MAT 280: Harmonic Analysis on Graphs & Networks
Lecture 19: Wavelets on Graphs IV
Hierarchical Graph Laplacian Eigen Transform(HGLET)

Naoki Saito

Department of Mathematics
University of California, Davis

December 3, 2019

Outline

Acknowledgment

- Jeff Irion (formerly UC Davis; currently Neato Robotics, Inc.)
- NSF Grants: DMS-1418779, IIS-1631329, DMS-1912747, CCF-1934568
- ONR Grants: N00014-16-1-2255

Outline

Motivation: Building Multiscale Graph Basis Dictionaries

- This lecture is based on the following papers:
  - J. Irion & N. Saito: "Hierarchical graph Laplacian eigen transforms," *JSIAM Letters*, vol.6, pp.21–24, 2014.
  - J. Irion & N. Saito: "Applied and computational harmonic analysis on graphs and networks," in *Wavelets and Sparsity XVI, Proc. SPIE 9597*, Paper # 95971F, 2015.
- *Wavelets* have been quite successful on regular domains
- They have been extended to irregular domains ⇒ "2nd Generation Wavelets" including graphs, e.g.:
  - Coifman and Maggioni (2006): diffusion wavelets; Bremer et al. (2006): diffusion wavelet *packets*
  - Jansen, Nason, and Silverman (2008): Adaptation of the *lifting scheme* to graphs
  - Hammond, Vandergheynst, and Gribonval (2011): Spectral graph wavelet transforms (via spectral graph theory)
  - Sharon and Shkolnisky (2015): using a subset of the Laplacian eigenvectors and a recursive partitioning tree to construct a *multiresolution analysis* and consequently *multiwavelet bases*
  - ...

Motivation: Building Multiscale Graph Basis Dictionaries

- This lecture is based on the following papers:
  - J. Irion & N. Saito: "Hierarchical graph Laplacian eigen transforms," *JSIAM Letters*, vol.6, pp.21–24, 2014.
  - J. Irion & N. Saito: "Applied and computational harmonic analysis on graphs and networks," in *Wavelets and Sparsity XVI, Proc. SPIE 9597*, Paper # 95971F, 2015.
- *Wavelets* have been quite successful on regular domains
- They have been extended to irregular domains ⇒ "2nd Generation Wavelets" including graphs, e.g.:
  - Coifman and Maggioni (2006): diffusion wavelets; Bremer et al. (2006): diffusion wavelet *packets*
  - Jansen, Nason, and Silverman (2008): Adaptation of the *lifting scheme* to graphs
  - Hammond, Vandergheynst, and Gribonval (2011): Spectral graph wavelet transforms (via spectral graph theory)
  - Sharon and Shkolnisky (2015): using a subset of the Laplacian eigenvectors and a recursive partitioning tree to construct a *multiresolution analysis* and consequently *multiwavelet bases*
  - ...

Motivation: Building Multiscale Graph Basis Dictionaries

- This lecture is based on the following papers:
  - J. Irion & N. Saito: "Hierarchical graph Laplacian eigen transforms," *JSIAM Letters*, vol.6, pp.21–24, 2014.
  - J. Irion & N. Saito: "Applied and computational harmonic analysis on graphs and networks," in *Wavelets and Sparsity XVI, Proc. SPIE 9597*, Paper # 95971F, 2015.
- *Wavelets* have been quite successful on regular domains
- They have been extended to irregular domains ⇒ "2nd Generation Wavelets" including graphs, e.g.:
  - Coifman and Maggioni (2006): diffusion wavelets; Bremer et al. (2006): diffusion wavelet *packets*
  - Jansen, Nason, and Silverman (2008): Adaptation of the *lifting scheme* to graphs
  - Hammond, Vandergheynst, and Gribonval (2011): Spectral graph wavelet transforms (via spectral graph theory)
  - Sharon and Shkolnisky (2015): using a subset of the Laplacian eigenvectors and a recursive partitioning tree to construct a *multiresolution analysis* and consequently *multiwavelet bases*
  - ...

Motivation: Building Multiscale Graph Basis Dictionaries

- This lecture is based on the following papers:
  - J. Irion & N. Saito: "Hierarchical graph Laplacian eigen transforms," *JSIAM Letters*, vol.6, pp.21–24, 2014.
  - J. Irion & N. Saito: "Applied and computational harmonic analysis on graphs and networks," in *Wavelets and Sparsity XVI, Proc. SPIE 9597*, Paper # 95971F, 2015.
- *Wavelets* have been quite successful on regular domains
- They have been extended to irregular domains ⇒ "2nd Generation Wavelets" including graphs, e.g.:
  - Coifman and Maggioni (2006): diffusion wavelets; Bremer et al. (2006): diffusion wavelet *packets*
  - Jansen, Nason, and Silverman (2008): Adaptation of the *lifting scheme* to graphs
  - Hammond, Vandergheynst, and Gribonval (2011): Spectral graph wavelet transforms (via spectral graph theory)
  - Sharon and Shkolnisky (2015): using a subset of the Laplacian eigenvectors and a recursive partitioning tree to construct a *multiresolution analysis* and consequently *multiwavelet bases*
  - ...

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 "sines" and "cosines" on graphs, has been quite popular
- However, they exhibit peculiar behaviors depending on *topology* and *structure* of given graphs!

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 "sines" and "cosines" on graphs, has been quite popular
- However, they exhibit peculiar behaviors depending on *topology* and *structure* of given graphs!

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 "sines" and "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 $j=0$, $|V|=n_0^0$)
2. Partition the graph using the Fiedler vector $\boldsymbol{\phi}_{k,1}^{j}$
3. Generate an orthonormal basis for each of the partitions ⇒ Laplacian eigenvectors
4. Repeat...
5. Select an orthonormal basis from this collection of orthonormal bases

$$\left[\begin{array}{cccc} \boldsymbol{\phi}_{0,0}^{0} & \boldsymbol{\phi}_{0,1}^{0} & \boldsymbol{\phi}_{0,2}^{0} & \cdots & \boldsymbol{\phi}_{0,n_0^0-1}^{0} \end{array}\right]$$

1. Generate an orthonormal basis for the entire graph $\Rightarrow$ Laplacian eigenvectors (Notation is $\boldsymbol{\phi}_{k,l}^{j}$ with $j=0$, $|V| = n_0^0$)
2. Partition the graph using the Fiedler vector $\boldsymbol{\phi}_{k,1}^{j}$
3. Generate an orthonormal basis for each of the partitions $\Rightarrow$ Laplacian eigenvectors
4. Repeat...
5. Select an orthonormal basis from this collection of orthonormal bases

$$\left[\begin{array}{cccc} \boldsymbol{\phi}_{0,0}^{0} & \boldsymbol{\phi}_{0,1}^{0} & \boldsymbol{\phi}_{0,2}^{0} & \cdots & \boldsymbol{\phi}_{0,n_0^0-1}^{0} \end{array} \right]$$

1. Generate an orthonormal basis for the entire graph ⇒ Laplacian eigenvectors (Notation is $\boldsymbol{\phi}_{k,l}^{j}$ with $j=0$, $|V|=n_0^0$)
2. Partition the graph using the Fiedler vector $\boldsymbol{\phi}_{k,1}^{j}$
3. Generate an orthonormal basis for each of the partitions ⇒ Laplacian eigenvectors
4. Repeat...
5. Select an orthonormal basis from this collection of orthonormal bases

$$\left[\begin{array}{ccccc} \boldsymbol{\phi}_{0,0}^{0} & \boldsymbol{\phi}_{0,1}^{0} & \boldsymbol{\phi}_{0,2}^{0} & \cdots & \boldsymbol{\phi}_{0,n_0^0-1}^{0} \end{array}\right]$$

$$\left[\begin{array}{ccccc} \boldsymbol{\phi}_{0,0}^{1} & \boldsymbol{\phi}_{0,1}^{1} & \boldsymbol{\phi}_{0,2}^{1} & \cdots & \boldsymbol{\phi}_{0,n_0^1-1}^{1} \end{array}\right] \quad \left[\begin{array}{ccccc} \boldsymbol{\phi}_{1,0}^{1} & \boldsymbol{\phi}_{1,1}^{1} & \boldsymbol{\phi}_{1,2}^{1} & \cdots & \boldsymbol{\phi}_{1,n_1^1-1}^{1} \end{array}\right]$$

1. Generate an orthonormal basis for the entire graph ⇒ Laplacian eigenvectors (Notation is $\boldsymbol{\phi}_{k,l}^{j}$ with $j=0$, $|V|=n_0^0$)
2. Partition the graph using the Fiedler vector $\boldsymbol{\phi}_{k,1}^{j}$
3. Generate an orthonormal basis for each of the partitions ⇒ Laplacian eigenvectors
4. Repeat...
5. Select an orthonormal basis from this collection of orthonormal bases

$$\left[\begin{array}{ccccc} \boldsymbol{\phi}_{0,0}^{0} & \boldsymbol{\phi}_{0,1}^{0} & \boldsymbol{\phi}_{0,2}^{0} & \cdots & \boldsymbol{\phi}_{0,n_0^0-1}^{0} \end{array}\right]$$

$$\left[\begin{array}{ccccc} \boldsymbol{\phi}_{0,0}^{1} & \boldsymbol{\phi}_{0,1}^{1} & \boldsymbol{\phi}_{0,2}^{1} & \cdots & \boldsymbol{\phi}_{0,n_0^1-1}^{1} \end{array}\right] \quad \left[\begin{array}{ccccc} \boldsymbol{\phi}_{1,0}^{1} & \boldsymbol{\phi}_{1,1}^{1} & \boldsymbol{\phi}_{1,2}^{1} & \cdots & \boldsymbol{\phi}_{1,n_1^1-1}^{1} \end{array}\right]$$

1. Generate an orthonormal basis for the entire graph ⇒ Laplacian eigenvectors (Notation is $\boldsymbol{\phi}_{k,l}^{j}$ with $j=0$, $|V| = n_0^0$)
2. Partition the graph using the Fiedler vector $\boldsymbol{\phi}_{k,1}^{j}$
3. Generate an orthonormal basis for each of the partitions ⇒ Laplacian eigenvectors
4. Repeat...
5. Select an orthonormal basis from this collection of orthonormal bases

$$\left[\begin{array}{ccccc} \boldsymbol{\phi}_{0,0}^{0} & \boldsymbol{\phi}_{0,1}^{0} & \boldsymbol{\phi}_{0,2}^{0} & \cdots & \boldsymbol{\phi}_{0,n_0^0-1}^{0} \end{array} \right]$$

$$\left[\begin{array}{ccccc} \boldsymbol{\phi}_{0,0}^{1} & \boldsymbol{\phi}_{0,1}^{1} & \boldsymbol{\phi}_{0,2}^{1} & \cdots & \boldsymbol{\phi}_{0,n_0^1-1}^{1} \end{array} \right] \quad \left[\begin{array}{ccccc} \boldsymbol{\phi}_{1,0}^{1} & \boldsymbol{\phi}_{1,1}^{1} & \boldsymbol{\phi}_{1,2}^{1} & \cdots & \boldsymbol{\phi}_{1,n_1^1-1}^{1} \end{array} \right]$$

1. Generate an orthonormal basis for the entire graph ⇒ Laplacian eigenvectors (Notation is $\boldsymbol{\phi}_{k,l}^{j}$ with $j = 0$, $|V| = n_0^0$)
2. Partition the graph using the Fiedler vector $\boldsymbol{\phi}_{k,1}^{j}$
3. Generate an orthonormal basis for each of the partitions ⇒ Laplacian eigenvectors
4. Repeat...
5. Select an orthonormal basis from this collection of orthonormal bases

$$\left[\begin{array}{cccc} \boldsymbol{\phi}_{0,0}^{0} & \boldsymbol{\phi}_{0,1}^{0} & \boldsymbol{\phi}_{0,2}^{0} & \cdots \quad \boldsymbol{\phi}_{0,n_0^0-1}^{0} \end{array} \right]$$

$$\left[\begin{array}{cccc} \boldsymbol{\phi}_{0,0}^{1} & \boldsymbol{\phi}_{0,1}^{1} & \boldsymbol{\phi}_{0,2}^{1} & \cdots \quad \boldsymbol{\phi}_{0,n_0^1-1}^{1} \end{array} \right] \quad \left[\begin{array}{cccc} \boldsymbol{\phi}_{1,0}^{1} & \boldsymbol{\phi}_{1,1}^{1} & \boldsymbol{\phi}_{1,2}^{1} & \cdots \quad \boldsymbol{\phi}_{1,n_1^1-1}^{1} \end{array} \right]$$

$$\left[\boldsymbol{\phi}_{0,0}^{2} \boldsymbol{\phi}_{0,1}^{2} \cdots \boldsymbol{\phi}_{0,n_0^2-1}^{2} \right] \left[\boldsymbol{\phi}_{1,0}^{2} \boldsymbol{\phi}_{1,1}^{2} \cdots \boldsymbol{\phi}_{1,n_1^2-1}^{2} \right] \left[\boldsymbol{\phi}_{2,0}^{2} \boldsymbol{\phi}_{2,1}^{2} \cdots \boldsymbol{\phi}_{2,n_2^2-1}^{2} \right] \left[\boldsymbol{\phi}_{3,0}^{2} \boldsymbol{\phi}_{3,1}^{2} \cdots \boldsymbol{\phi}_{3,n_3^2-1}^{2} \right]$$

1. Generate an orthonormal basis for the entire graph ⇒ Laplacian eigenvectors (Notation is $\boldsymbol{\phi}_{k,l}^{j}$ with $j=0$, $|V|=n_0^0$)
2. Partition the graph using the Fiedler vector $\boldsymbol{\phi}_{k,1}^{j}$
3. Generate an orthonormal basis for each of the partitions ⇒ Laplacian eigenvectors
4. Repeat...
5. Select an orthonormal basis from this collection of orthonormal bases

$$\left[\begin{array}{ccccc} \boldsymbol{\phi}_{0,0}^{0} & \boldsymbol{\phi}_{0,1}^{0} & \boldsymbol{\phi}_{0,2}^{0} & \cdots & \boldsymbol{\phi}_{0,n_0^0-1}^{0} \end{array}\right]$$

$$\left[\begin{array}{ccccc} \boldsymbol{\phi}_{0,0}^{1} & \boldsymbol{\phi}_{0,1}^{1} & \boldsymbol{\phi}_{0,2}^{1} & \cdots & \boldsymbol{\phi}_{0,n_0^1-1}^{1} \end{array}\right] \quad \left[\begin{array}{ccccc} \boldsymbol{\phi}_{1,0}^{1} & \boldsymbol{\phi}_{1,1}^{1} & \boldsymbol{\phi}_{1,2}^{1} & \cdots & \boldsymbol{\phi}_{1,n_1^1-1}^{1} \end{array}\right]$$

$$\left[\boldsymbol{\phi}_{0,0}^{2}\, \boldsymbol{\phi}_{0,1}^{2} \cdots \boldsymbol{\phi}_{0,n_0^2-1}^{2}\right] \left[\boldsymbol{\phi}_{1,0}^{2}\, \boldsymbol{\phi}_{1,1}^{2} \cdots \boldsymbol{\phi}_{1,n_1^2-1}^{2}\right] \left[\boldsymbol{\phi}_{2,0}^{2}\, \boldsymbol{\phi}_{2,1}^{2} \cdots \boldsymbol{\phi}_{2,n_2^2-1}^{2}\right] \left[\boldsymbol{\phi}_{3,0}^{2}\, \boldsymbol{\phi}_{3,1}^{2} \cdots \boldsymbol{\phi}_{3,n_3^2-1}^{2}\right]$$

