The Generalized Haar-Walsh Transform (GHWT) for Data Analysis on Graphs and Networks

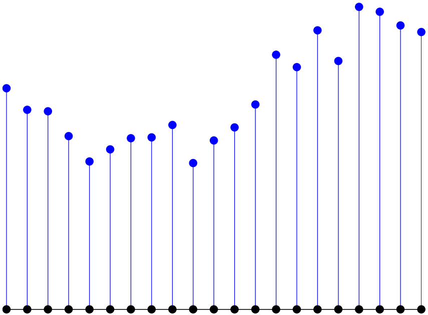
Jeff Irion & Naoki Saito

Department of Mathematics
University of California, Davis

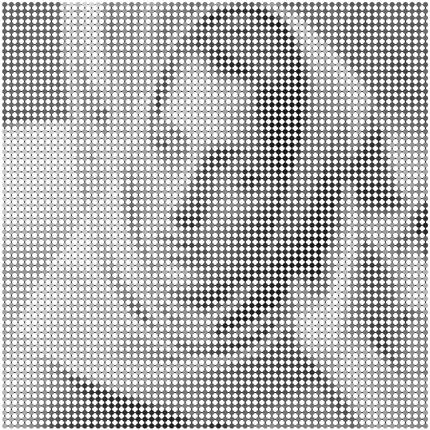
SIAM Annual Meeting 2014
Chicago, IL
July 7, 2014

Motivation

Classical signals can be viewed as signals on graphs with simple structures. For example:



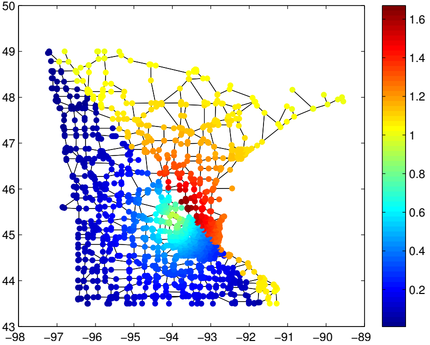
(a) A portion of a 1D signal



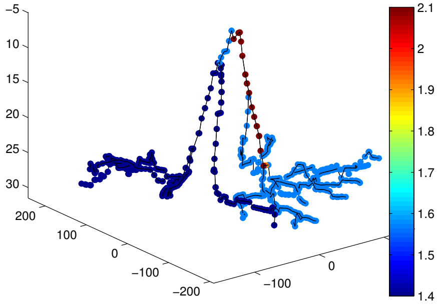
(b) Barbara

Motivation

We wish to extend techniques from classical signal processing to the setting of general graphs. For example:



(a) A mutilated Gaussian signal on the MN road network



(b) Thickness data on a dendritic tree

Motivation

Aims & Objectives:

1. Develop an overcomplete multiscale transform for signals on graphs
2. Develop a corresponding best-basis search algorithm
3. Investigate usefulness for approximation, denoising, classification, function estimation, etc.

Challenges:

1. Irregular structure of the domain
2. Lack of translation, dilation, and a general notion of frequency
   - Critical elements in the wavelet transform
3. Computational complexity!

Motivation

Aims & Objectives:

1. Develop an overcomplete multiscale transform for signals on graphs
2. Develop a corresponding best-basis search algorithm
3. Investigate usefulness for approximation, denoising, classification, function estimation, etc.

Challenges:

1. Irregular structure of the domain
2. Lack of translation, dilation, and a general notion of frequency
    - Critical elements in the wavelet transform
3. **Computational complexity!**

Definitions and Notation

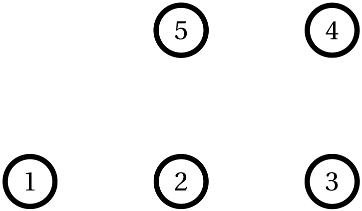
Let G be a graph.

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

Definitions and Notation

Let G be a graph.

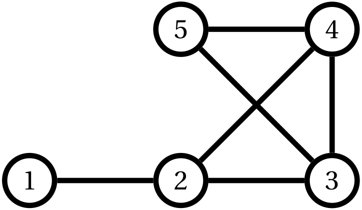
- $V = V(G) = \{v_1, \ldots, v_N\}$ is the set of vertices.
- $E = E(G) = \{e_1, \ldots, e_{N'}\}$ is the set of edges, where $e_k = (v_i, v_j)$ represents an edge (or line segment) connecting between adjacent vertices v_i, v_j for some $1 \le i, j \le N$.
- $W = W(G) \in \mathbb{R}^{N \times N}$ is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



Definitions and Notation

Let G be a graph.

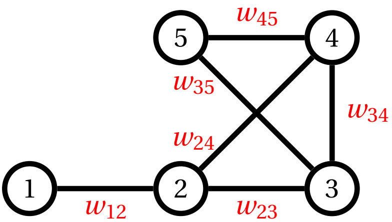
- $V = V(G) = \{v_1, \dots, v_N\}$ is the set of vertices.
- $E = E(G) = \{e_1, \dots, e_{N'}\}$ is the set of edges, where $e_k = (v_i, v_j)$ represents an edge (or line segment) connecting between adjacent vertices v_i, v_j for some $1 \le i, j \le N$.
- $W = W(G) \in \mathbb{R}^{N \times N}$ is the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



Definitions and Notation

Let G be a graph.

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



Our Assumptions

In this talk, we assume that the graph is

- **connected.**
- **undirected.** $w_{ij} = w_{ji}$, and thus W is *symmetric*.

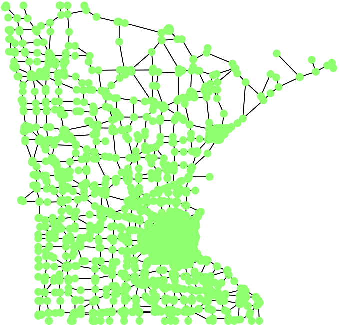
Recursive Partitioning

Our transform requires as input a recursive partitioning of the graph.

- We used Fielder vectors of the Laplacian matrices.
- The associated cost is from $O(N \log N)$ to $O(N^2)$, depending on the graph and the implementation.
- More info ⇒ J. Irion, N. Saito: "The Generalized Haar-Walsh Transform," *Proceedings of 2014 IEEE Workshop on Statistical Signal Processing*, pp. 488–491, 2014.

Recursive Partitioning

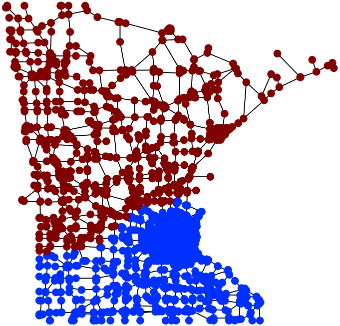
Our transform requires as input a recursive partitioning of the graph.



- We used Fielder vectors of the Laplacian matrices.
- The associated cost is from $O(N \log N)$ to $O(N^2)$, depending on the graph and the implementation.
- More info ⇒ J. Irion, N. Saito: "The Generalized Haar-Walsh Transform," *Proceedings of 2014 IEEE Workshop on Statistical Signal Processing*, pp. 488–491, 2014.

Recursive Partitioning

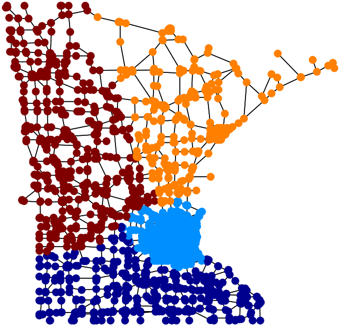
Our transform requires as input a recursive partitioning of the graph.



- We used Fielder vectors of the Laplacian matrices.
- The associated cost is from $O(N \log N)$ to $O(N^2)$, depending on the graph and the implementation.
- More info ⇒ J. Irion, N. Saito: "The Generalized Haar-Walsh Transform," *Proceedings of 2014 IEEE Workshop on Statistical Signal Processing*, pp. 488–491, 2014.

Recursive Partitioning

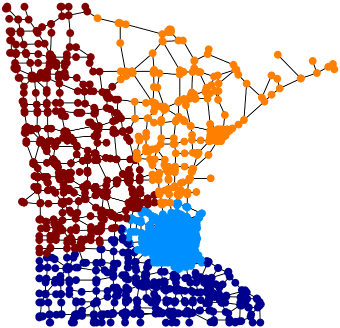
Our transform requires as input a recursive partitioning of the graph.



- We used Fielder vectors of the Laplacian matrices.
- The associated cost is from $O(N \log N)$ to $O(N^2)$, depending on the graph and the implementation.
- More info ⇒ J. Irion, N. Saito: "The Generalized Haar-Walsh Transform," *Proceedings of 2014 IEEE Workshop on Statistical Signal Processing*, pp. 488–491, 2014.

Recursive Partitioning

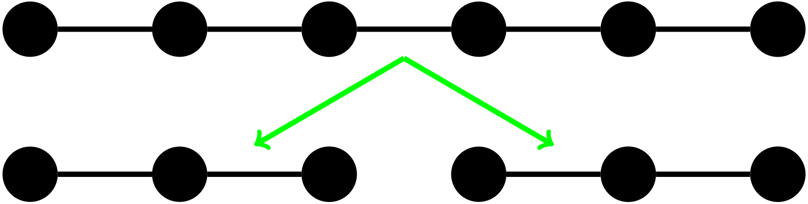
Our transform requires as input a recursive partitioning of the graph.



