHWT f2c BB: SNR = 10.09 dB

Preliminary Results (L's for Recursive Partitioning)

Transform	SNR (dB)	Coefficients Kept (%)
GHWT c2f BB	9.75	25.30
GHWT f2c BB	10.77	1.33
GHWT $j = 6$	**12.81**	4.40
GHWT $j = 0$ (= Walsh)	10.05	1.37
Haar $0 \le j \le j_{\max} = 14$	11.29	1.48
Haar $0 \le j \le 6$	11.63	2.43
HGLET BB (L)	9.93	26.25
HGLET $j = 6$ (L)	**13.29**	3.98
HGLET $j = 0$ (L)	11.06	1.33
HGLET BB (L_{rw})	5.02	98.07
HGLET $j = 6$ (L_{rw})	11.56	4.21
HGLET $j = 0$ (L_{rw})	11.18	2.69
HGLET BB (L_{sym})	5.24	27.35
HGLET $j = 6$ (L_{sym})	6.11	5.27
HGLET $j = 0$ (L_{sym})	5.60	3.15

Observations

- Overall, the bases at the fixed level $j = 6$ performed best for this dataset whereas the best bases performed relatively poor.
- This is because at $j = 6$ the partition turned out to be *just right* for removing noise: small enough to capture details, but large enough to drown out noise.
- The best bases with the sparsity criterion with $\ell^{0.1}$ norm seem to have adjusted to noises.
- Results were not overly sensitive between the recursive partitioning based on the Fiedler vectors of L matrices and L_{rw} matrices.

Observations

- Overall, the bases at the fixed level $j = 6$ performed best for this dataset whereas the best bases performed relatively poor.
- This is because at $j = 6$ the partition turned out to be *just right* for removing noise: small enough to capture details, but large enough to drown out noise.
- The best bases with the sparsity criterion with $\ell^{0.1}$ norm seem to have adjusted to noises.
- Results were not overly sensitive between the recursive partitioning based on the Fiedler vectors of L matrices and L_{rw} matrices.

Observations

- Overall, the bases at the fixed level $j = 6$ performed best for this dataset whereas the best bases performed relatively poor.
- This is because at $j = 6$ the partition turned out to be *just right* for removing noise: small enough to capture details, but large enough to drown out noise.
- The best bases with the sparsity criterion with $\ell^{0.1}$ norm seem to have adjusted to noises.
- Results were not overly sensitive between the recursive partitioning based on the Fiedler vectors of L matrices and L_{rw} matrices.

Observations

- Overall, the bases at the fixed level $j = 6$ performed best for this dataset whereas the best bases performed relatively poor.
- This is because at $j = 6$ the partition turned out to be *just right* for removing noise: small enough to capture details, but large enough to drown out noise.
- The best bases with the sparsity criterion with $\ell^{0.1}$ norm seem to have adjusted to noises.
- Results were not overly sensitive between the recursive partitioning based on the Fiedler vectors of L matrices and L_{rw} matrices.

Outline

Discussions on Potential Agricultural Applications

- What kind of sensor networks are currently being used for "Green and Clean Food Productions"?
- What kind of sensor networks should be used "Green and Clean Food Productions" in the future?
- What are the aspects/features of measured data you want to obtain?
- What kind of mathematical and software tools do you need to extract such information?
- How will you utilize such information for "Green and Clean Food Productions"?
- What would be the cost of deploying such sensor networks in the fields?

Discussions on Potential Agricultural Applications

- What kind of sensor networks are currently being used for "Green and Clean Food Productions"?
- What kind of sensor networks should be used "Green and Clean Food Productions" in the future?
- What are the aspects/features of measured data you want to obtain?
- What kind of mathematical and software tools do you need to extract such information?
- How will you utilize such information for "Green and Clean Food Productions"?
- What would be the cost of deploying such sensor networks in the fields?