1. Generate an orthonormal basis for the entire graph ⇒ Laplacian eigenvectors (Notation is $\boldsymbol{\phi}_{k,l}^{j}$ with $j=0$, $|V|=n_0^0$)
2. Partition the graph using the Fiedler vector $\boldsymbol{\phi}_{k,1}^{j}$
3. Generate an orthonormal basis for each of the partitions ⇒ Laplacian eigenvectors
4. Repeat...
5. Select an orthonormal basis from this collection of orthonormal bases

$$\left[\begin{array}{ccccc} \boldsymbol{\phi}_{0,0}^{0} & \boldsymbol{\phi}_{0,1}^{0} & \boldsymbol{\phi}_{0,2}^{0} & \cdots & \boldsymbol{\phi}_{0,n_0^0-1}^{0} \end{array}\right]$$

$$\left[\begin{array}{ccccc} \boldsymbol{\phi}_{0,0}^{1} & \boldsymbol{\phi}_{0,1}^{1} & \boldsymbol{\phi}_{0,2}^{1} & \cdots & \boldsymbol{\phi}_{0,n_0^1-1}^{1} \end{array}\right] \quad \left[\begin{array}{ccccc} \boldsymbol{\phi}_{1,0}^{1} & \boldsymbol{\phi}_{1,1}^{1} & \boldsymbol{\phi}_{1,2}^{1} & \cdots & \boldsymbol{\phi}_{1,n_1^1-1}^{1} \end{array}\right]$$

$$\left[\boldsymbol{\phi}_{0,0}^{2}\, \boldsymbol{\phi}_{0,1}^{2} \cdots \boldsymbol{\phi}_{0,n_0^2-1}^{2}\right] \left[\boldsymbol{\phi}_{1,0}^{2}\, \boldsymbol{\phi}_{1,1}^{2} \cdots \boldsymbol{\phi}_{1,n_1^2-1}^{2}\right] \left[\boldsymbol{\phi}_{2,0}^{2}\, \boldsymbol{\phi}_{2,1}^{2} \cdots \boldsymbol{\phi}_{2,n_2^2-1}^{2}\right] \left[\boldsymbol{\phi}_{3,0}^{2}\, \boldsymbol{\phi}_{3,1}^{2} \cdots \boldsymbol{\phi}_{3,n_3^2-1}^{2}\right]$$

$$\vdots$$

1. Generate an orthonormal basis for the entire graph ⇒ Laplacian eigenvectors (Notation is $\boldsymbol{\phi}_{k,l}^{j}$ with $j=0$, $|V|=n_0^0$)
2. Partition the graph using the Fiedler vector $\boldsymbol{\phi}_{k,1}^{j}$
3. Generate an orthonormal basis for each of the partitions ⇒ Laplacian eigenvectors
4. Repeat...
5. Select an orthonormal basis from this collection of orthonormal bases

$$\left[\begin{array}{ccccc} \boldsymbol{\phi}_{0,0}^{0} & \boldsymbol{\phi}_{0,1}^{0} & \boldsymbol{\phi}_{0,2}^{0} & \cdots & \boldsymbol{\phi}_{0,n_0^0-1}^{0} \end{array}\right]$$

$$\left[\begin{array}{ccccc} \boldsymbol{\phi}_{0,0}^{1} & \boldsymbol{\phi}_{0,1}^{1} & \boldsymbol{\phi}_{0,2}^{1} & \cdots & \boldsymbol{\phi}_{0,n_0^1-1}^{1} \end{array}\right] \quad \left[\begin{array}{ccccc} \boldsymbol{\phi}_{1,0}^{1} & \boldsymbol{\phi}_{1,1}^{1} & \boldsymbol{\phi}_{1,2}^{1} & \cdots & \boldsymbol{\phi}_{1,n_1^1-1}^{1} \end{array}\right]$$

$$\left[\boldsymbol{\phi}_{0,0}^{2}\, \boldsymbol{\phi}_{0,1}^{2} \cdots \boldsymbol{\phi}_{0,n_0^2-1}^{2}\right] \left[\boldsymbol{\phi}_{1,0}^{2}\, \boldsymbol{\phi}_{1,1}^{2} \cdots \boldsymbol{\phi}_{1,n_1^2-1}^{2}\right] \left[\boldsymbol{\phi}_{2,0}^{2}\, \boldsymbol{\phi}_{2,1}^{2} \cdots \boldsymbol{\phi}_{2,n_2^2-1}^{2}\right] \left[\boldsymbol{\phi}_{3,0}^{2}\, \boldsymbol{\phi}_{3,1}^{2} \cdots \boldsymbol{\phi}_{3,n_3^2-1}^{2}\right]$$

$$\vdots$$

Remarks

- For an unweighted path graph, this *exactly* yields the *Hierarchical Block DCT Dictionary*
- 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 ⇒ the best-basis algorithm, the local discriminant basis algorithm, ...
  - A union of bases on disjoint subsets is obviously orthonormal
  - There are about $O\left(1.5^{n_0^0}\right)$ searchable ONBs in this basis dictionary

Remarks

- For an unweighted path graph, this *exactly* yields the *Hierarchical Block DCT Dictionary*
- 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 ⇒ the best-basis algorithm, the local discriminant basis algorithm, ...
  - A union of bases on disjoint subsets is obviously orthonormal
  - There are about $O\left(1.5^{n_0^0}\right)$ searchable ONBs in this basis dictionary

Remarks

- For an unweighted path graph, this *exactly* yields the *Hierarchical Block DCT Dictionary*
- 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 ⇒ the best-basis algorithm, the local discriminant basis algorithm, ...
  - A union of bases on disjoint subsets is obviously orthonormal
  - There are about $O\left(1.5^{n_0^0}\right)$ searchable ONBs in this basis dictionary

Remarks

- For an unweighted path graph, this *exactly* yields the *Hierarchical Block DCT Dictionary*
- 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 ⇒ the best-basis algorithm, the local discriminant basis algorithm, ...
  - A union of bases on disjoint subsets is obviously orthonormal
  - There are about $O\left(1.5^{n_0^0}\right)$ searchable ONBs in this basis dictionary

$$\left[\begin{array}{cccc} \boldsymbol{\phi}_{0,0}^0 & \boldsymbol{\phi}_{0,1}^0 & \boldsymbol{\phi}_{0,2}^0 & \cdots & \boldsymbol{\phi}_{0,n_0^0-1}^0 \end{array}\right]$$

$$\left[\begin{array}{cccccc} \boldsymbol{\phi}_{0,0}^1 & \boldsymbol{\phi}_{0,1}^1 & \boldsymbol{\phi}_{0,2}^1 & \cdots & \boldsymbol{\phi}_{0,n_0^1-1}^1 \end{array}\right] \left[\begin{array}{cccccc} \boldsymbol{\phi}_{1,0}^1 & \boldsymbol{\phi}_{1,1}^1 & \boldsymbol{\phi}_{1,2}^1 & \cdots & \boldsymbol{\phi}_{1,n_1^1-1}^1 \end{array}\right]$$

$$\left[\begin{array}{cccc} \boldsymbol{\phi}_{0,0}^2 & \cdots & \boldsymbol{\phi}_{0,n_0^2-1}^2 \end{array}\right] \left[\begin{array}{cccc} \boldsymbol{\phi}_{1,0}^2 & \cdots & \boldsymbol{\phi}_{1,n_1^2-1}^2 \end{array}\right] \left[\begin{array}{cccc} \boldsymbol{\phi}_{2,0}^2 & \cdots & \boldsymbol{\phi}_{2,n_2^2-1}^2 \end{array}\right] \left[\begin{array}{cccc} \boldsymbol{\phi}_{3,0}^2 & \cdots & \boldsymbol{\phi}_{3,n_3^2-1}^2 \end{array}\right]$$

Remarks

- For an unweighted path graph, this *exactly* yields the *Hierarchical Block DCT Dictionary*
- 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 ⇒ the best-basis algorithm, the local discriminant basis algorithm, ...
  - A union of bases on disjoint subsets is obviously orthonormal
  - There are about $O\left(1.5^{n_0^0}\right)$ searchable ONBs in this basis dictionary

$$\left[\begin{array}{cccc} \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,n_0^0-1} \end{array}\right]$$

$$\left[\boldsymbol{\phi}^1_{0,0} \quad \boldsymbol{\phi}^1_{0,1} \quad \boldsymbol{\phi}^1_{0,2} \quad \cdots \quad \boldsymbol{\phi}^1_{0,n_0^1-1}\right]\left[\boldsymbol{\phi}^1_{1,0} \quad \boldsymbol{\phi}^1_{1,1} \quad \boldsymbol{\phi}^1_{1,2} \quad \cdots \quad \boldsymbol{\phi}^1_{1,n_1^1-1}\right]$$

$$\left[\boldsymbol{\phi}^2_{0,0} \quad \cdots \quad \boldsymbol{\phi}^2_{0,n_0^2-1}\right]\left[\boldsymbol{\phi}^2_{1,0} \quad \cdots \quad \boldsymbol{\phi}^2_{1,n_1^2-1}\right]\left[\boldsymbol{\phi}^2_{2,0} \quad \cdots \quad \boldsymbol{\phi}^2_{2,n_2^2-1}\right]\left[\boldsymbol{\phi}^2_{3,0} \quad \cdots \quad \boldsymbol{\phi}^2_{3,n_3^2-1}\right]$$

Remarks

- For an unweighted path graph, this *exactly* yields the *Hierarchical Block DCT Dictionary*
- 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 ⇒ the best-basis algorithm, the local discriminant basis algorithm, ...
  - A union of bases on disjoint subsets is obviously orthonormal
  - There are about $O\left(1.5^{n_0^0}\right)$ searchable ONBs in this basis dictionary

$$\left[\begin{array}{cccc} \boldsymbol{\phi}_{0,0}^0 & \boldsymbol{\phi}_{0,1}^0 & \boldsymbol{\phi}_{0,2}^0 & \cdots & \boldsymbol{\phi}_{0,n_0^0-1}^0 \end{array}\right]$$

$$\left[\boldsymbol{\phi}_{0,0}^1 \quad \boldsymbol{\phi}_{0,1}^1 \quad \boldsymbol{\phi}_{0,2}^1 \quad \cdots \quad \boldsymbol{\phi}_{0,n_0^1-1}^1\right]\left[\boldsymbol{\phi}_{1,0}^1 \quad \boldsymbol{\phi}_{1,1}^1 \quad \boldsymbol{\phi}_{1,2}^1 \quad \cdots \quad \boldsymbol{\phi}_{1,n_1^1-1}^1\right]$$

$$\left[\boldsymbol{\phi}_{0,0}^2 \quad \cdots \quad \boldsymbol{\phi}_{0,n_0^2-1}^2\right]\left[\boldsymbol{\phi}_{1,0}^2 \quad \cdots \quad \boldsymbol{\phi}_{1,n_1^2-1}^2\right]\left[\boldsymbol{\phi}_{2,0}^2 \quad \cdots \quad \boldsymbol{\phi}_{2,n_2^2-1}^2\right]\left[\boldsymbol{\phi}_{3,0}^2 \quad \cdots \quad \boldsymbol{\phi}_{3,n_3^2-1}^2\right]$$

Remarks

- For an unweighted path graph, this *exactly* yields the *Hierarchical Block DCT Dictionary*
- 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 ⇒ the best-basis algorithm, the local discriminant basis algorithm, ...
  - A union of bases on disjoint subsets is obviously orthonormal
  - There are about $O\left(1.5^{n_0^0}\right)$ searchable ONBs in this basis dictionary

$$\left[\begin{array}{cccc} \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,n_0^0-1} \end{array}\right]$$

$$\left[\boldsymbol{\phi}^1_{0,0} \quad \boldsymbol{\phi}^1_{0,1} \quad \boldsymbol{\phi}^1_{0,2} \quad \cdots \quad \boldsymbol{\phi}^1_{0,n_0^1-1}\right]\left[\boldsymbol{\phi}^1_{1,0} \quad \boldsymbol{\phi}^1_{1,1} \quad \boldsymbol{\phi}^1_{1,2} \quad \cdots \quad \boldsymbol{\phi}^1_{1,n_1^1-1}\right]$$

$$\left[\boldsymbol{\phi}^2_{0,0} \quad \cdots \quad \boldsymbol{\phi}^2_{0,n_0^2-1}\right]\left[\boldsymbol{\phi}^2_{1,0} \quad \cdots \quad \boldsymbol{\phi}^2_{1,n_1^2-1}\right]\left[\boldsymbol{\phi}^2_{2,0} \quad \cdots \quad \boldsymbol{\phi}^2_{2,n_2^2-1}\right]\left[\boldsymbol{\phi}^2_{3,0} \quad \cdots \quad \boldsymbol{\phi}^2_{3,n_3^2-1}\right]$$

Remarks

- For an unweighted path graph, this *exactly* yields the *Hierarchical Block DCT Dictionary*
- 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 ⇒ the best-basis algorithm, the local discriminant basis algorithm, ...
  - A union of bases on disjoint subsets is obviously orthonormal
  - There are about $O\left(1.5^{n_0^0}\right)$ searchable ONBs in this basis dictionary

$$\left[\begin{array}{cccc} \boldsymbol{\phi}_{0,0}^0 & \boldsymbol{\phi}_{0,1}^0 & \boldsymbol{\phi}_{0,2}^0 & \cdots & \boldsymbol{\phi}_{0,n_0^0-1}^0 \end{array}\right]$$

$$\left[\begin{array}{cccc} \boldsymbol{\phi}_{0,0}^1 & \boldsymbol{\phi}_{0,1}^1 & \boldsymbol{\phi}_{0,2}^1 & \cdots & \boldsymbol{\phi}_{0,n_0^1-1}^1 \end{array}\right] \left[\begin{array}{cccc} \boldsymbol{\phi}_{1,0}^1 & \boldsymbol{\phi}_{1,1}^1 & \boldsymbol{\phi}_{1,2}^1 & \cdots & \boldsymbol{\phi}_{1,n_1^1-1}^1 \end{array}\right]$$

$$\left[\boldsymbol{\phi}_{0,0}^2 \quad \cdots \quad \boldsymbol{\phi}_{0,n_0^2-1}^2\right] \left[\boldsymbol{\phi}_{1,0}^2 \quad \cdots \quad \boldsymbol{\phi}_{1,n_1^2-1}^2\right] \left[\boldsymbol{\phi}_{2,0}^2 \quad \cdots \quad \boldsymbol{\phi}_{2,n_2^2-1}^2\right] \left[\boldsymbol{\phi}_{3,0}^2 \quad \cdots \quad \boldsymbol{\phi}_{3,n_3^2-1}^2\right]$$

Remarks

- For an unweighted path graph, this *exactly* yields the *Hierarchical Block DCT Dictionary*
- 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 ⇒ the best-basis algorithm, the local discriminant basis algorithm, ...
  - A union of bases on disjoint subsets is obviously orthonormal
  - There are about $O\left(1.5^{n_0^0}\right)$ searchable ONBs in this basis dictionary

$$\left[\begin{array}{cccc} \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,n_0^0-1} \end{array}\right]$$

$$\left[\boldsymbol{\phi}^1_{0,0} \quad \boldsymbol{\phi}^1_{0,1} \quad \boldsymbol{\phi}^1_{0,2} \quad \cdots \quad \boldsymbol{\phi}^1_{0,n_0^1-1}\right]\left[\boldsymbol{\phi}^1_{1,0} \quad \boldsymbol{\phi}^1_{1,1} \quad \boldsymbol{\phi}^1_{1,2} \quad \cdots \quad \boldsymbol{\phi}^1_{1,n_1^1-1}\right]$$

$$\left[\boldsymbol{\phi}^2_{0,0} \quad \cdots \quad \boldsymbol{\phi}^2_{0,n_0^2-1}\right]\left[\boldsymbol{\phi}^2_{1,0} \quad \cdots \quad \boldsymbol{\phi}^2_{1,n_1^2-1}\right]\left[\boldsymbol{\phi}^2_{2,0} \quad \cdots \quad \boldsymbol{\phi}^2_{2,n_2^2-1}\right]\left[\boldsymbol{\phi}^2_{3,0} \quad \cdots \quad \boldsymbol{\phi}^2_{3,n_3^2-1}\right]$$