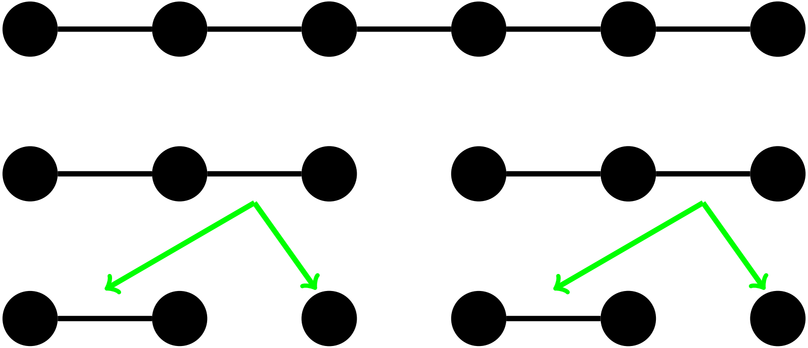
- We used Fielder vectors of the Laplacian matrices.
- The associated cost is from $O(N \log N)$ to $O(N^2)$, depending on the graph and the implementation.
- More info ⇒ J. Irion, N. Saito: "The Generalized Haar-Walsh Transform," *Proceedings of 2014 IEEE Workshop on Statistical Signal Processing*, pp. 488–491, 2014.