Discussions on Potential Agricultural Applications

- What kind of sensor networks are currently being used for "Green and Clean Food Productions"?
- What kind of sensor networks should be used "Green and Clean Food Productions" in the future?
- What are the aspects/features of measured data you want to obtain?
- What kind of mathematical and software tools do you need to extract such information?
- How will you utilize such information for "Green and Clean Food Productions"?
- What would be the cost of deploying such sensor networks in the fields?

Discussions on Potential Agricultural Applications

- What kind of sensor networks are currently being used for "Green and Clean Food Productions"?
- What kind of sensor networks should be used "Green and Clean Food Productions" in the future?
- What are the aspects/features of measured data you want to obtain?
- What kind of mathematical and software tools do you need to extract such information?
- How will you utilize such information for "Green and Clean Food Productions"?
- What would be the cost of deploying such sensor networks in the fields?

Discussions on Potential Agricultural Applications

- What kind of sensor networks are currently being used for "Green and Clean Food Productions"?
- What kind of sensor networks should be used "Green and Clean Food Productions" in the future?
- What are the aspects/features of measured data you want to obtain?
- What kind of mathematical and software tools do you need to extract such information?
- How will you utilize such information for "Green and Clean Food Productions"?
- What would be the cost of deploying such sensor networks in the fields?

Discussions on Potential Agricultural Applications

- What kind of sensor networks are currently being used for "Green and Clean Food Productions"?
- What kind of sensor networks should be used "Green and Clean Food Productions" in the future?
- What are the aspects/features of measured data you want to obtain?
- What kind of mathematical and software tools do you need to extract such information?
- How will you utilize such information for "Green and Clean Food Productions"?
- What would be the cost of deploying such sensor networks in the fields?

Report on Potential Agricultural Applications of Graph Data Analysis

- Write your thoughts on a potential agricultural application of the concepts and tools you learned from my lectures (you can further elaborate some of the topics and questions raised in the previous page)
- Submit your via email to `saito@math.ucdavis.edu`
- Page limit: 5 pages or less
- File Format: PDF (preferrable) or MS Word
- **Deadline: 5pm, September 15, 2014**

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 Transform* and the *Walsh-Hadamard Transform*.
- Both of these transforms allow us to choose an orthonormal basis suitable for the task at hand, e.g., approximation, classification, regression, ...
- They may also be useful for regularly-sampled signals, e.g., can deal with signals of non-dyadic length; adaptive segmentation, ...
- Developing harmonic analysis tools for directed graphs will be challenging $\Longrightarrow$ our idea: use *distance matrix + SVD* instead; to be continued!
- Connect to lots of interesting mathematics and applications: *harmonic analysis, discrete mathematics, mathematical physics, PDEs, differential geometry, signal & image processing, statistics*, ...

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).

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

- N. Saito & J.-F. Remy: "The polyharmonic local sine transform: A new tool for local image analysis and synthesis without edge effect," *Applied & Computational Harmonic Analysis*, vol. 20, no. 1, pp. 41-73, 2006.
- N. Saito: "Data analysis and representation using eigenfunctions of Laplacian on a general domain," *Applied & Computational Harmonic Analysis*, vol. 25, no. 1, pp. 68–97, 2008.
- N. Saito & E. Woei: "Analysis of neuronal dendrite patterns using eigenvalues of graph Laplacians," *Japan SIAM Letters*, vol. 1, pp. 13–16, 2009.
- Y. Nakatsukasa, N. Saito, & E. Woei: "Mysteries around graph Laplacian eigenvalue 4," *Linear Algebra & Its Applications*, vol. 438, no. 8, pp. 3231–3246, 2013.
- J. Irion & N. Saito: "Hierarchical graph Laplacian eigen transforms," *Japan SIAM Letters*, vol. 6, pp. 21–24, 2014.
- J. Irion & N. Saito: "The generalized Haar-Walsh transform," *Proc. 2014 IEEE Workshop on Statistical Signal Processing*, pp. 488-491, 2014.

Thank you very much for your attention!