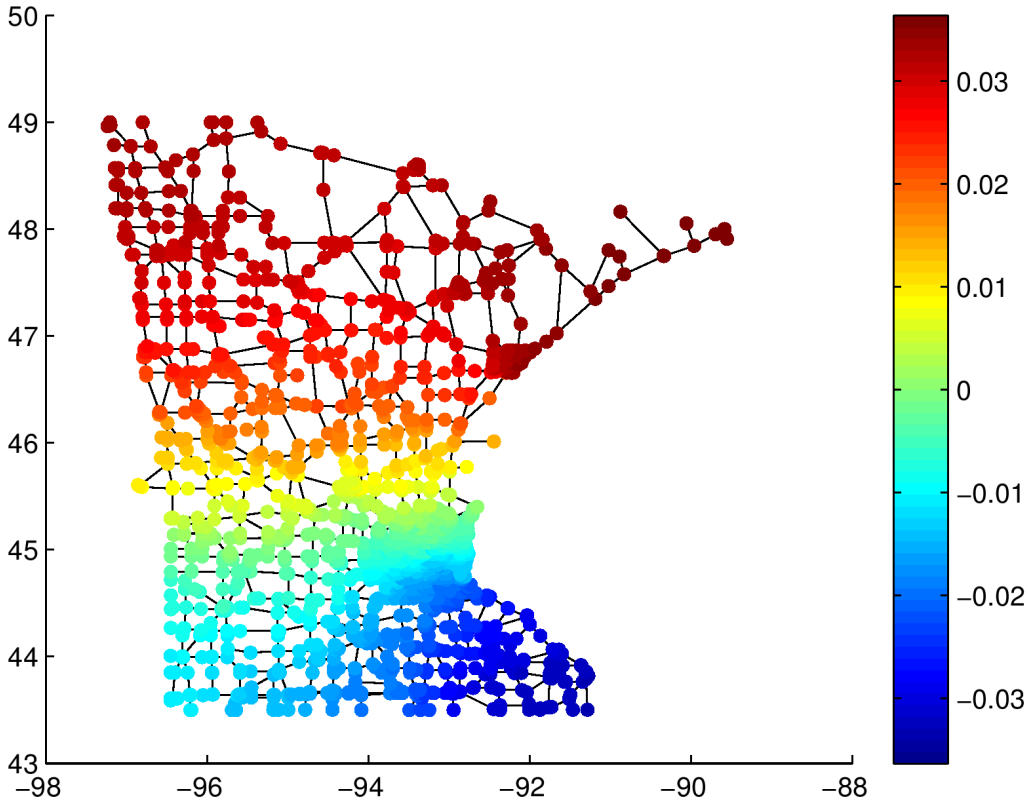
HGLET Basis Vectors on MN

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

HGLET Basis Vectors on MN

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

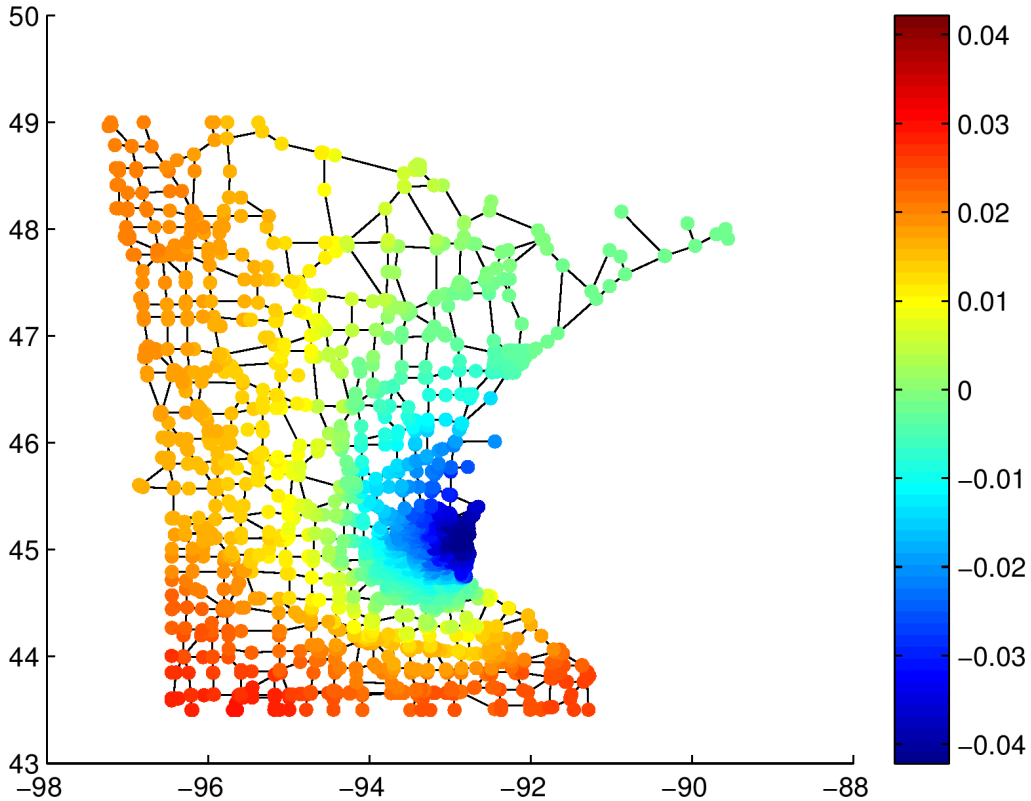
Level $j = 0$, Region $k = 0$, $\boldsymbol{\phi}_{0,1}^{0}$



HGLET Basis Vectors on MN

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

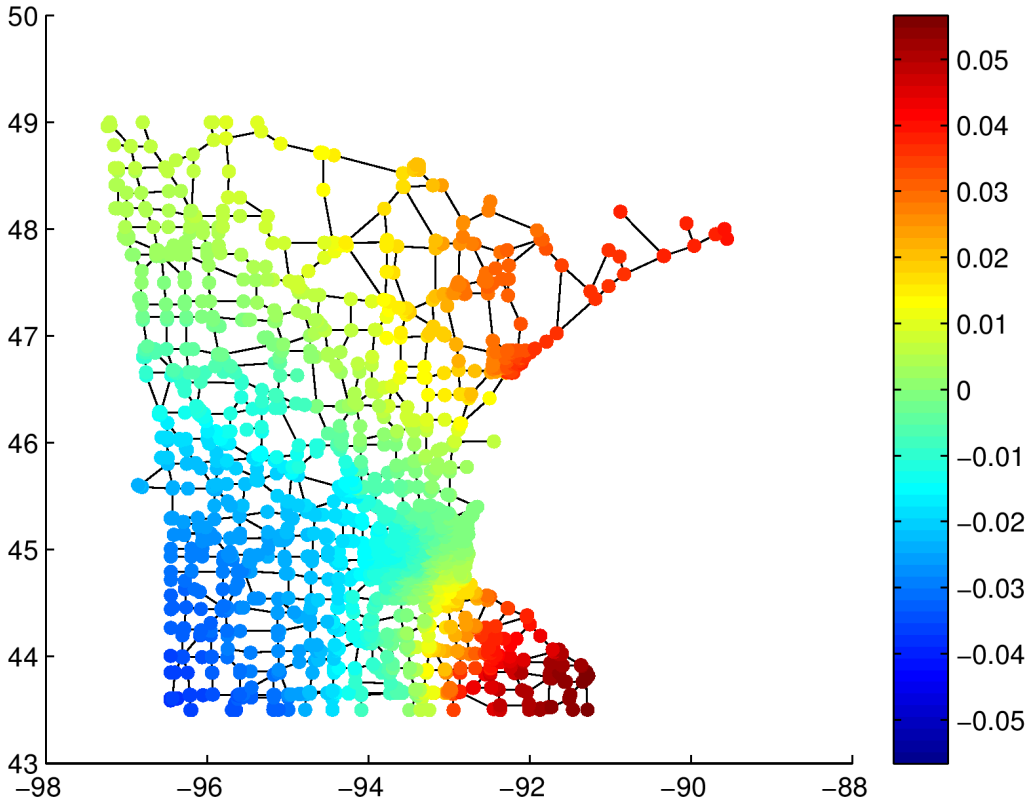
Level $j = 0$, Region $k = 0$, $\boldsymbol{\phi}_{0,2}^{0}$



HGLET Basis Vectors on MN

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

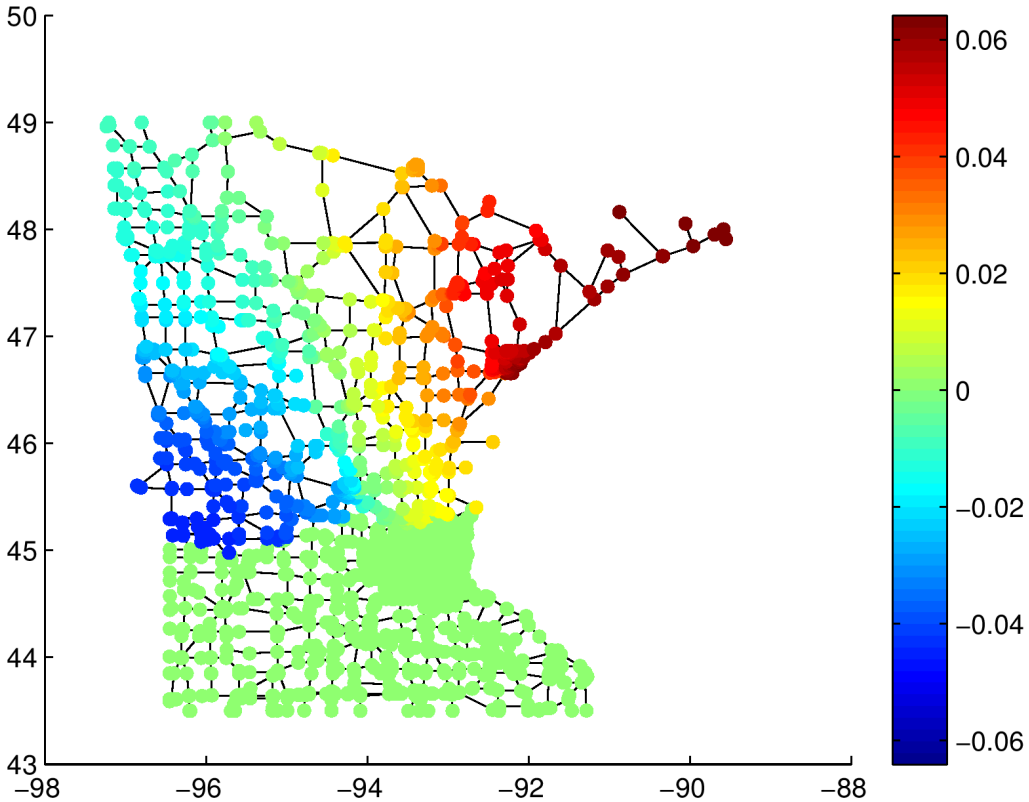
Level $j = 0$, Region $k = 0$, $\boldsymbol{\phi}_{0,3}^{0}$



HGLET Basis Vectors on MN

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

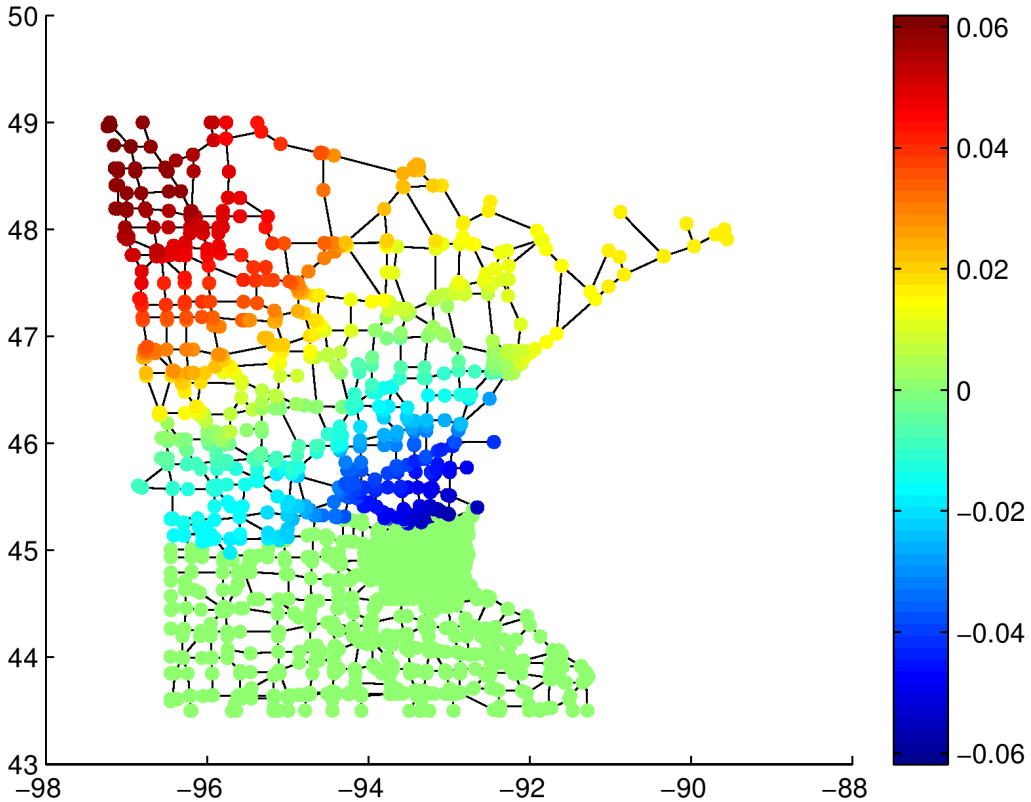
Level $j = 1$, Region $k = 0$, $\boldsymbol{\phi}_{0,1}^{1}$



HGLET Basis Vectors on MN

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

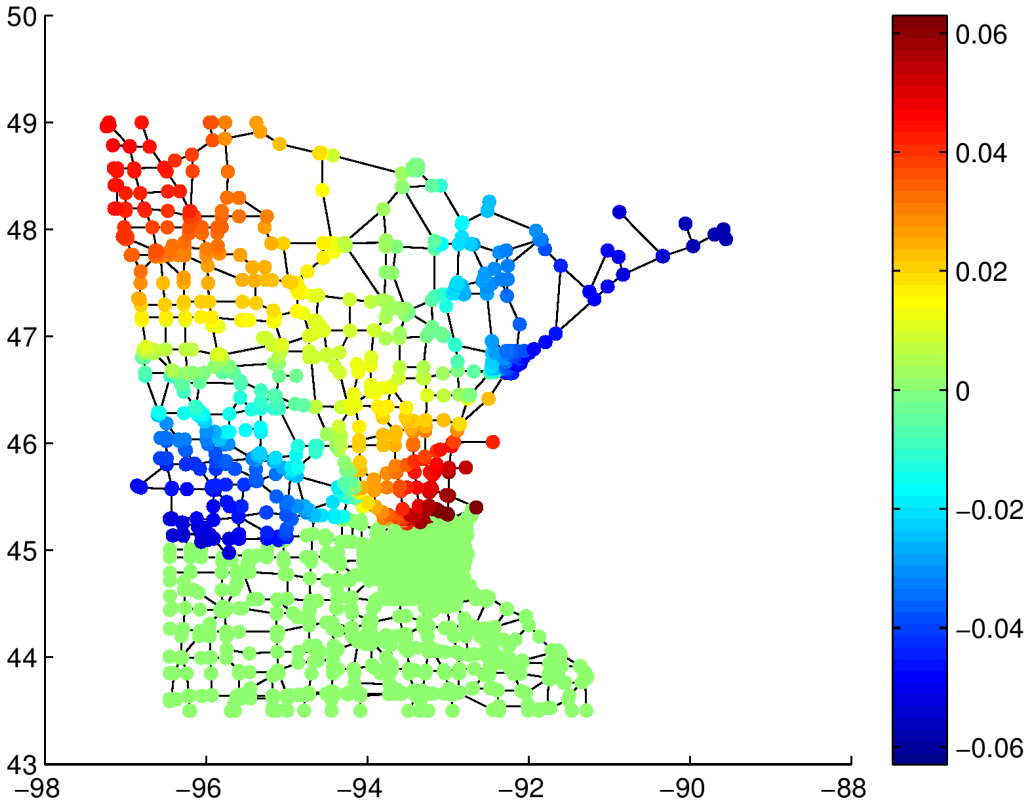
Level $j = 1$, Region $k = 0$, $\boldsymbol{\phi}_{0,2}^{1}$



HGLET Basis Vectors on MN

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

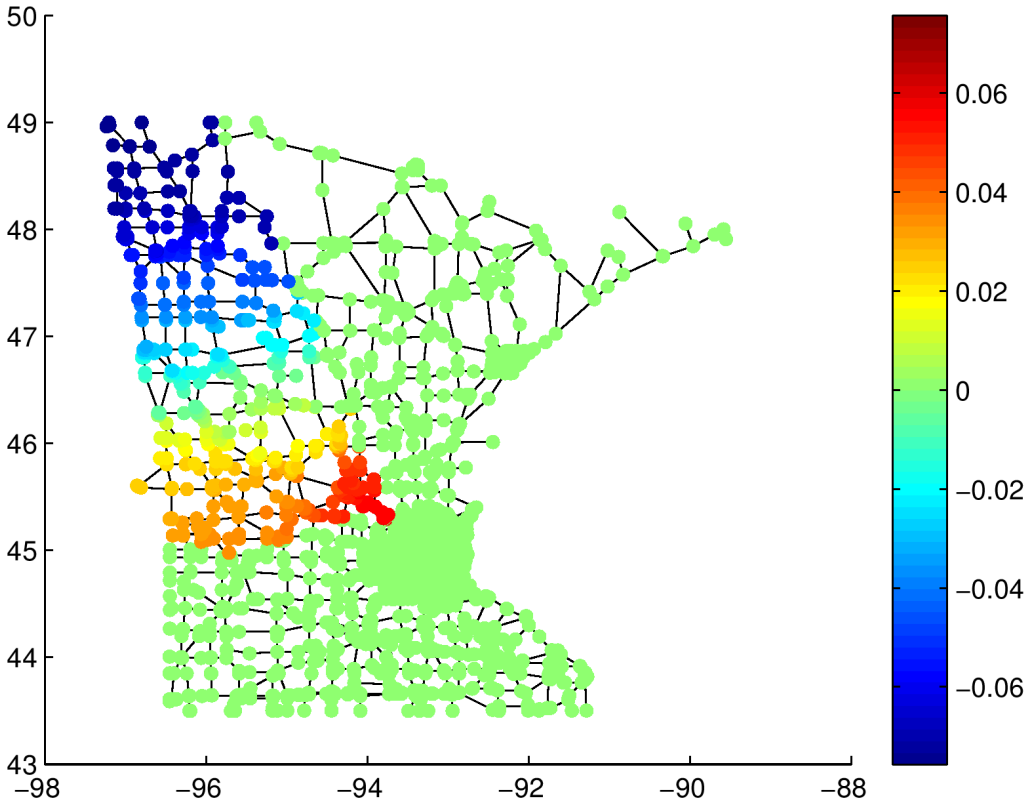
Level $j = 1$, Region $k = 0$, $\boldsymbol{\phi}_{0,3}^{1}$