GHWT on P_6

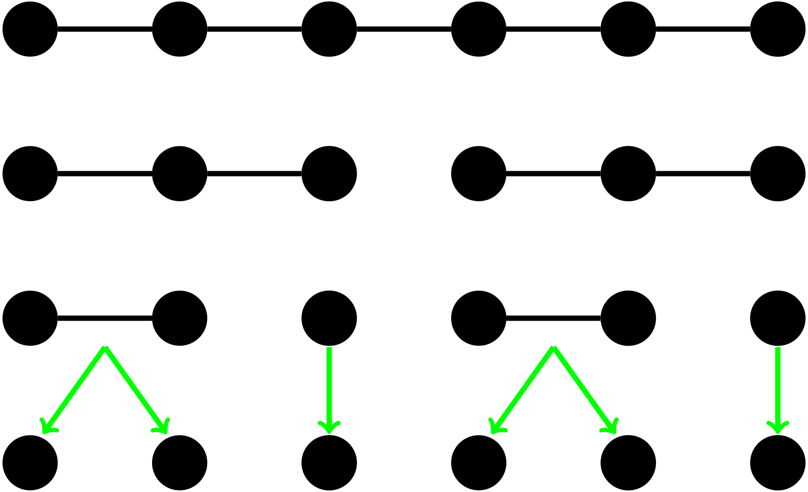
GHWT on P_6



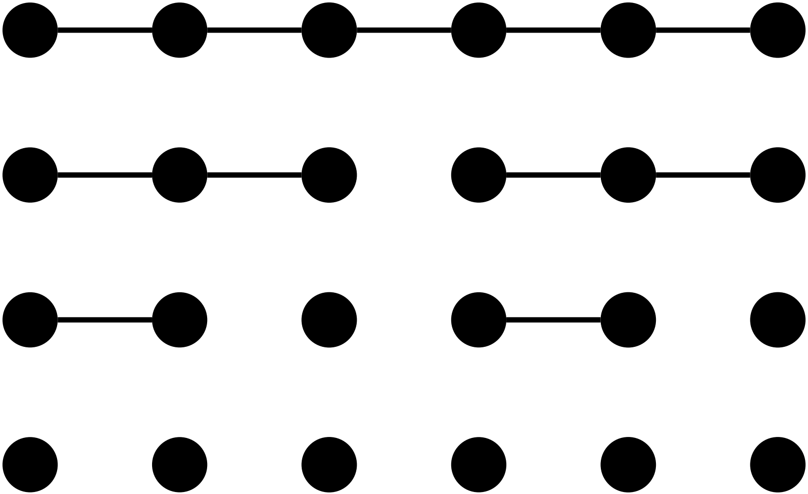
GHWT on P_6



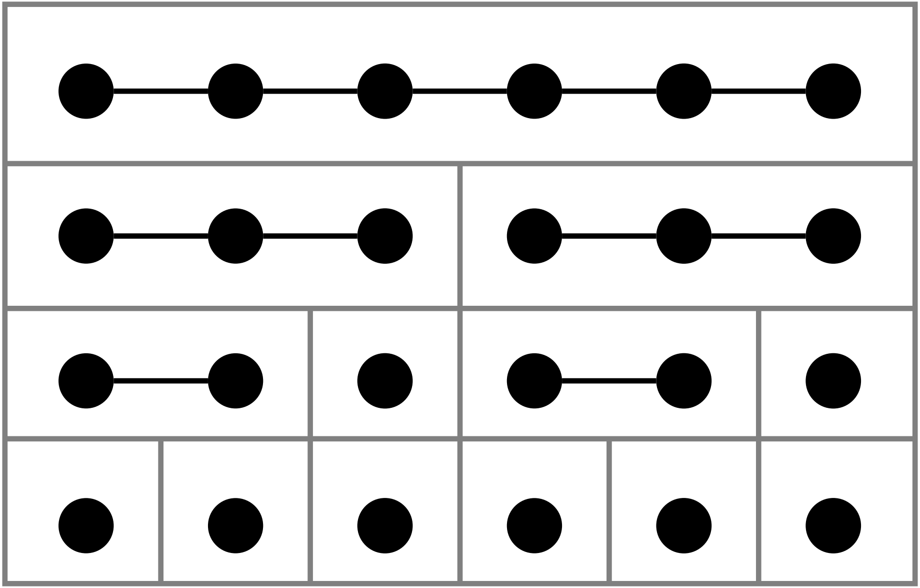
GHWT on P_6



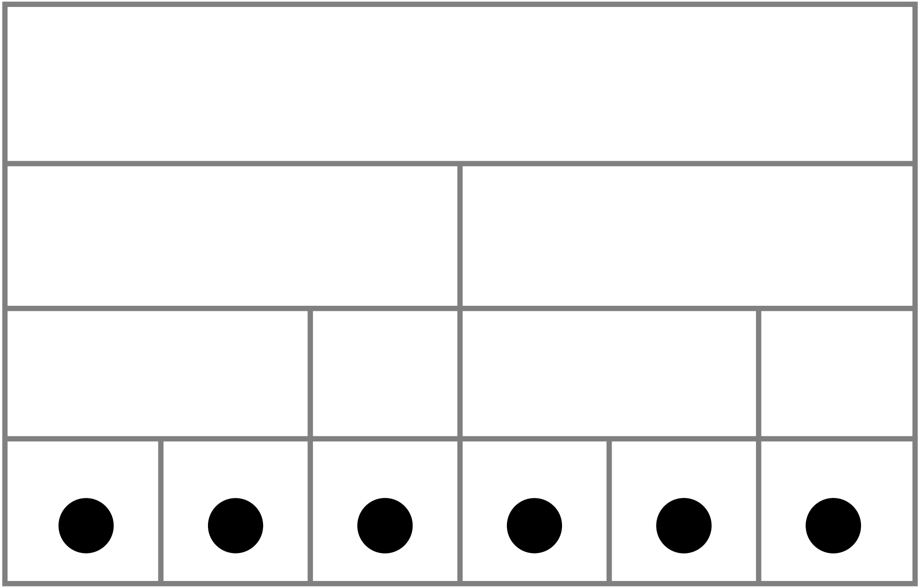
GHWT on P_6



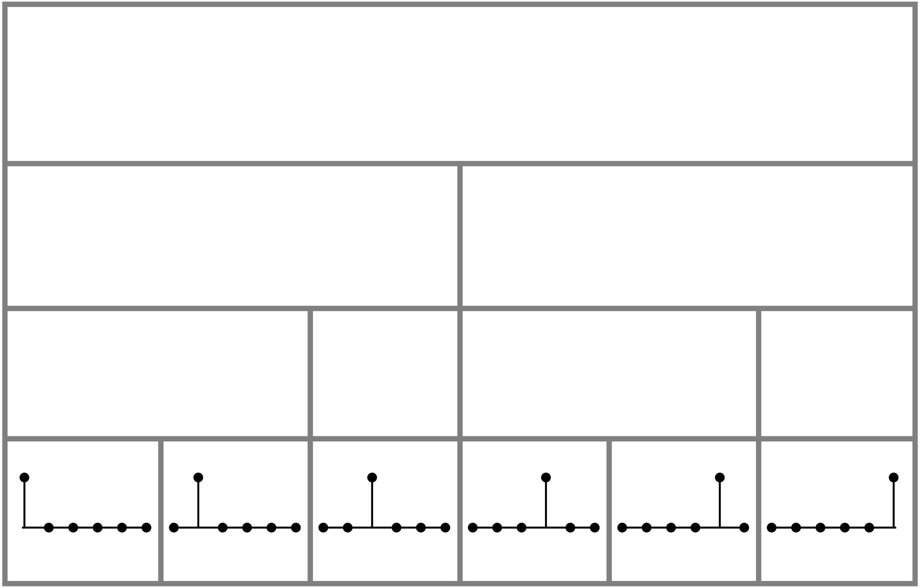
GHWT on P_6



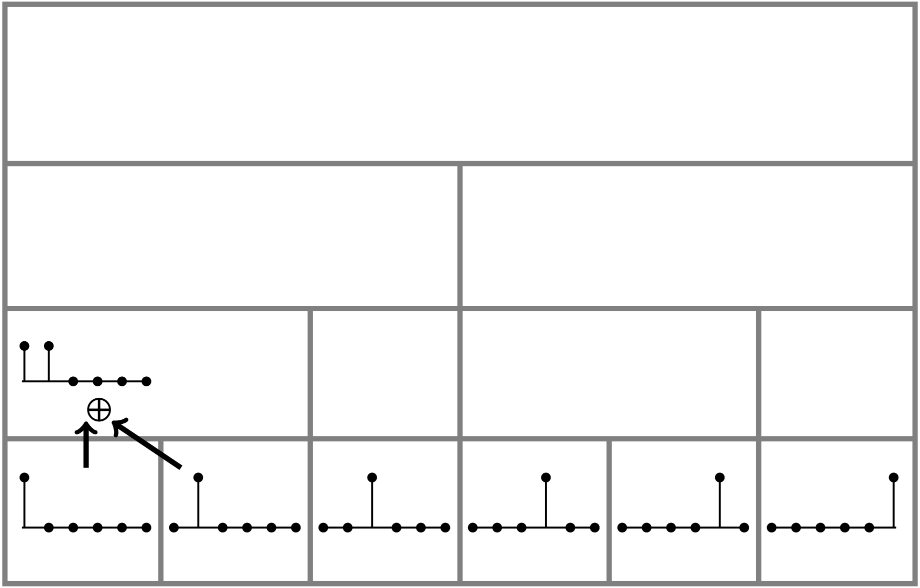
GHWT on P_6



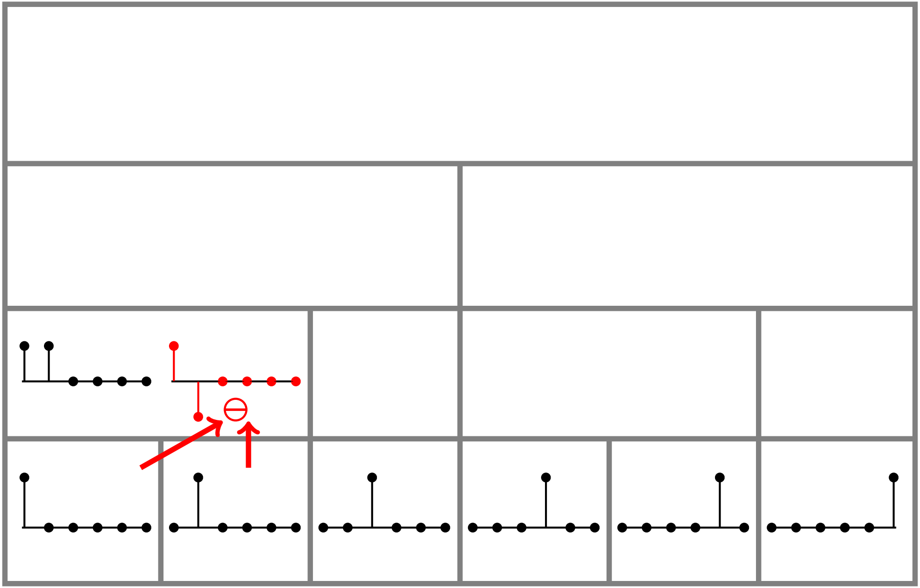
GHWT on P_6



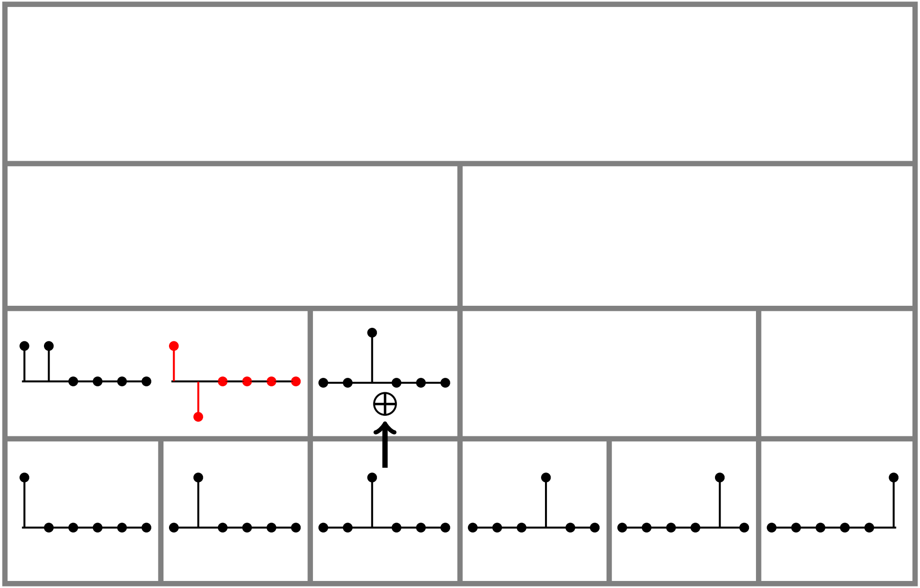
GHWT on P_6



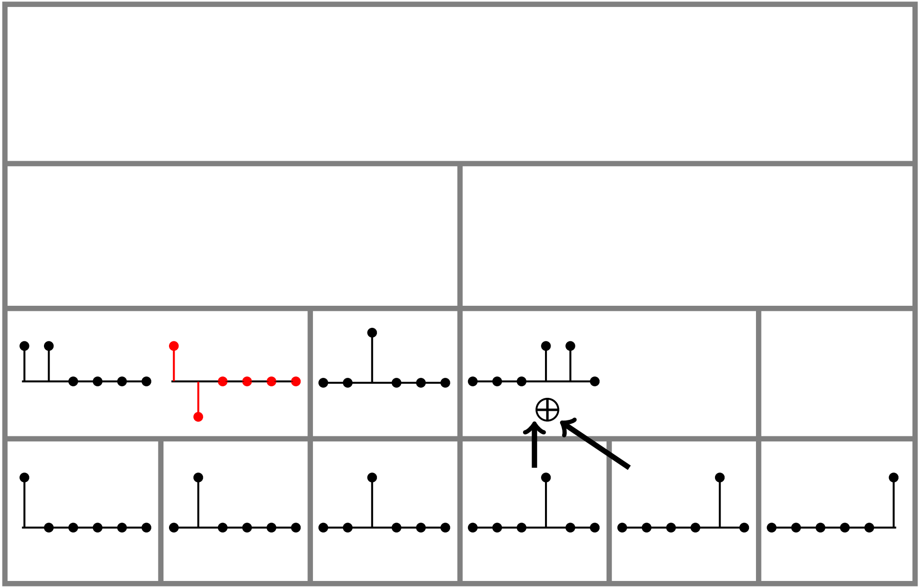
GHWT on P_6



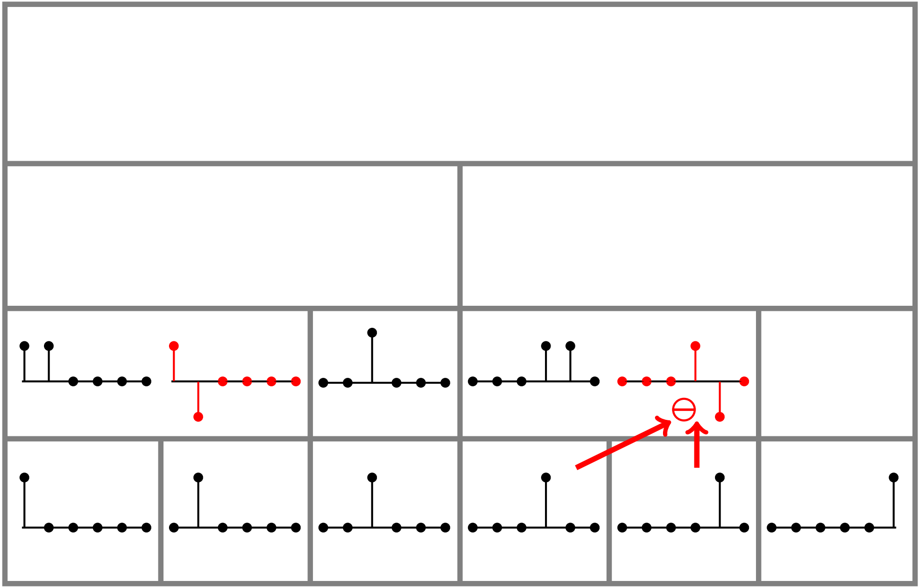
GHWT on P_6



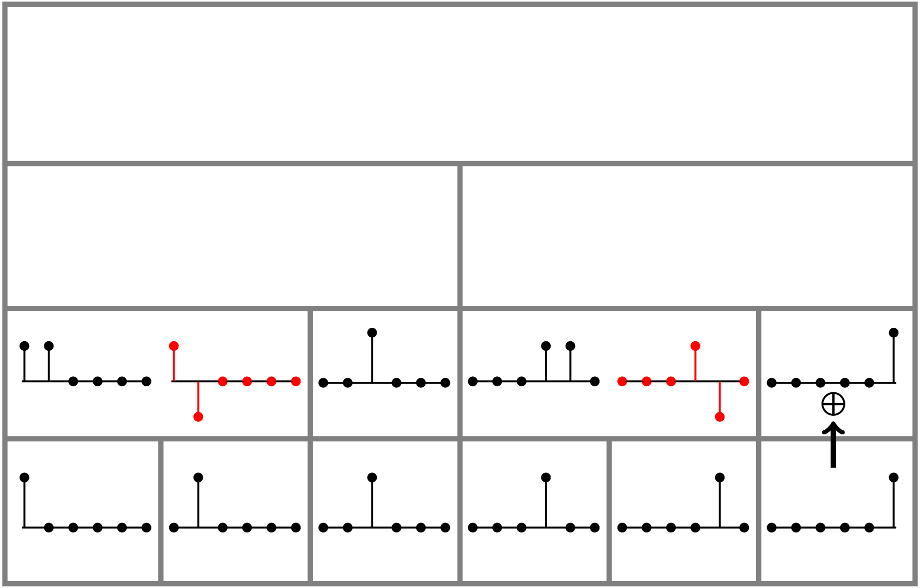
GHWT on P_6



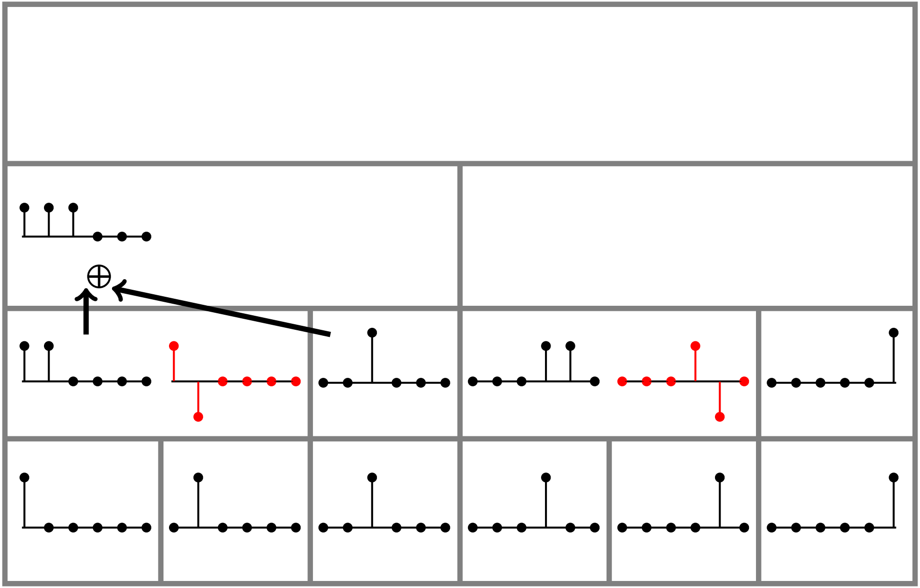
GHWT on P_6



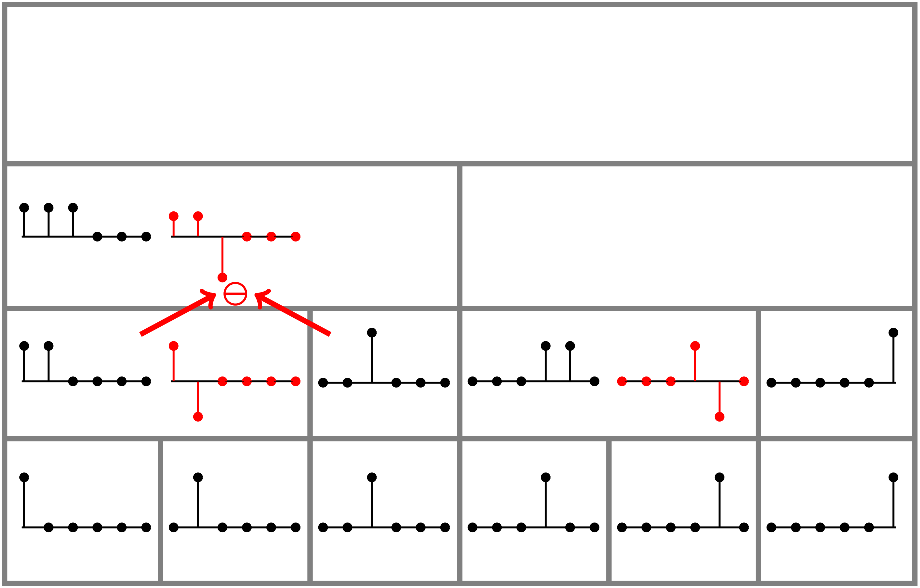
GHWT on P_6



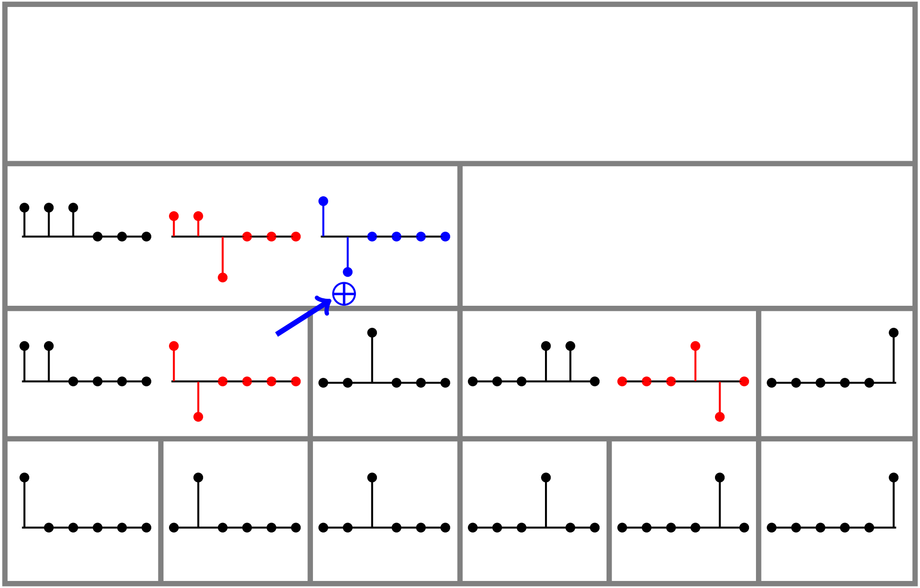
GHWT on P_6



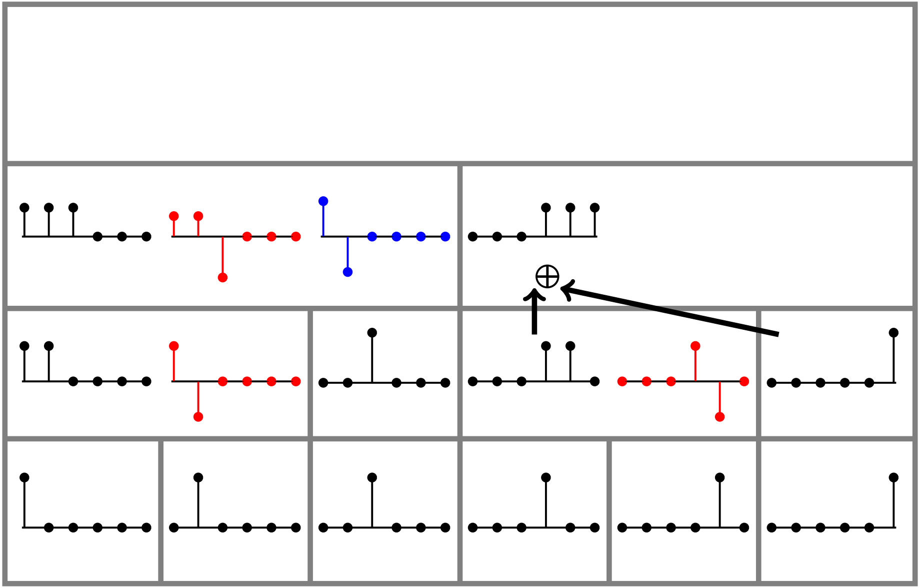
GHWT on P_6



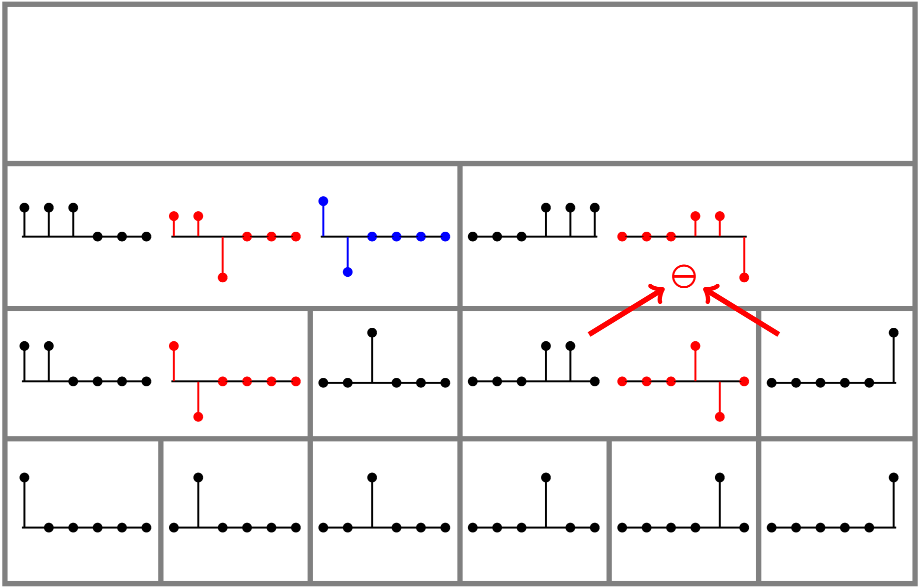
GHWT on P_6



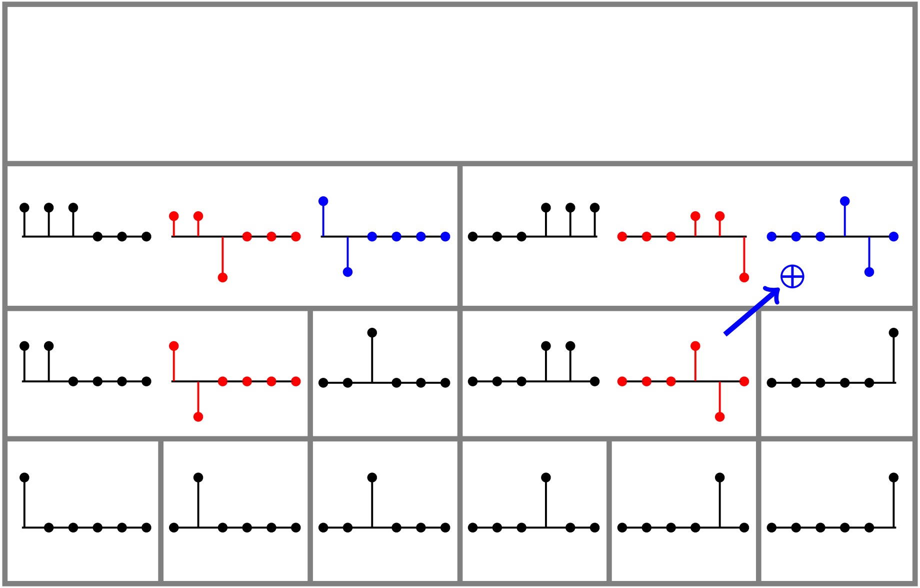
GHWT on P_6



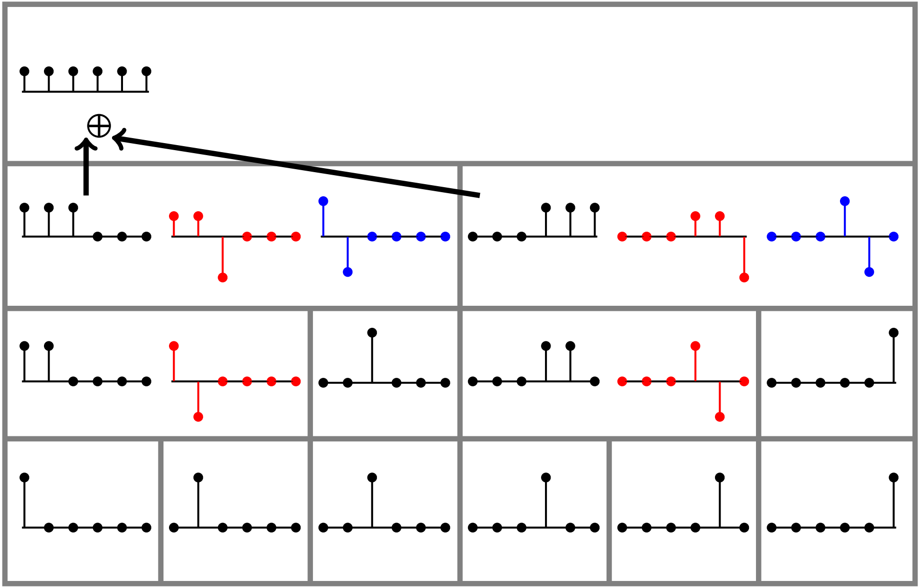
GHWT on P_6



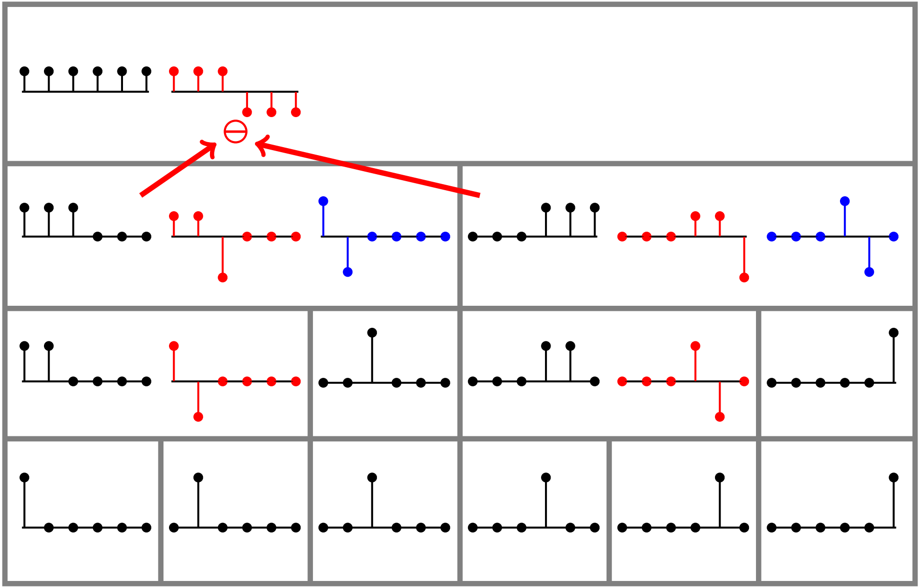
GHWT on P_6



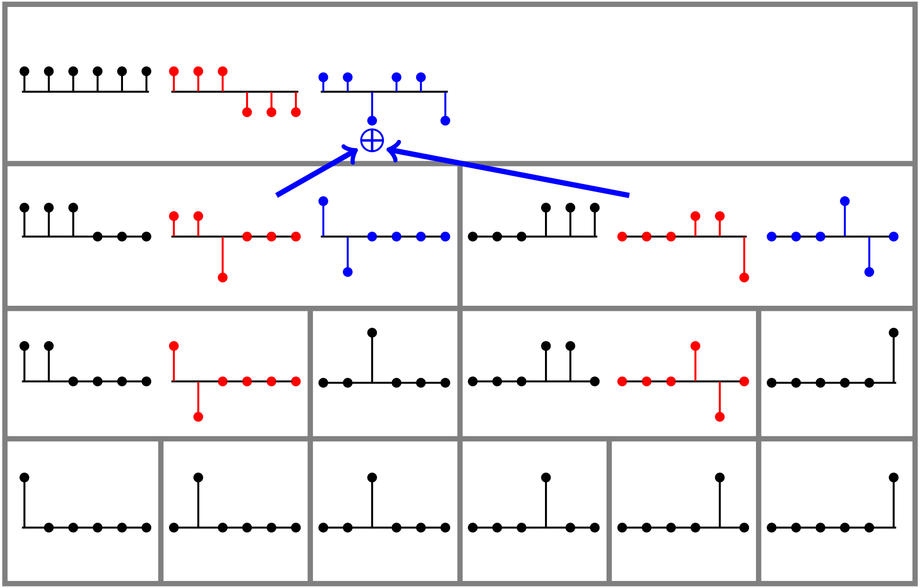