HGLET Basis Vectors on MN

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

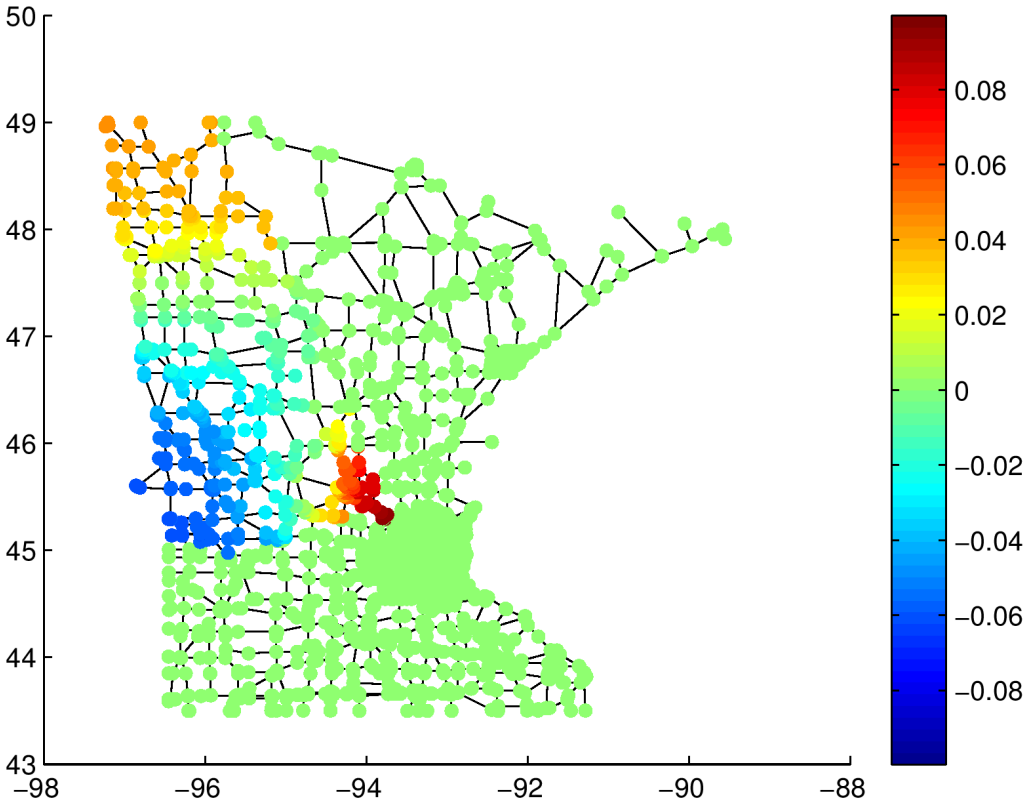
Level $j = 2$, Region $k = 0$, $\boldsymbol{\phi}_{0,1}^{2}$



HGLET Basis Vectors on MN

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

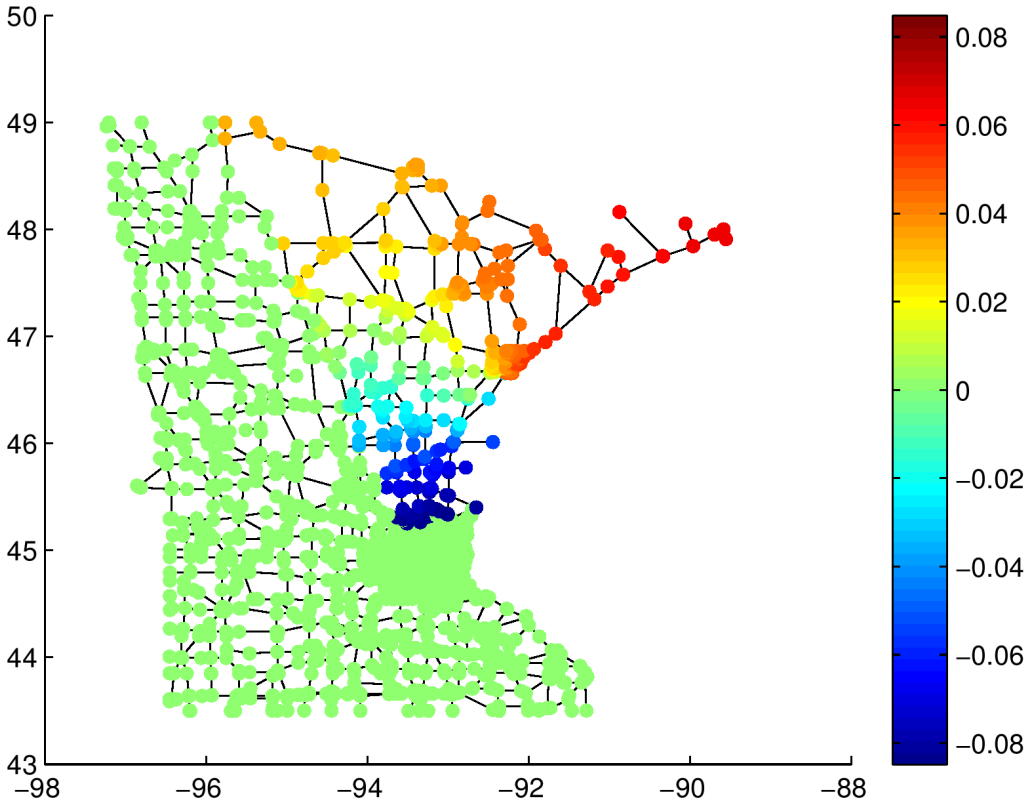
Level $j = 2$, Region $k = 0$, $\boldsymbol{\phi}_{0,2}^2$



HGLET Basis Vectors on MN

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

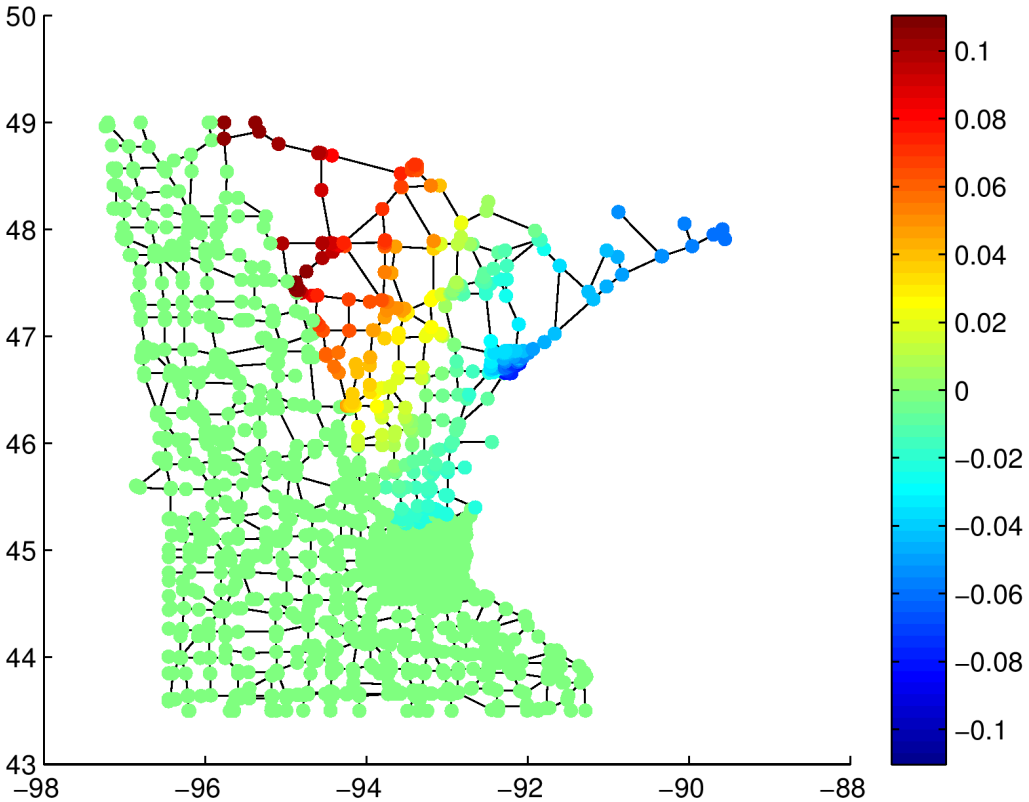
Level $j = 2$, Region $k = 1$, $\boldsymbol{\phi}_{1,1}^{2}$



HGLET Basis Vectors on MN

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

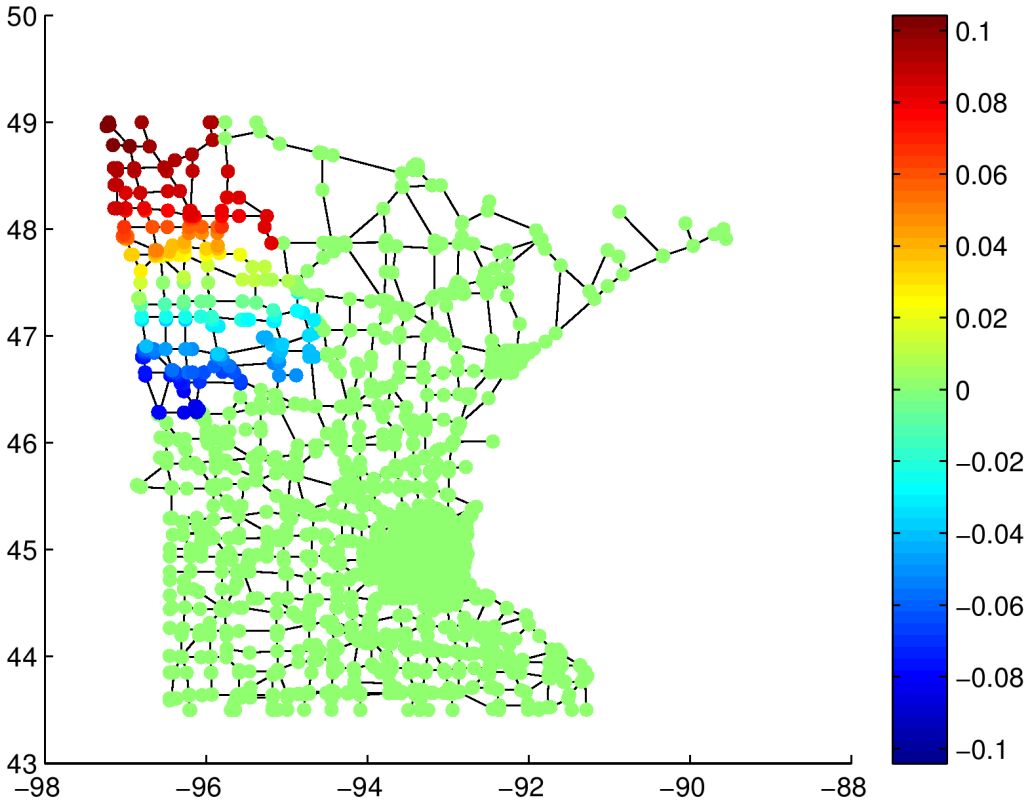
Level $j = 2$, Region $k = 1$, $\boldsymbol{\phi}_{1,2}^2$



HGLET Basis Vectors on MN

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

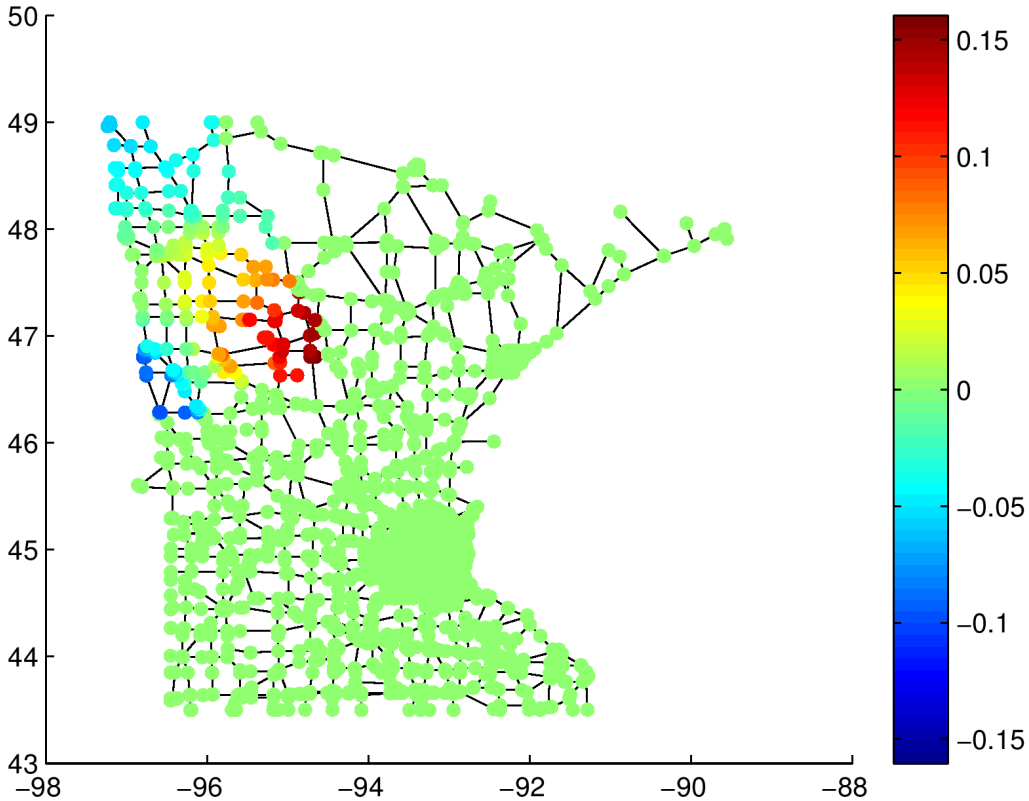
Level $j = 3$, Region $k = 0$, $\boldsymbol{\phi}_{0,1}^{3}$



HGLET Basis Vectors on MN

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

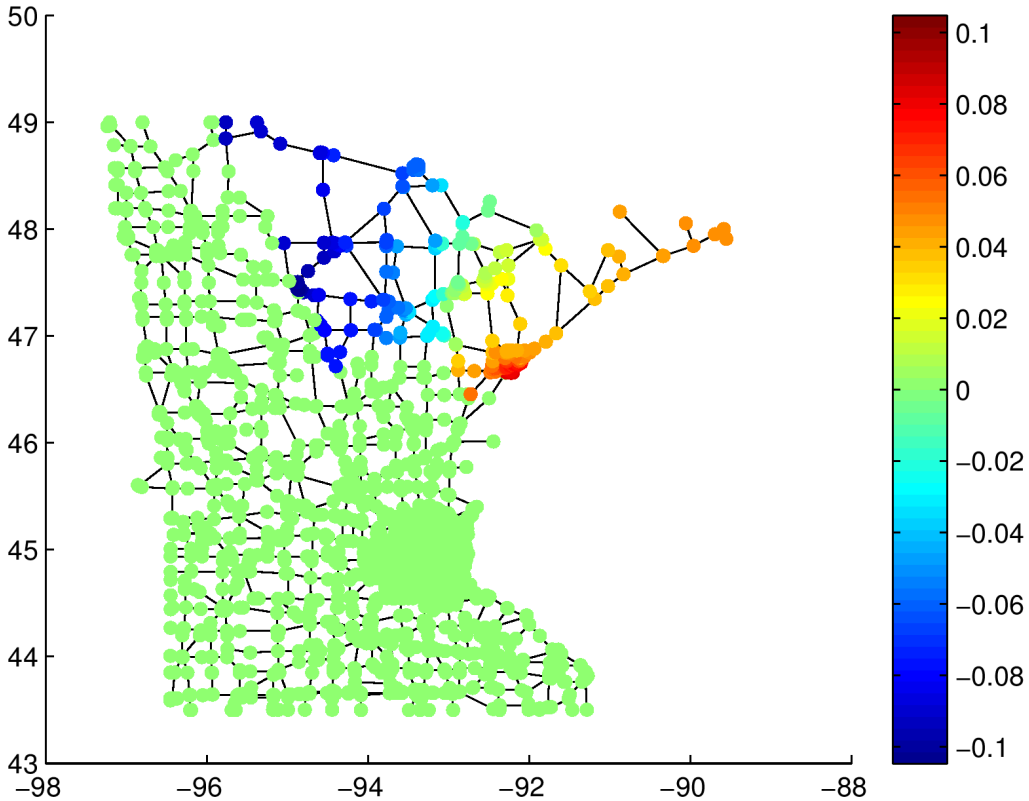
Level $j = 3$, Region $k = 0$, $\boldsymbol{\phi}_{0,2}^{3}$