GHWT on P_6



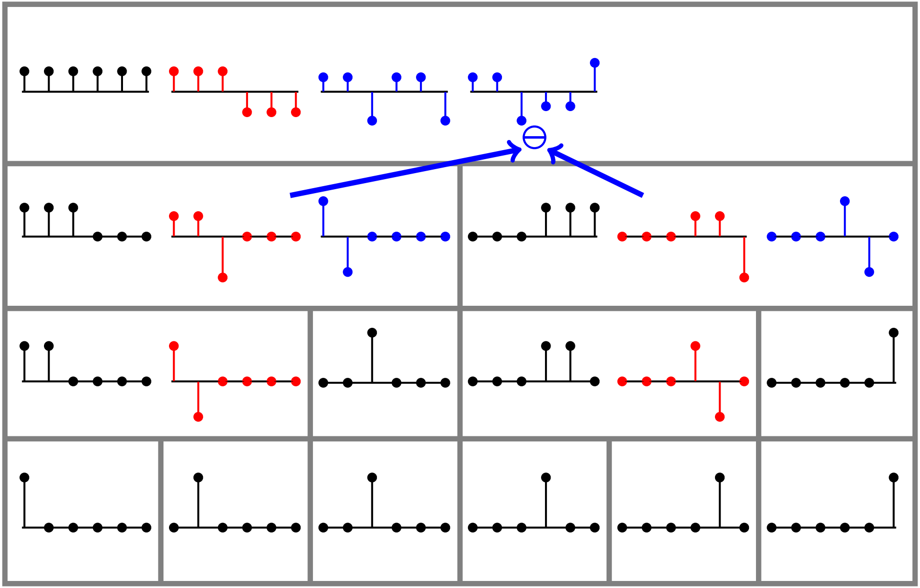
GHWT on P_6



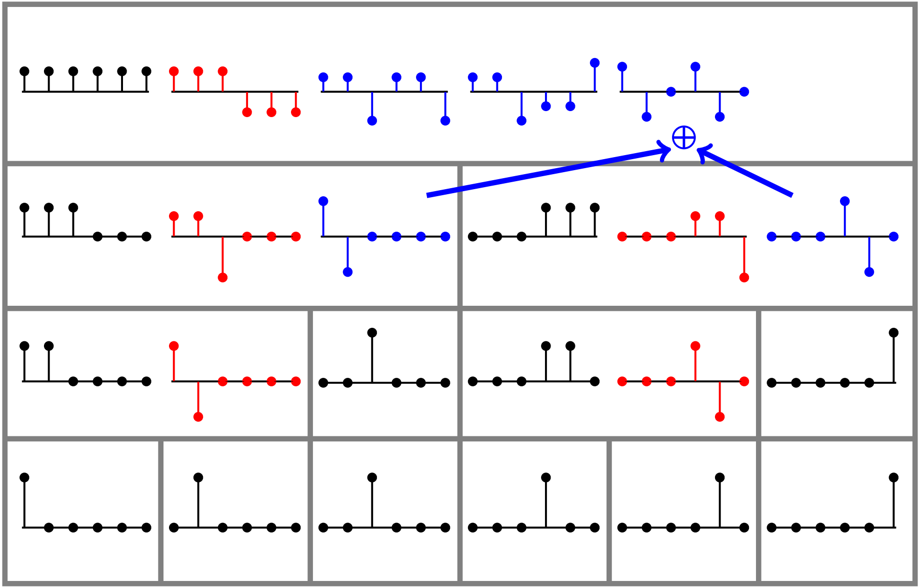
GHWT on P_6



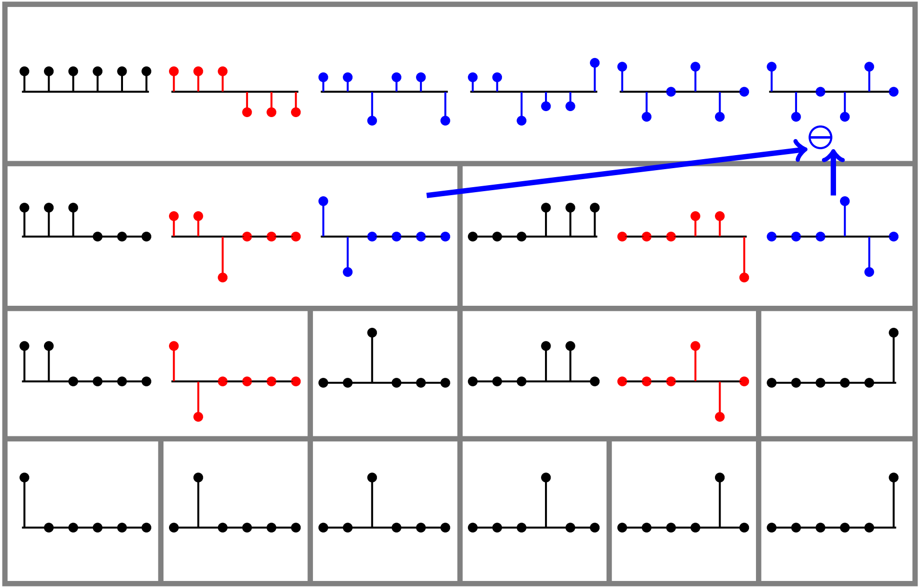
GHWT on P_6



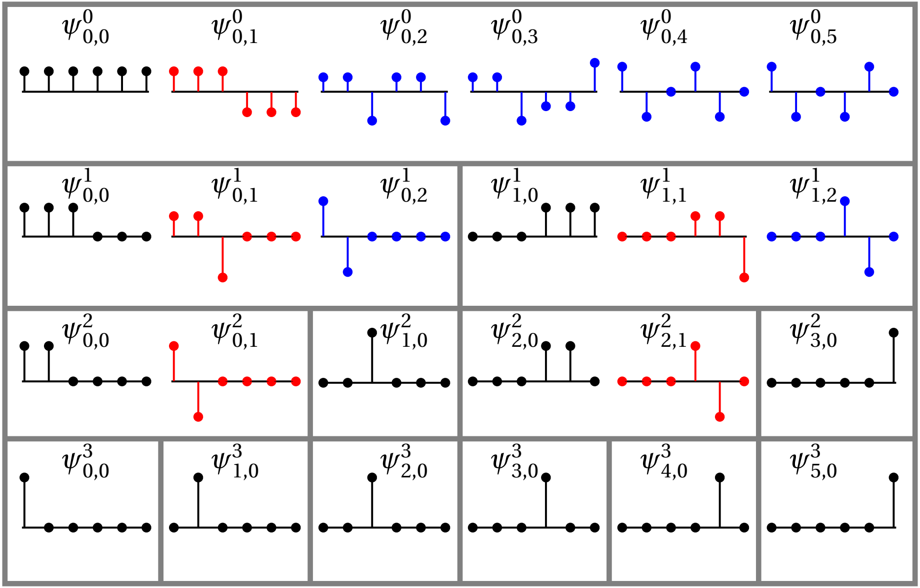
GHWT on P_6



GHWT on P_6

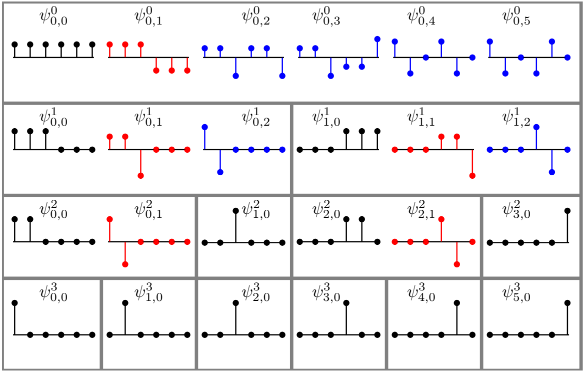


GHWT on P_6


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

GHWT on P_6 – Coarse-to-Fine Dictionary

We call this the *coarse-to-fine dictionary*.



Notation is $\psi^j_{k,\ell}$, where

- j is the level
- k denotes the region on level j
- ℓ is the *tag*

We have 3 types of basis vectors:

- **scaling vectors** $(\ell = 0)$
- **Haar-like vectors** $(\ell = 1)$
- **Walsh-like vectors** $(\ell \geq 2)$

GHWT Algorithm

Input: a graph partitioning and a signal $f \in \mathbb{R}^N$ on the graph

Output: a dictionary of expansion coefficients $\{d^j_{k,l}\}$

Step 1: Obtain expansion coefficients on level $j = j_{\max}$ ⇔ reorder the original signal

Step 2: Use the coefficients on level j to compute the coefficients on level $j-1$ as follows:

Step 2a: Compute the scaling coefficient