HGLET Basis Vectors on MN

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

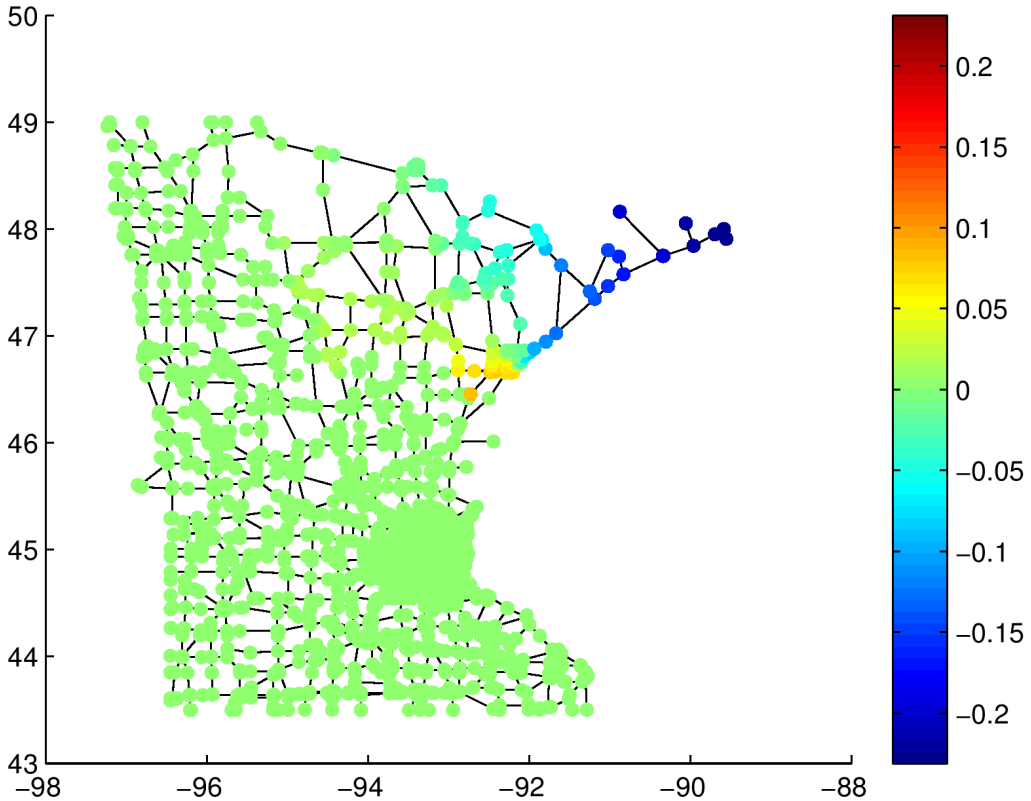
Level $j = 3$, Region $k = 1$, $\boldsymbol{\phi}_{1,1}^{3}$



HGLET Basis Vectors on MN

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

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



Related Work

- The following work also proposed a similar strategy to construct a multiscale basis dictionary, i.e., *local cosine dictionary on a graph*:
  - A. D. Szlam, M. Maggioni, R. R. Coifman, and J. C. Bremer, Jr., "Diffusion-driven multiscale analysis on manifolds and graphs: top-down and bottom-up constructions," in *Wavelets XI* (M. Papadakis et al. eds.), *Proc. SPIE 5914*, Paper # 59141D, 2005.

Related Work

- The following work also proposed a similar strategy to construct a multiscale basis dictionary, i.e., *local cosine dictionary on a graph*:
  - A. D. Szlam, M. Maggioni, R. R. Coifman, and J. C. Bremer, Jr., "Diffusion-driven multiscale analysis on manifolds and graphs: top-down and bottom-up constructions," in *Wavelets XI* (M. Papadakis et al. eds.), *Proc. SPIE 5914*, Paper # 59141D, 2005.
- However, in our opinion, the generalization of the folding/unfolding operations (originally used in the construction of the local cosine transforms on a regular domain) to the graph setting must be done cautiously. If one needs smoother and overlapping basis vectors, then a better partitioning scheme other than the folding/unfolding operations is called for.

Computational Complexity: HGLET

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

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

Outline

The Best-Basis Algorithm

- 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 bases that is "best" for approximation and compression.
- We require an appropriate cost functional $\mathscr{J}$. For example:

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

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

The Best-Basis Algorithm

- 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 bases that is "best" for approximation and compression.
- We require an appropriate cost functional $\mathscr{J}$. For example:

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

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

The Best-Basis Algorithm

- 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 bases that is "best" for approximation and compression.
- We require an appropriate cost functional $\mathscr{J}$. For example:

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

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

The Best-Basis Algorithm

- 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 bases that is "best" for approximation and compression.
- We require an appropriate cost functional $\mathscr{J}$. For example:

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

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

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

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

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

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

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

$$c^2_{0,0} \; c^2_{0,1} \cdots c^2_{0,n^2_0-1} \qquad c^2_{1,0} \; c^2_{1,1} \cdots c^2_{1,n^2_1-1} \qquad c^2_{2,0} \; c^2_{2,1} \cdots c^2_{2,n^2_2-1} \qquad c^2_{3,0} \; c^2_{3,1} \cdots c^2_{3,n^2_3-1}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

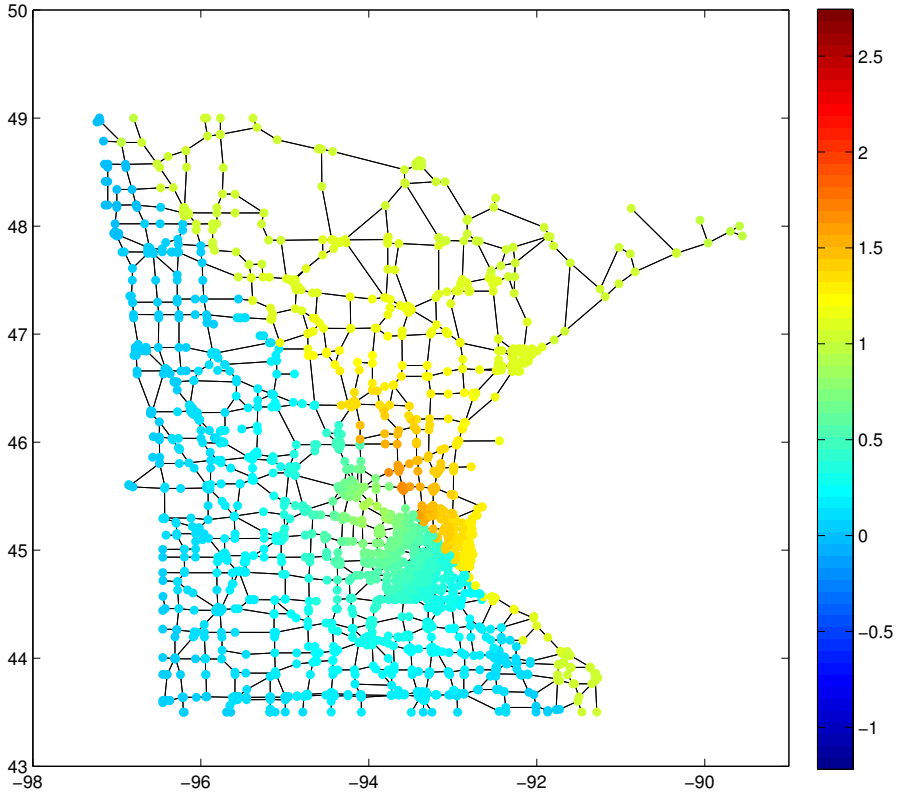
$$\mathscr{J}\left(\boldsymbol{c}^0_0\right) > \mathscr{J}\left(\boldsymbol{c}^1_0; \boldsymbol{c}^2_2; \boldsymbol{c}^2_3\right)$$

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

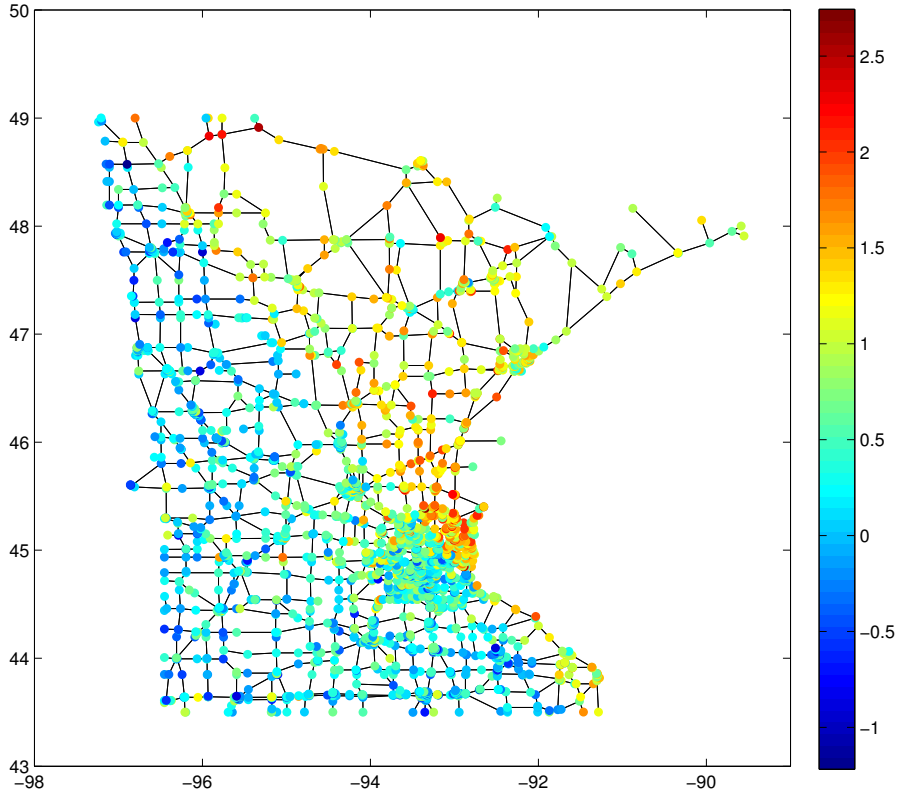
Outline

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 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:

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

   - Note: keep all scaling coefficients intact

Denoising Algorithm

1. Construct the HGLET 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:

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

   - Note: keep all scaling coefficients intact

Denoising Algorithm

1. Construct the HGLET 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:

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

   - Note: keep all scaling coefficients intact

Denoising Algorithm

1. Construct the HGLET 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:

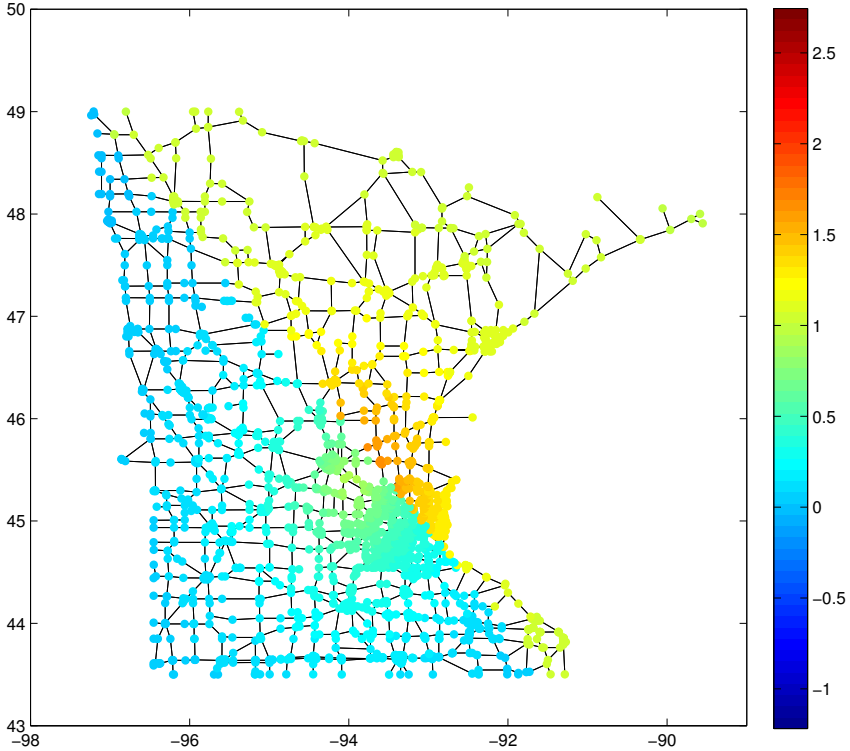
$$c_{\mathrm{ST}}(i) = \begin{cases} \mathrm{sign}(c(i)) \cdot (|c(i)| - T) & \text{if } |c(i)| > T \\ 0 & \text{otherwise} \end{cases}$$

   - Note: keep all scaling coefficients intact

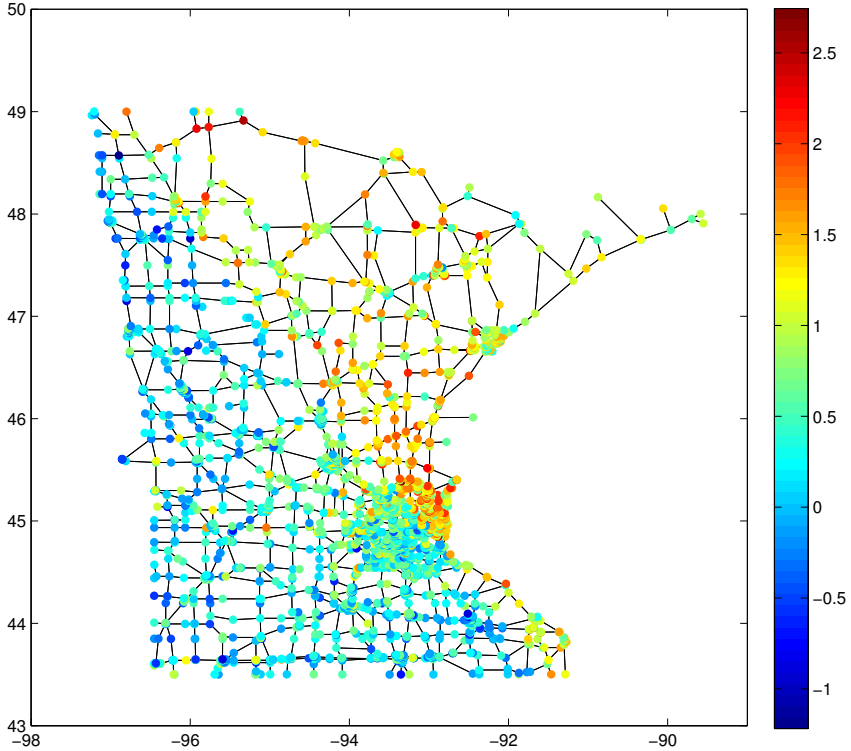
Preliminary Results (L_{rw}'s for Recursive Partitioning)

Transform	SNR (dB)	Coefficients Kept (%)
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) ...