Case 1: **If** $N^j_k = 1$, **then** set

$$d^{j-1}_{k,0} := d^j_{k',0}$$

Case 2: **If** $N^j_k \geq 2$, **then** set

$$d^{j-1}_{k,0} := \frac{\sqrt{N^j_{k'}}\, d^j_{k',0} + \sqrt{N^j_{k'+1}}\, d^j_{k'+1,0}}{\sqrt{N^{j-1}_k}}$$

Step 2b: **If** $N_k^{j-1} \geq 2$, **then** compute the Haar-like coefficient as

$$d_{k,0}^{j-1} := \frac{\sqrt{N_{k'+1}^{j}}\, d_{k',0}^{j} - \sqrt{N_{k'}^{j}}\, d_{k'+1,0}^{j}}{\sqrt{N_k^{j-1}}}$$

Step 2c: **If** $N_k^{j-1} \geq 3$, **then** compute the Walsh-like coefficients. **For** $\ell = 1, \ldots, 2^{j_{\max}-j} - 1$:

Case 1: **If** neither subregion has a coefficient with tag ℓ, **then** do nothing

Case 2: **If** (without loss of generality) only subregion k' has a coefficient with tag ℓ, **then** set

$$d_{k,2\ell}^{j-1} := d_{k',\ell}^{j}$$

Case 3: **If** both subregions have coefficients with tag ℓ, **then** compute

$$d_{k,2\ell}^{j-1} := \left(d_{k',\ell}^{j} + d_{k'+1,\ell}^{j} \right) \Big/ \sqrt{2}$$
$$d_{k,2\ell+1}^{j-1} := \left(d_{k',\ell}^{j} - d_{k'+1,\ell}^{j} \right) \Big/ \sqrt{2}$$

GHWT on P_6 – Fine-to-Coarse Dictionary

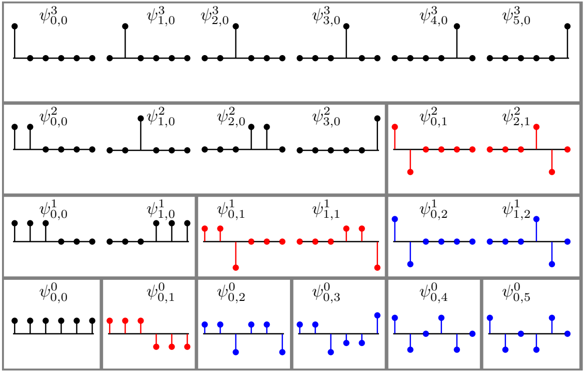
Note that the basis vectors (or coefficients) with tag ℓ on level j were used to generate those with tags 2ℓ and $2\ell+1$ on level $j-1$.

Using this fact, we can reorganize the basis vectors by their tags to yield the *fine-to-coarse dictionary*:

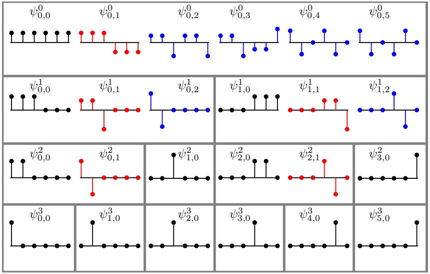
GHWT on P_6 – Fine-to-Coarse Dictionary

Note that the basis vectors (or coefficients) with tag ℓ on level j were used to generate those with tags 2ℓ and $2\ell+1$ on level $j-1$.

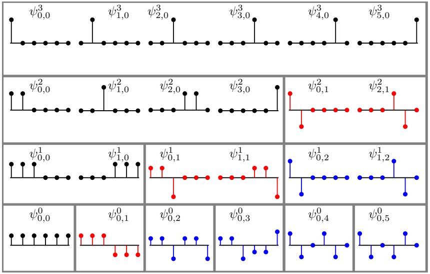
Using this fact, we can reorganize the basis vectors by their tags to yield the *fine-to-coarse dictionary*:



GHWT on P_6



(a) Coarse-to-fine dictionary



(b) Fine-to-coarse dictionary

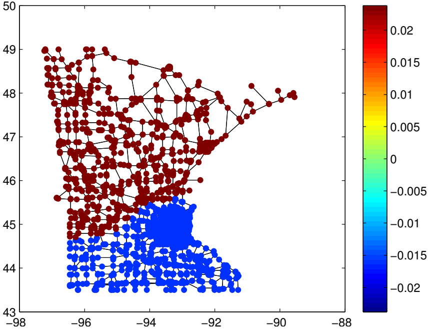
GHWT on MN Road Network

- $j = 0$ is the coarsest level, $j = 16$ is the finest
- Inverse Euclidean weights, partitioned via the Fiedler vector of L_{rw}

GHWT on MN Road Network

- $j = 0$ is the coarsest level, $j = 16$ is the finest
- Inverse Euclidean weights, partitioned via the Fiedler vector of L_{rw}