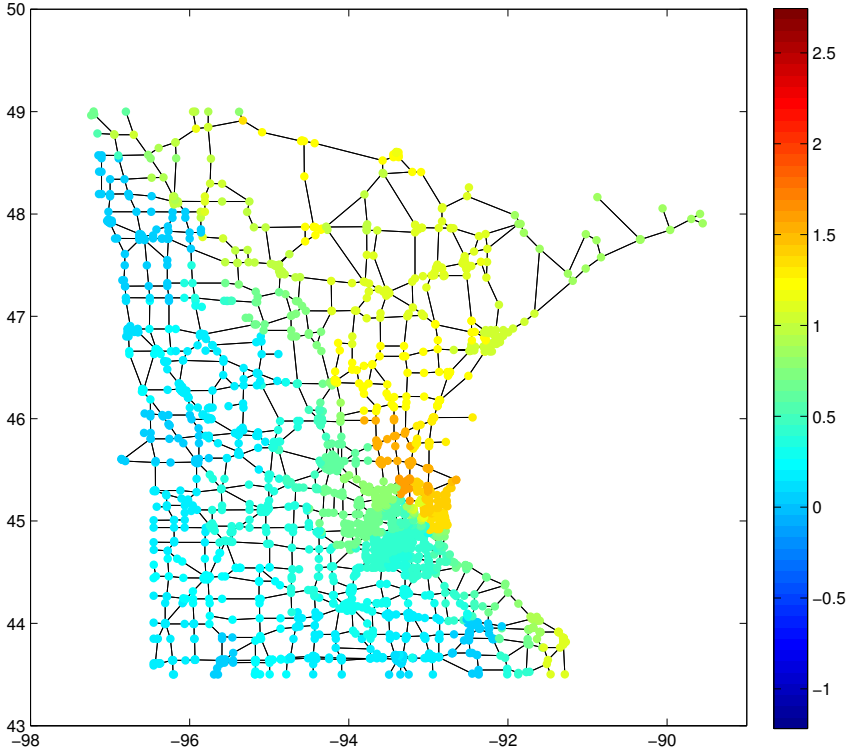


(a) Original: SNR = ∞

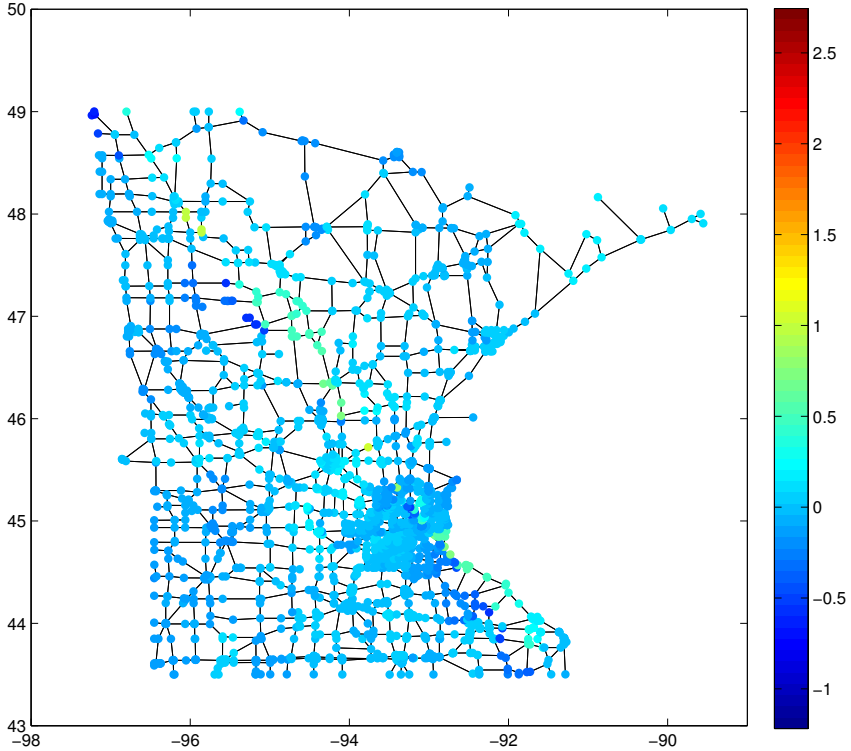


(b) Noisy signal: SNR = 5 dB

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

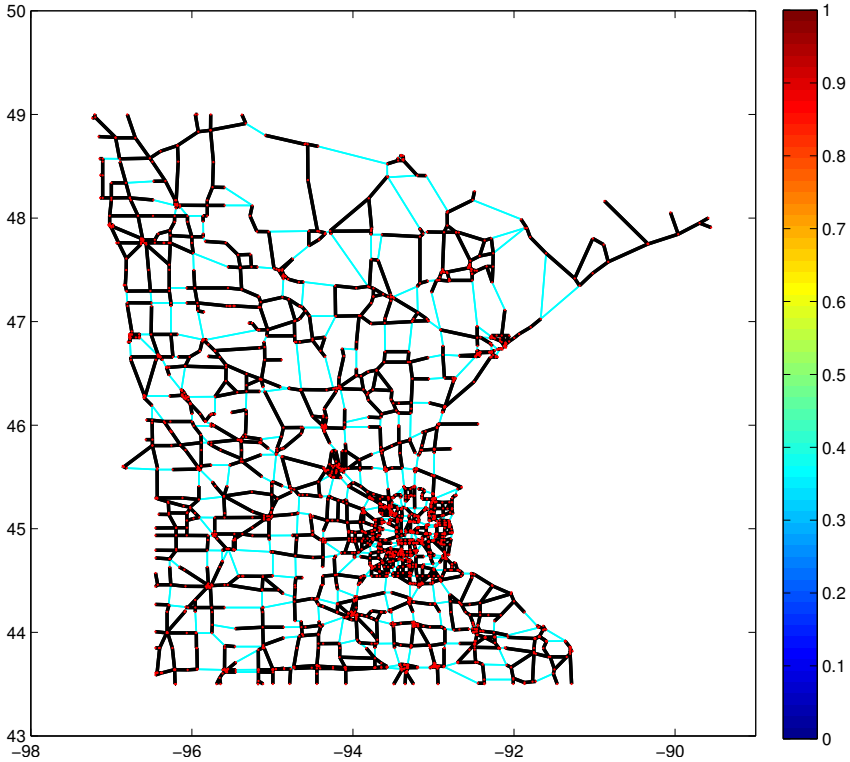


(a) HGLET $j = 6$ (L): SNR = 14.01 dB

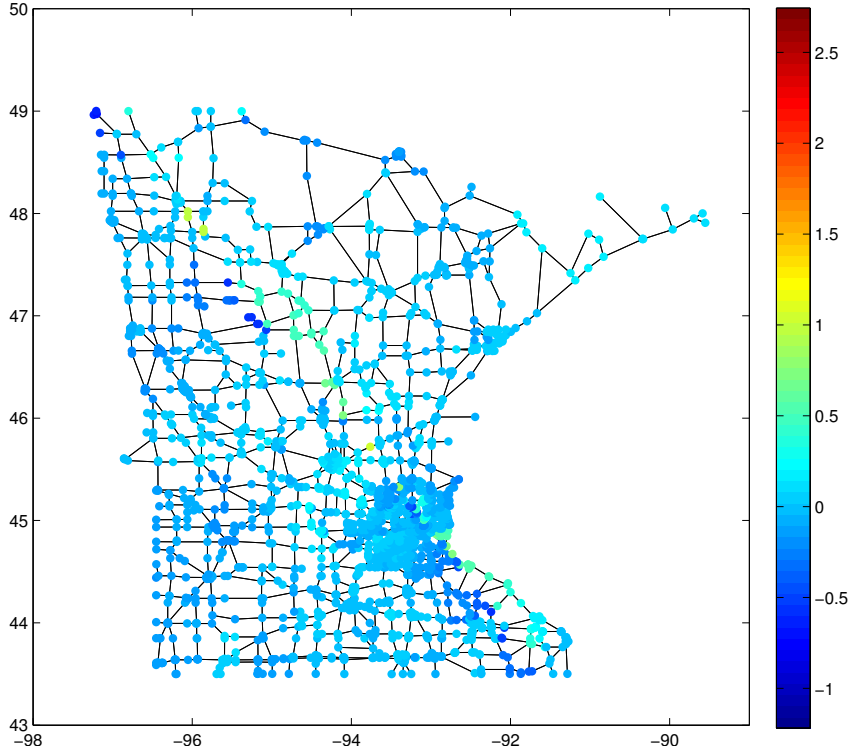


(b) Residual of HGLET $j = 6$ (L): SNR = 14.01 dB

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

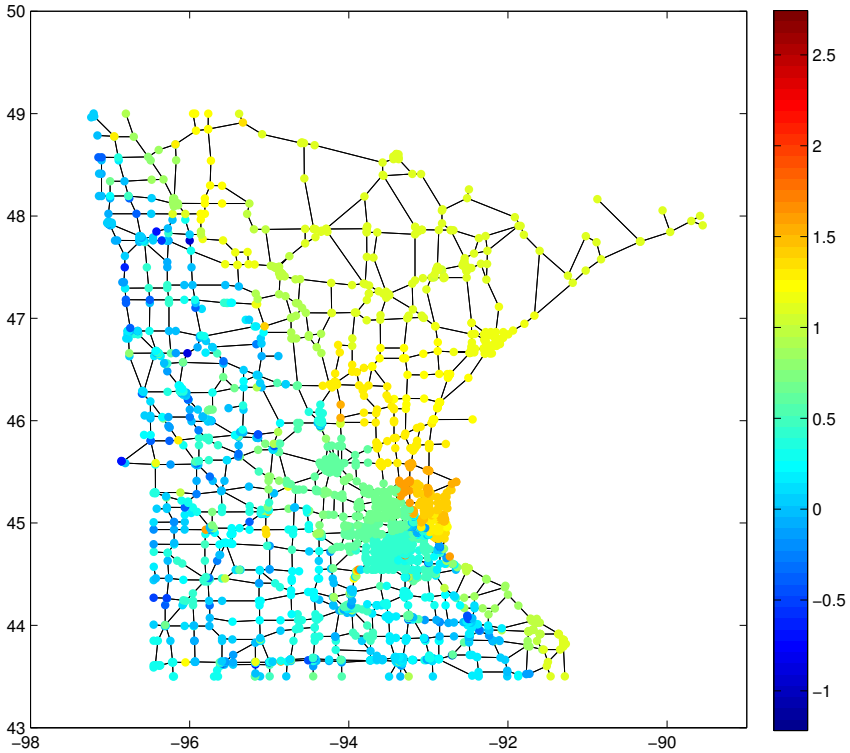


(a) Partition at $j = 6$

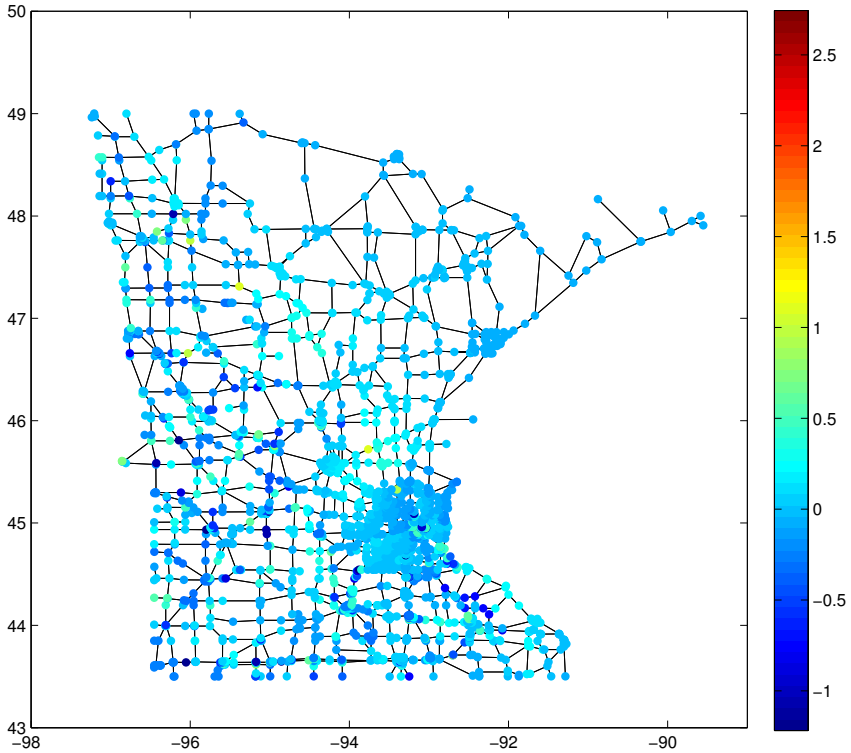


(b) Residual of HGLET $j = 6$ (L): SNR = 14.01 dB

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

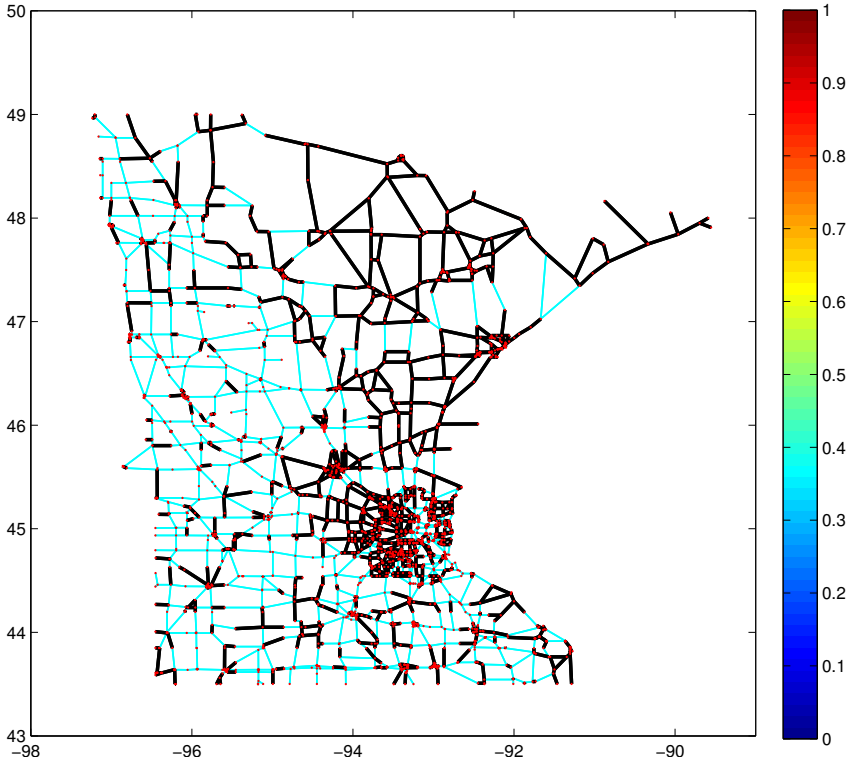


(a) HGLET BB (L): SNR = 10.15 dB

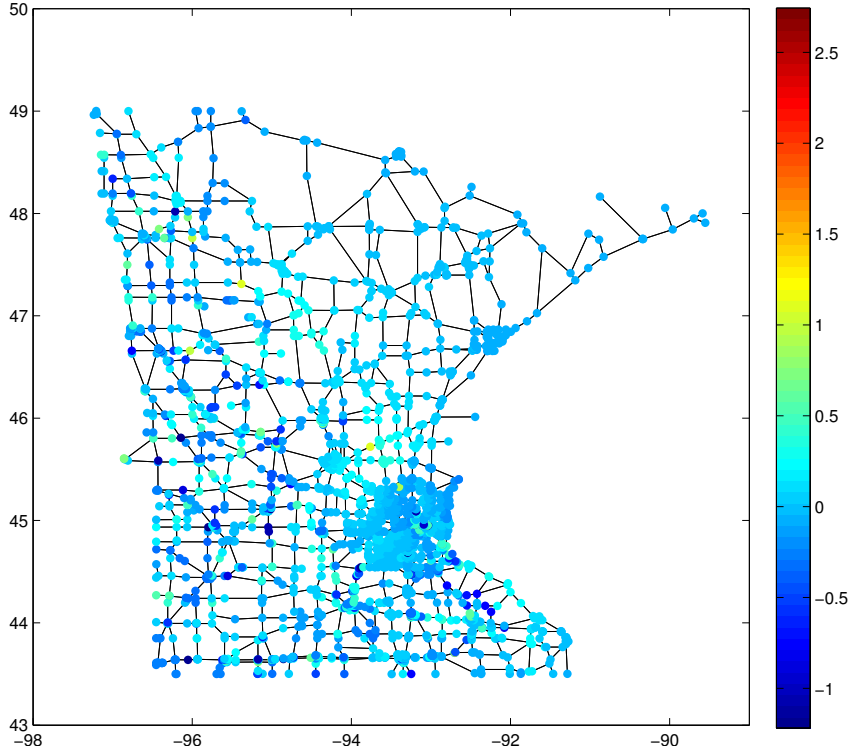


(b) Residual of HGLET BB (L): SNR = 10.15 dB

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

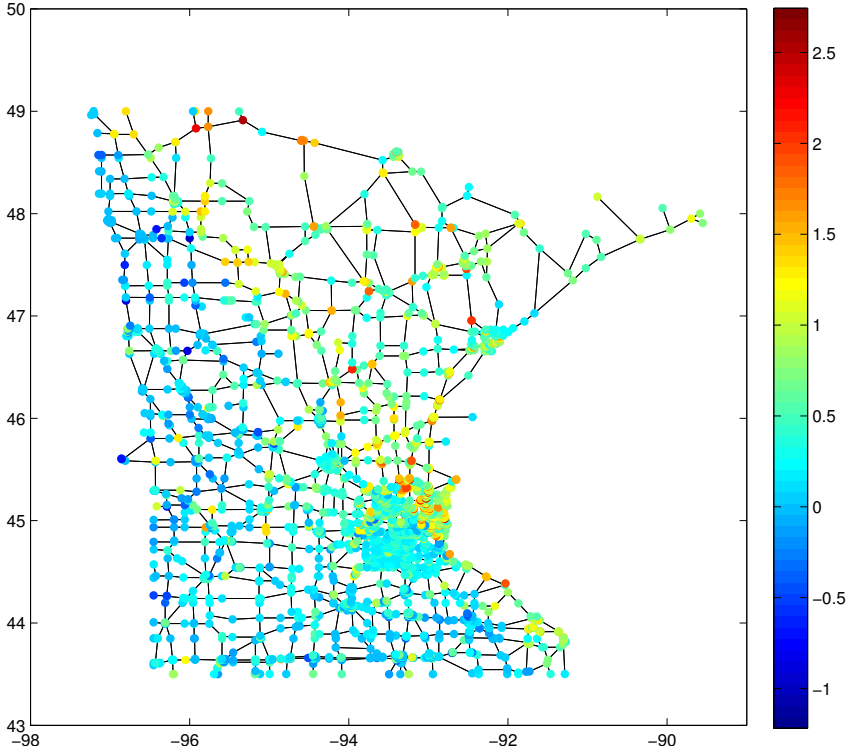


(a) HGLET BB (L) Partition

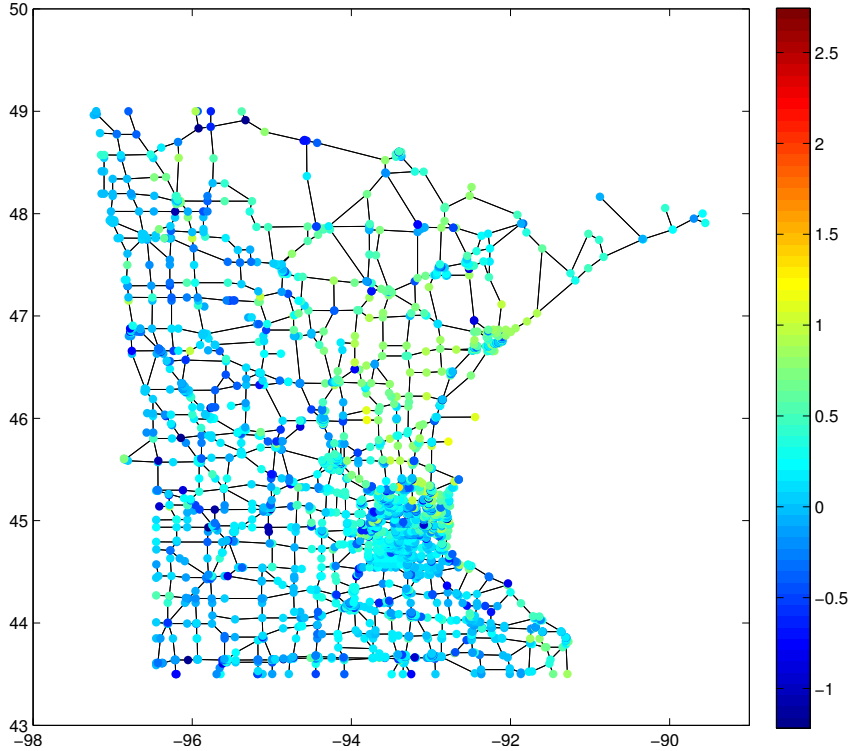


(b) Residual of HGLET BB (L): SNR = 10.15 dB

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

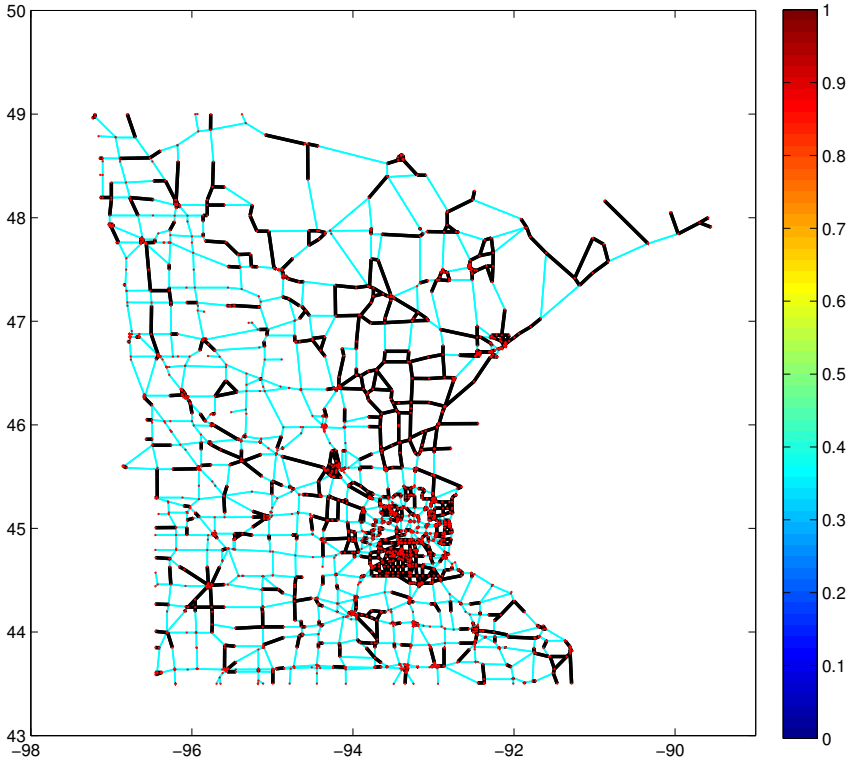


(a) HGLET BB (L_{sym}): SNR = 5.65 dB

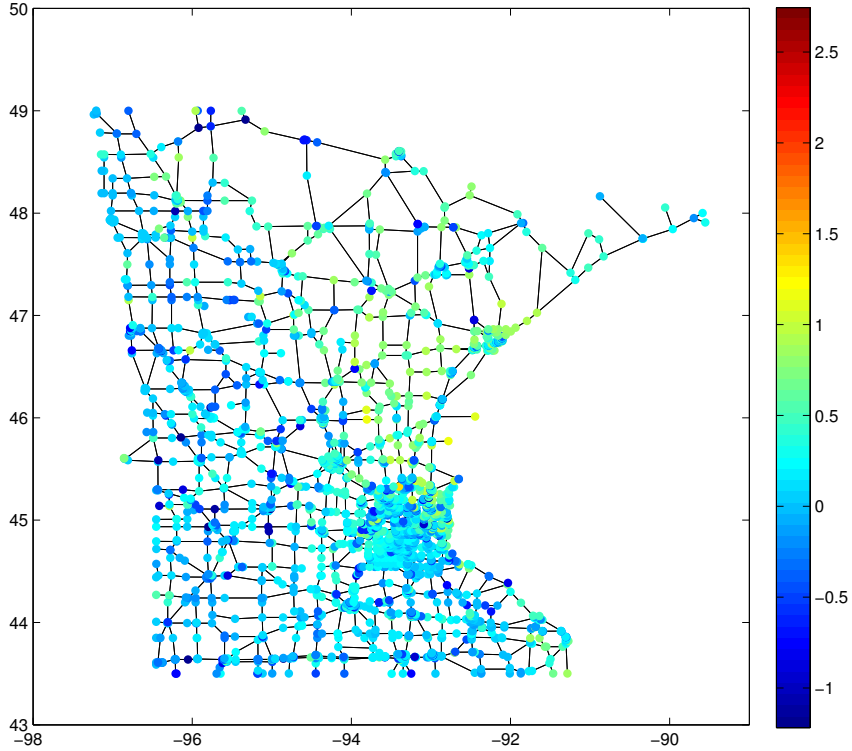


(b) Residual of HGLET BB (L_{sym}): SNR = 5.65 dB

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

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.
- We also truncated this graph to have $2^{11} = 2048$ vertices, and denoised the data using the standard transforms such as the Haar, Symmlet, and BDCT *by ignoring the graph structure and viewing them as samples on the 1D regular lattice*. The resulting SNR values range from 5.01 dB to 6.98 dB, i.e., much worse than the graph-based denoising methods in general.

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.
- We also truncated this graph to have $2^{11} = 2048$ vertices, and denoised the data using the standard transforms such as the Haar, Symmlet, and BDCT *by ignoring the graph structure and viewing them as samples on the 1D regular lattice*. The resulting SNR values range from 5.01 dB to 6.98 dB, i.e., much worse than the graph-based denoising methods in general.

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.
- We also truncated this graph to have $2^{11} = 2048$ vertices, and denoised the data using the standard transforms such as the Haar, Symmlet, and BDCT *by ignoring the graph structure and viewing them as samples on the 1D regular lattice*. The resulting SNR values range from 5.01 dB to 6.98 dB, i.e., much worse than the graph-based denoising methods in general.

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.
- We also truncated this graph to have $2^{11} = 2048$ vertices, and denoised the data using the standard transforms such as the Haar, Symmlet, and BDCT *by ignoring the graph structure and viewing them as samples on the 1D regular lattice*. The resulting SNR values range from 5.01 dB to 6.98 dB, i.e., much worse than the graph-based denoising methods in general.

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.
- We also truncated this graph to have $2^{11} = 2048$ vertices, and denoised the data using the standard transforms such as the Haar, Symmlet, and BDCT *by ignoring the graph structure and viewing them as samples on the 1D regular lattice*. The resulting SNR values range from 5.01 dB to 6.98 dB, i.e., much worse than the graph-based denoising methods in general.

Outline

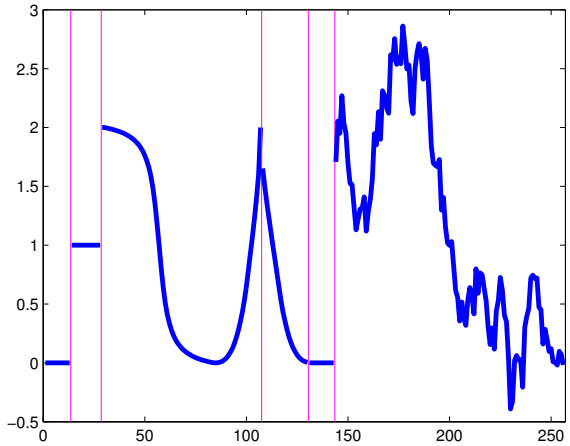
Motivation

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

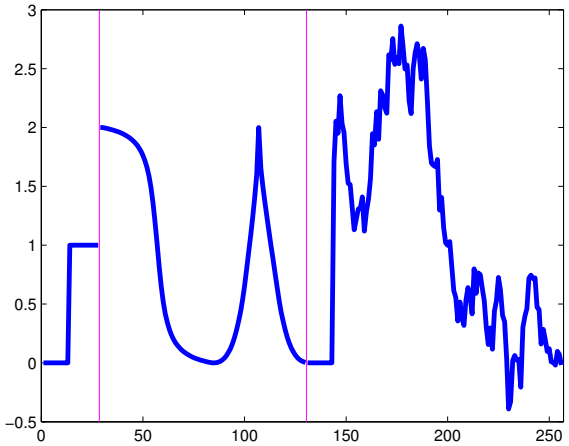
Motivation

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

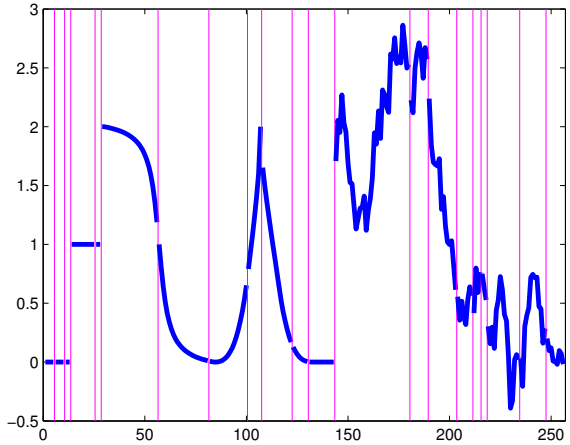
Simply put, the goal is to partition a given 1-D signal into segments based on the characteristics of the signal, which may help interpretation, analysis, compression, etc.



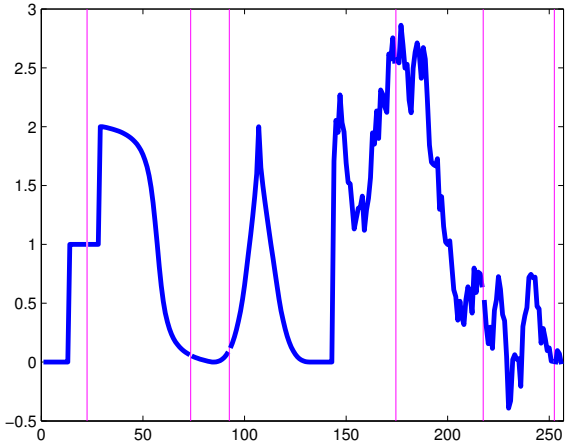
(a) Good



(b) Bad – too few segments



(c) Bad – too many segments



(d) Bad – segmentation lines are poorly placed

Method

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

Iterate until the best-basis segmentation converges:

1. **Recursively partition the graph:** Construct a recursive bipartitioning by minimizing *NCut* (*without* using the Fiedler vectors).