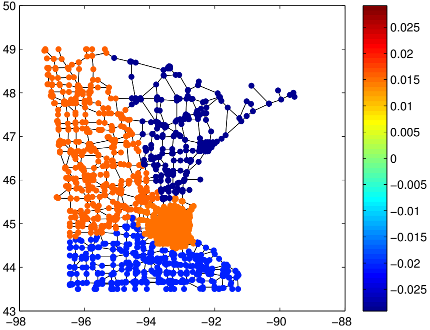
Level $j = 0$, Region $k = 0$, $l = 1$



GHWT on MN Road Network

- $j = 0$ is the coarsest level, $j = 16$ is the finest
- Inverse Euclidean weights, partitioned via the Fiedler vector of L_{rw}

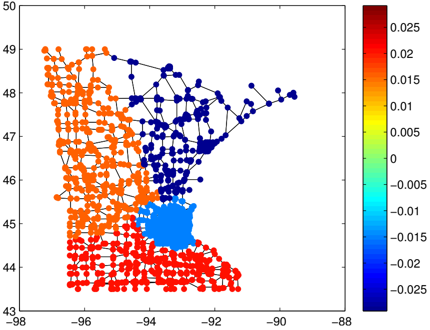
Level $j = 0$, Region $k = 0$, $l = 2$



GHWT on MN Road Network

- $j = 0$ is the coarsest level, $j = 16$ is the finest
- Inverse Euclidean weights, partitioned via the Fiedler vector of L_{rw}

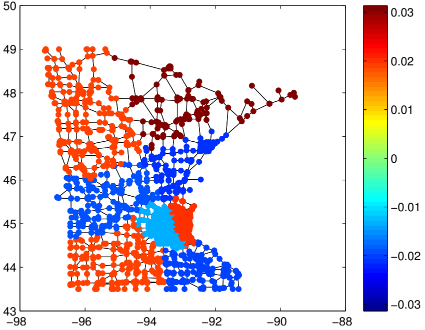
Level $j = 0$, Region $k = 0$, $l = 3$



GHWT on MN Road Network

- $j = 0$ is the coarsest level, $j = 16$ is the finest
- Inverse Euclidean weights, partitioned via the Fiedler vector of L_{rw}

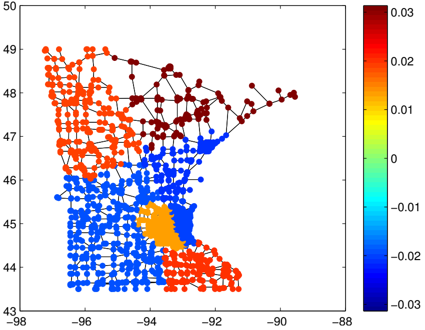
Level $j = 0$, Region $k = 0$, $l = 4$



GHWT on MN Road Network

- $j = 0$ is the coarsest level, $j = 16$ is the finest
- Inverse Euclidean weights, partitioned via the Fiedler vector of L_{rw}

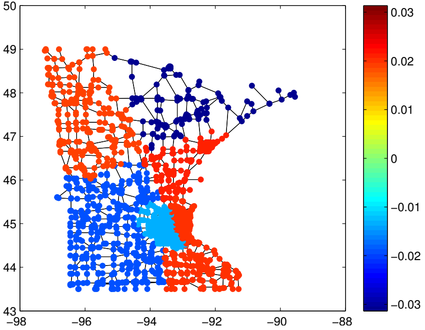
Level $j = 0$, Region $k = 0$, $l = 5$



GHWT on MN Road Network

- $j = 0$ is the coarsest level, $j = 16$ is the finest
- Inverse Euclidean weights, partitioned via the Fiedler vector of L_{rw}

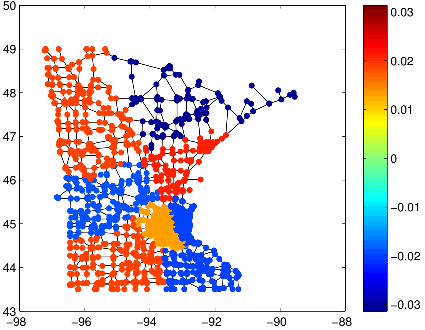
Level $j = 0$, Region $k = 0$, $l = 6$



GHWT on MN Road Network

- $j = 0$ is the coarsest level, $j = 16$ is the finest
- Inverse Euclidean weights, partitioned via the Fiedler vector of L_{rw}

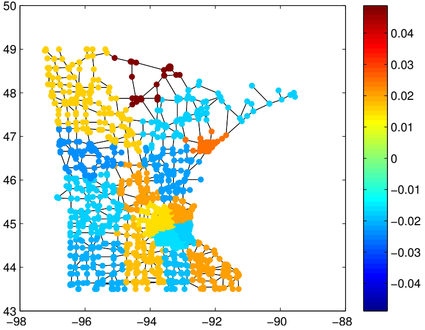
Level $j = 0$, Region $k = 0$, $l = 7$



GHWT on MN Road Network

- $j = 0$ is the coarsest level, $j = 16$ is the finest
- Inverse Euclidean weights, partitioned via the Fiedler vector of L_{rw}

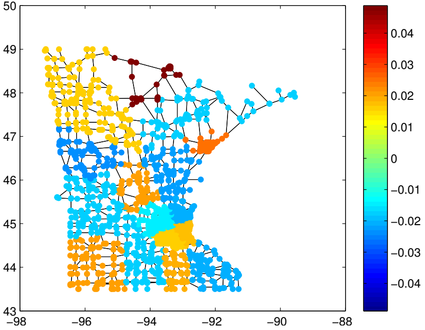
Level $j = 0$, Region $k = 0$, $l = 8$



GHWT on MN Road Network

- $j = 0$ is the coarsest level, $j = 16$ is the finest
- Inverse Euclidean weights, partitioned via the Fiedler vector of L_{rw}

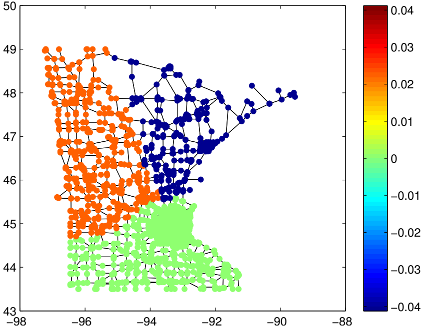
Level $j = 0$, Region $k = 0$, $l = 9$



GHWT on MN Road Network

- $j = 0$ is the coarsest level, $j = 16$ is the finest
- Inverse Euclidean weights, partitioned via the Fiedler vector of L_{rw}

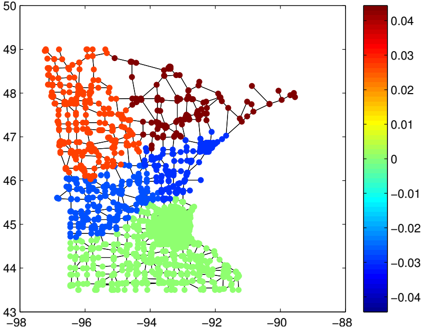
Level $j = 1$, Region $k = 0$, $l = 1$



GHWT on MN Road Network

- $j = 0$ is the coarsest level, $j = 16$ is the finest
- Inverse Euclidean weights, partitioned via the Fiedler vector of L_{rw}

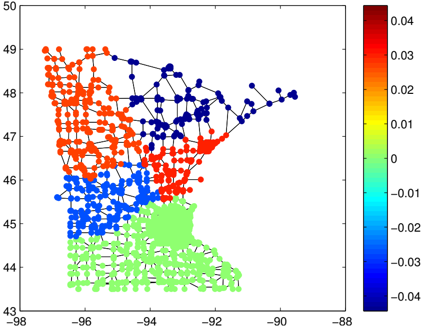
Level $j = 1$, Region $k = 0$, $l = 2$



GHWT on MN Road Network

- $j = 0$ is the coarsest level, $j = 16$ is the finest
- Inverse Euclidean weights, partitioned via the Fiedler vector of L_{rw}

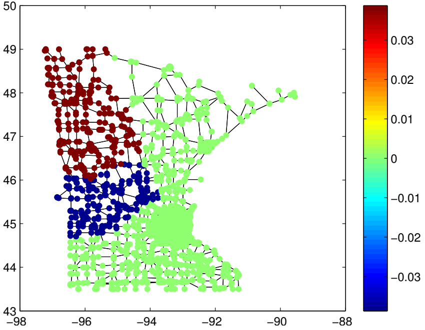
Level $j = 1$, Region $k = 0$, $l = 3$



GHWT on MN Road Network

- $j = 0$ is the coarsest level, $j = 16$ is the finest
- Inverse Euclidean weights, partitioned via the Fiedler vector of L_{rw}

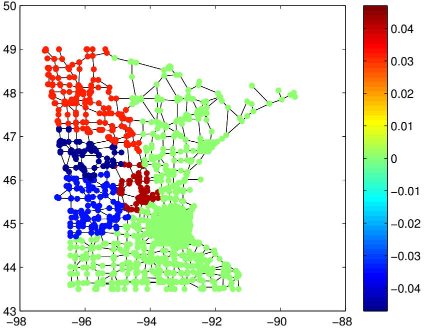
Level $j = 2$, Region $k = 0$, $l = 1$



GHWT on MN Road Network

- $j = 0$ is the coarsest level, $j = 16$ is the finest
- Inverse Euclidean weights, partitioned via the Fiedler vector of L_{rw}