2. **Perform the 3 HGLET transforms:** Use the eigenvectors of L, L_{rw}, and L_{sym} of the *unweighted path graph*, i.e., three types of the DCTs (no eigenvector computation necessary).
3. **Find the hybrid best basis:** Use the MDL cost functional to search among the coefficients from the 3 HGLET variations.
4. **Modify the graph's edge weights:** If the two adjacent segments in the resulting best basis are represented by the same HGLET transform, then we double the weight of the edge joining them and halve the weights of its two neighboring edges.

Post-process: the segments of the final best basis and reconstruct the denoised signal from the quantized coefficients.

Method

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

Iterate until the best-basis segmentation converges:

1. **Recursively partition the graph:** Construct a recursive bipartitioning by minimizing *NCut* (*without* using the Fiedler vectors).
2. **Perform the 3 HGLET transforms:** Use the eigenvectors of L, L_{rw}, and L_{sym} of the *unweighted path graph*, i.e., three types of the DCTs (no eigenvector computation necessary).
3. **Find the hybrid best basis:** Use the MDL cost functional to search among the coefficients from the 3 HGLET variations.
4. **Modify the graph's edge weights:** If the two adjacent segments in the resulting best basis are represented by the same HGLET transform, then we double the weight of the edge joining them and halve the weights of its two neighboring edges.

Post-process: the segments of the final best basis and reconstruct the denoised signal from the quantized coefficients.

Method

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

Iterate until the best-basis segmentation converges:

1. **Recursively partition the graph:** Construct a recursive bipartitioning by minimizing *NCut* (*without* using the Fiedler vectors).
2. **Perform the 3 HGLET transforms:** Use the eigenvectors of L, L_{rw}, and L_{sym} of the *unweighted path graph*, i.e., three types of the DCTs (no eigenvector computation necessary).
3. **Find the hybrid best basis:** Use the MDL cost functional to search among the coefficients from the 3 HGLET variations.
4. **Modify the graph's edge weights:** If the two adjacent segments in the resulting best basis are represented by the same HGLET transform, then we double the weight of the edge joining them and halve the weights of its two neighboring edges.

Post-process: the segments of the final best basis and reconstruct the denoised signal from the quantized coefficients.

Method

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

Iterate until the best-basis segmentation converges:

1. **Recursively partition the graph:** Construct a recursive bipartitioning by minimizing *NCut* (*without* using the Fiedler vectors).
2. **Perform the 3 HGLET transforms:** Use the eigenvectors of L, L_{rw}, and L_{sym} of the *unweighted path graph*, i.e., three types of the DCTs (no eigenvector computation necessary).
3. **Find the hybrid best basis:** Use the MDL cost functional to search among the coefficients from the 3 HGLET variations.
4. **Modify the graph's edge weights:** If the two adjacent segments in the resulting best basis are represented by the same HGLET transform, then we double the weight of the edge joining them and halve the weights of its two neighboring edges.

Post-process: the segments of the final best basis and reconstruct the denoised signal from the quantized coefficients.

Method

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

Iterate until the best-basis segmentation converges:

1. **Recursively partition the graph:** Construct a recursive bipartitioning by minimizing *NCut* (*without* using the Fiedler vectors).
2. **Perform the 3 HGLET transforms:** Use the eigenvectors of L, L_{rw}, and L_{sym} of the *unweighted path graph*, i.e., three types of the DCTs (no eigenvector computation necessary).
3. **Find the hybrid best basis:** Use the MDL cost functional to search among the coefficients from the 3 HGLET variations.
4. **Modify the graph's edge weights:** If the two adjacent segments in the resulting best basis are represented by the same HGLET transform, then we double the weight of the edge joining them and halve the weights of its two neighboring edges.

Post-process: the segments of the final best basis and reconstruct the denoised signal from the quantized coefficients.

Method

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

Iterate until the best-basis segmentation converges:

1. **Recursively partition the graph:** Construct a recursive bipartitioning by minimizing *NCut* (*without* using the Fiedler vectors).
2. **Perform the 3 HGLET transforms:** Use the eigenvectors of L, L_{rw}, and L_{sym} of the *unweighted path graph*, i.e., three types of the DCTs (no eigenvector computation necessary).
3. **Find the hybrid best basis:** Use the MDL cost functional to search among the coefficients from the 3 HGLET variations.
4. **Modify the graph's edge weights:** If the two adjacent segments in the resulting best basis are represented by the same HGLET transform, then we double the weight of the edge joining them and halve the weights of its two neighboring edges.

Post-process: the segments of the final best basis and reconstruct the denoised signal from the quantized coefficients.

Minimum Description Length (MDL)

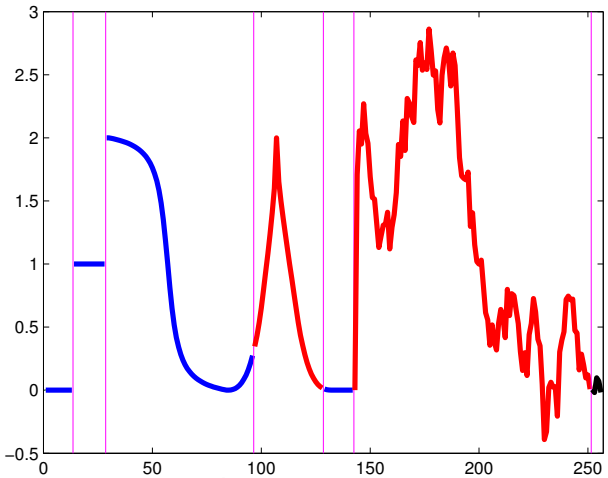
MDL principle: given 2 or more models for representing a signal, *choose the model that allows us to reconstruct the signal using the least amount of **bits***.

⇒ Need to specify the model, its parameters, and the expansion coefficients of the signal.

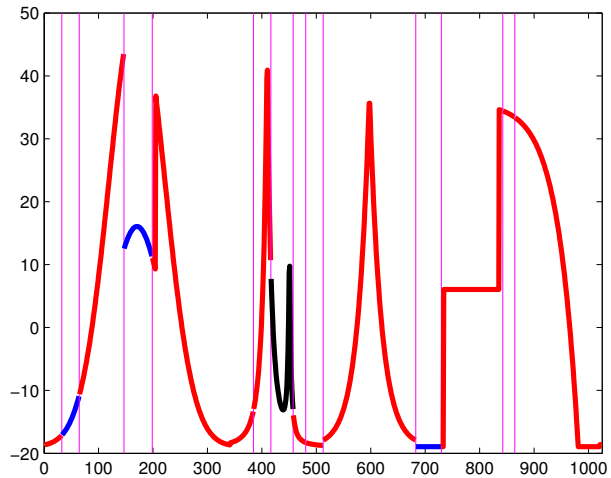
Our MDL costs:

1. Specify the segments $\boldsymbol{f}_i$ of the signal $\boldsymbol{f}$
2. Specify the transform for each segment (HGLET L, HGLET L_{rw}, or HGLET L_{sym})
3. The quantization precision $\delta_i = 2^{-k_i}$
4. The quantized expansion coefficients
5. The quantized noise $\hat{\sigma}_i^2$ (or error, if there is no noise in $\boldsymbol{f}_i$)
6. The codelength of $\boldsymbol{f}_i$ given (1)-(5)

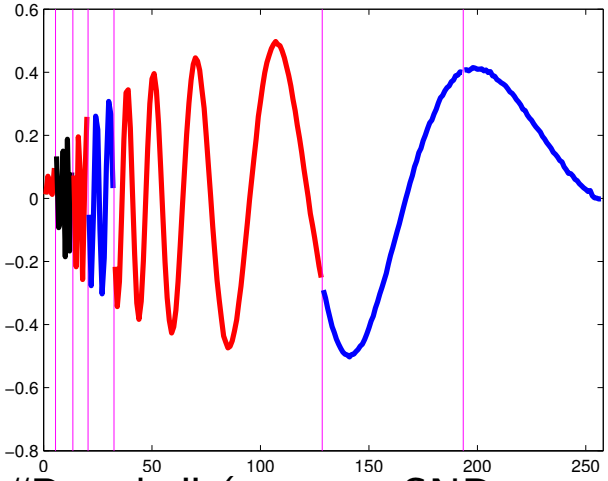
Results



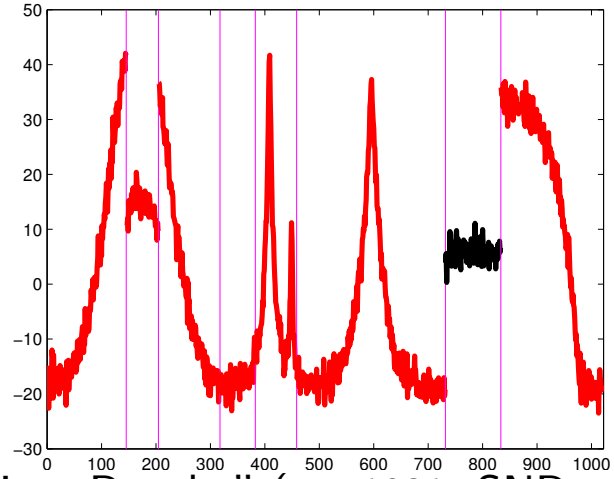
(a) "MSignal" ($n = 256$, SNR $= 200$ dB)



(c) "Piece-Regular" ($n = 1024$, SNR $= 200$ dB)



(b) "Doppler" ($n = 257$, SNR $= 40$ dB)



(d) "Piece-Regular" ($n = 1021$, SNR $= 20$ dB)

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

A Real Signal Example

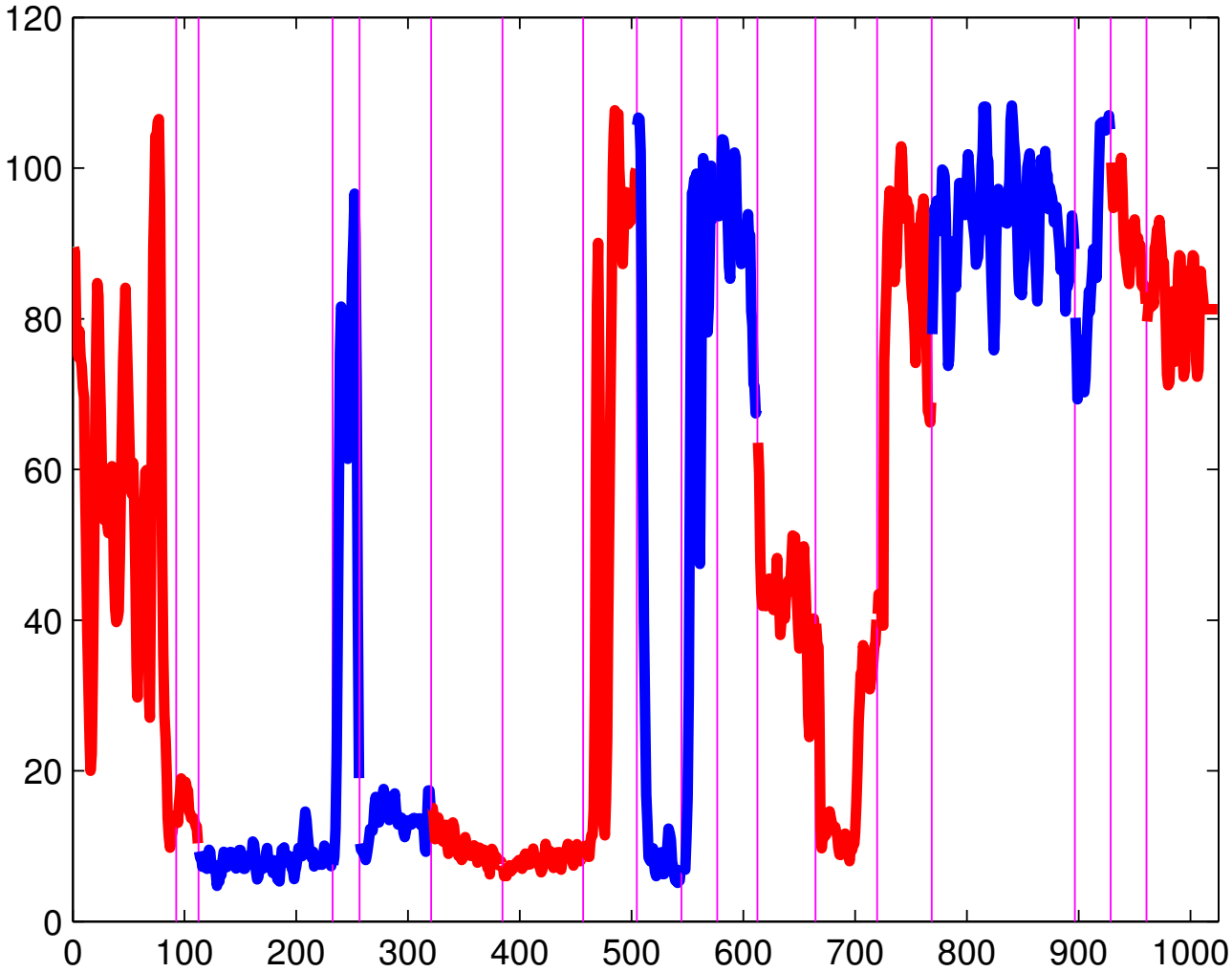
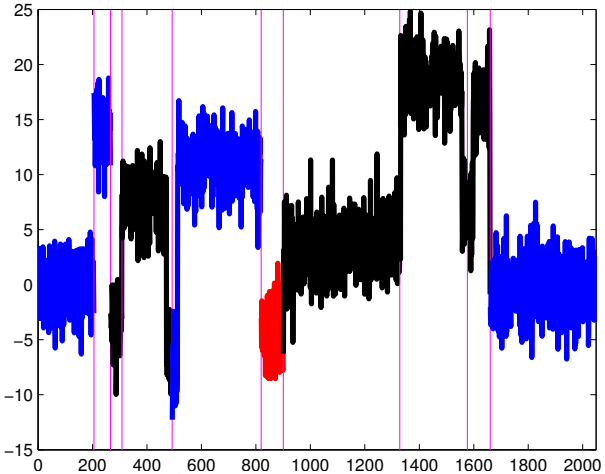
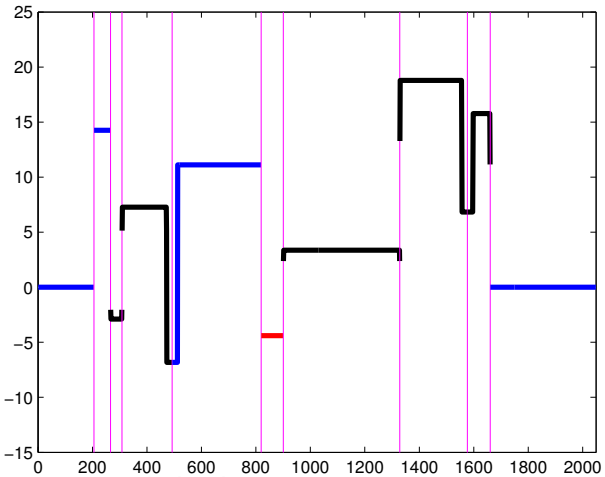


Figure: Gamma-ray log from North Sea subsurface formations. $\Delta z = 6$inches.

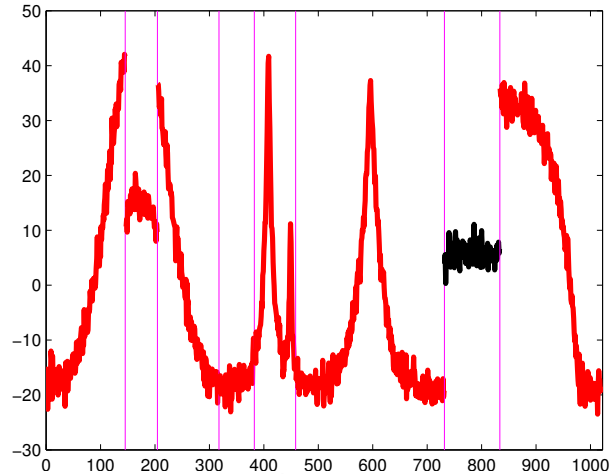
Simultaneous Segmentation and Denoising



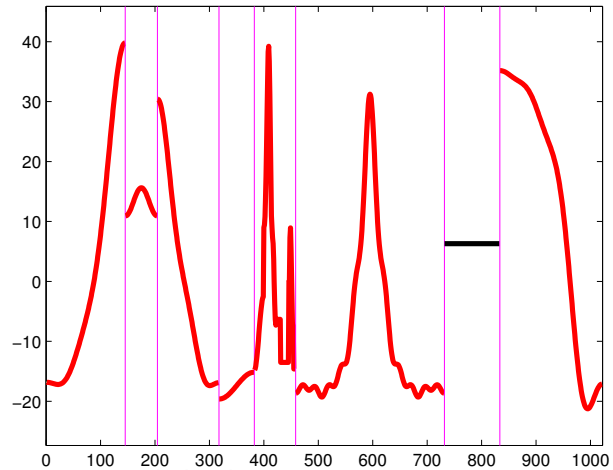
(a) "Blocks" ($n = 2048$, SNR $= 11.95$ dB)



(b) Denoised (a) ($n = 2048$, SNR $= 21.60$ dB)



(c) "Piece-Regular" ($n = 1021$, SNR $= 20$ dB)



(d) Denoised (c) ($n = 1021$, SNR $= 22.82$ dB)

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

Discussion

Why does the MDL perform well for this task?

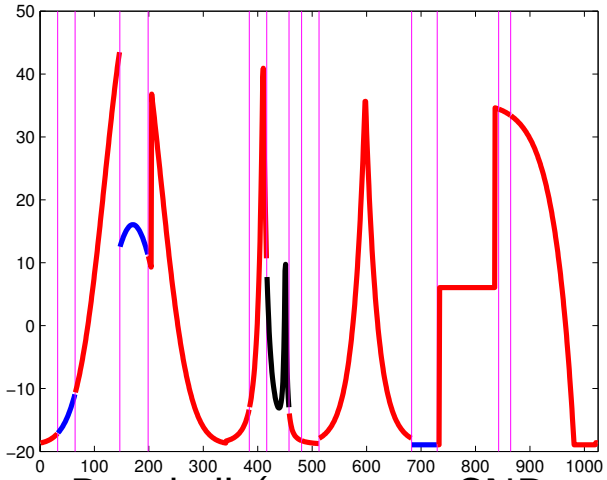
1. The MDL seeks an efficient way to represent the signal ⇒ dissimilar regions are more efficiently represented separately than together
2. Partitions have a cost, and so regions will be merged unless keeping them separate offers a savings in cost that warrants the extra cost of the partition

Discussion

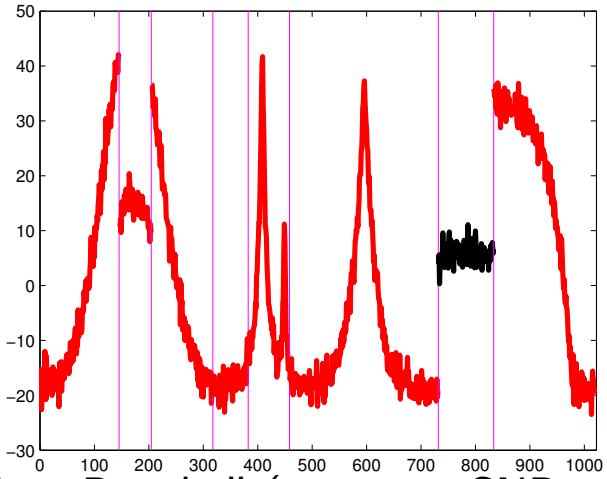
Why does the MDL perform well for this task?

1. The MDL seeks an efficient way to represent the signal ⇒ dissimilar regions are more efficiently represented separately than together
2. Partitions have a cost, and so regions will be merged unless keeping them separate offers a savings in cost that warrants the extra cost of the partition

Discussion ...



(c) "Piece-Regular" ($n = 1024$, SNR $= 200$ dB)



(d) "Piece-Regular" ($n = 1021$, SNR $= 20$ dB)

What do we learn from this particular example?

- Our method is versatile in that it is compatible with signals of arbitrary length $\Rightarrow$ signals whose length are non-dyadic and even prime are perfectly fine
- Even with the addition of significant noise and changing the length from dyadic to prime, the resulting partitions look similar