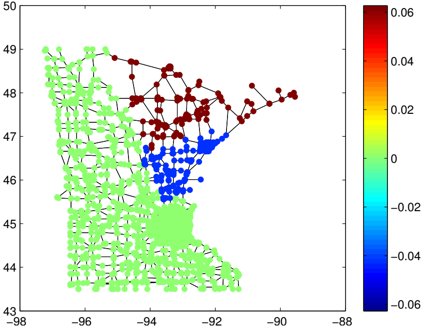
Level $j = 2$, Region $k = 0$, $l = 2$



GHWT on MN Road Network

- $j = 0$ is the coarsest level, $j = 16$ is the finest
- Inverse Euclidean weights, partitioned via the Fiedler vector of L_{rw}

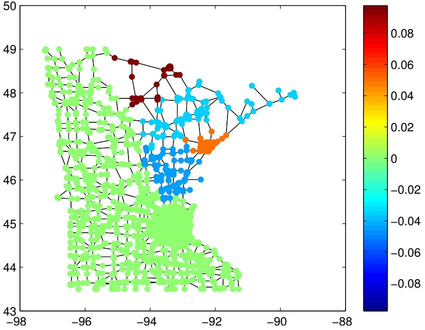
Level $j = 2$, Region $k = 1$, $l = 1$



GHWT on MN Road Network

- $j = 0$ is the coarsest level, $j = 16$ is the finest
- Inverse Euclidean weights, partitioned via the Fiedler vector of L_{rw}

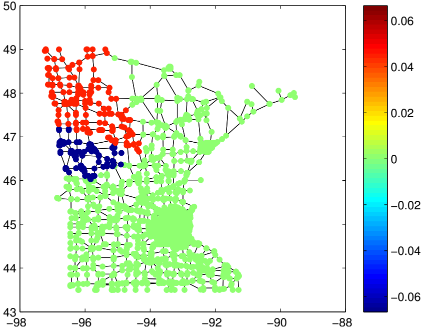
Level $j = 2$, Region $k = 1$, $l = 2$



GHWT on MN Road Network

- $j = 0$ is the coarsest level, $j = 16$ is the finest
- Inverse Euclidean weights, partitioned via the Fiedler vector of L_{rw}

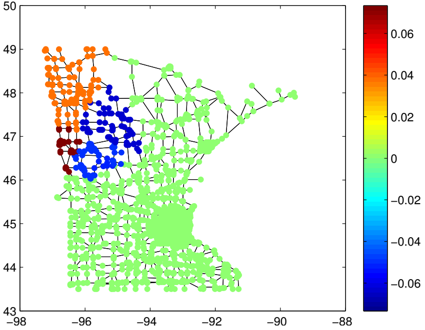
Level $j = 3$, Region $k = 0$, $l = 1$



GHWT on MN Road Network

- $j = 0$ is the coarsest level, $j = 16$ is the finest
- Inverse Euclidean weights, partitioned via the Fiedler vector of L_{rw}

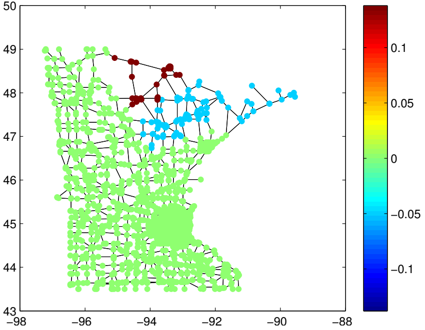
Level $j = 3$, Region $k = 0$, $l = 2$



GHWT on MN Road Network

- $j = 0$ is the coarsest level, $j = 16$ is the finest
- Inverse Euclidean weights, partitioned via the Fiedler vector of L_{rw}

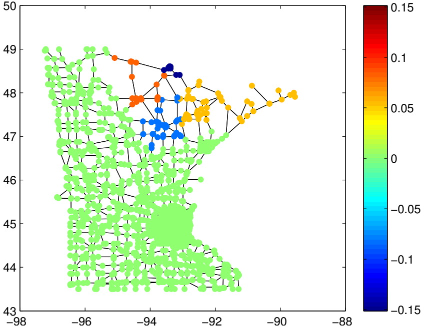
Level $j = 3$, Region $k = 2$, $l = 1$



GHWT on MN Road Network

- $j = 0$ is the coarsest level, $j = 16$ is the finest
- Inverse Euclidean weights, partitioned via the Fiedler vector of L_{rw}

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



Observations

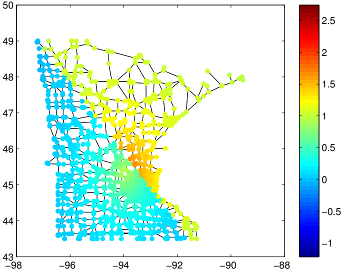
- When performed on an unweighted dyadic path graph (partitioned dyadically), the GHWT corresponds exactly to the Haar-Walsh wavelet packet transform
- The generalized Haar basis is a choosable basis from the fine-to-coarse dictionary
- Given a recursive partitioning with $O(\log N)$ levels, the computational cost of the GHWT is $\boxed{O(N \log N)}$

	N	$j_{\max}$	GHWT Run Time
MN Road Network	$2,636$	14	0.11 s
Facebook Dataset	$4,039$	26	0.62 s
Brain Mesh Dataset	$127,083$	20	4.29 s

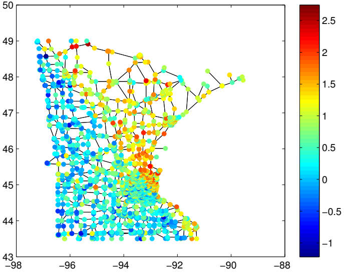
(Experiments performed using MATLAB on a personal laptop.)

Denoising Experiment

1. Added noise to the mutilated Gaussian on the MN road network (left) to yield a signal with an SNR of 5.00 dB (right)



(a) Original signal

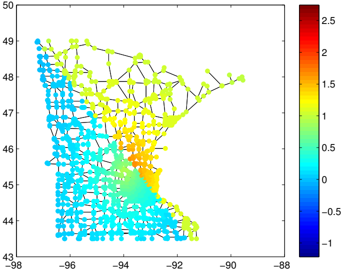


(b) Noisy signal

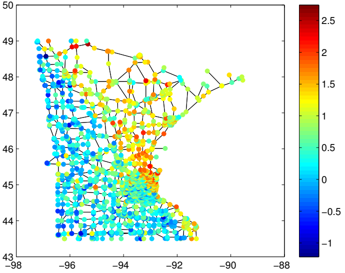
2. Expanded into 3 bases: Laplacian eigenvectors, the generalized Haar basis, and the GHWT coarse-to-fine level $j = 6$ basis
3. Denoised via soft-thresholding, using an elbow detection algorithm to determine the thresholds

Denoising Experiment

1. Added noise to the mutilated Gaussian on the MN road network (left) to yield a signal with an SNR of 5.00 dB (right)



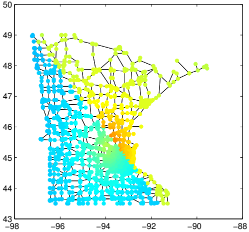
(a) Original signal



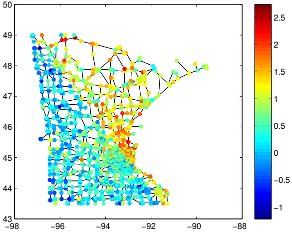
(b) Noisy signal

2. Expanded into 3 bases: Laplacian eigenvectors, the generalized Haar basis, and the GHWT coarse-to-fine level $j = 6$ basis
3. Denoised via soft-thresholding, using an elbow detection algorithm to determine the thresholds

Denoising Experiment

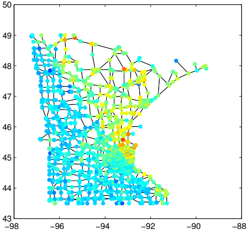


(a) Original signal

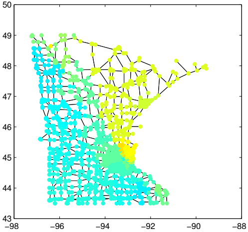


(b) Noisy signal

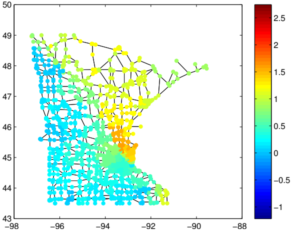
Signal	SNR
Noisy	5.00 dB
Lap. Eigs.	8.29 dB
Haar	10.98 dB
GHWT $j = 6$	13.38 dB



(c) Laplacian eigenvectors



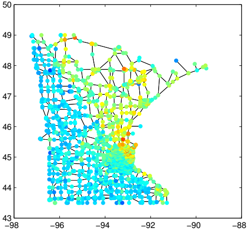
(d) Generalized Haar basis



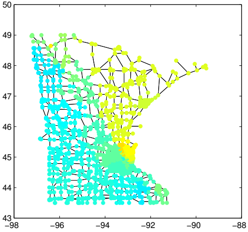
(e) GHWT Coarse-to-Fine Level $j = 6$

Denoising Experiment

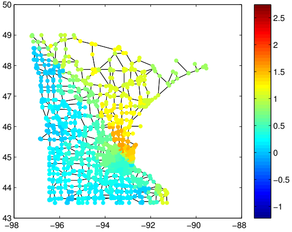
- GHWT level 6 has the best localization for this denoising task
  - Small enough to capture details, large enough to drown out noise
  - The generalized Haar basis is too localized and the Laplacian eigenvectors are too global
- Most retained coefficients for GHWT level 6 are **scaling coefficients**, not **Haar-like** (as in the generalized Haar basis)



(a) Laplacian eigenvectors



(b) Generalized Haar basis



(c) GHWT Coarse-to-Fine Level $j = 6$

Summary

- The GHWT is a *multiscale transform* that is a generalization of the Haar Transform and the Walsh-Hadamard Transform
- It produces an overcomplete dictionary of orthonormal bases from which we can choose a basis most suitable for the task at hand

Future Work

- Allow for partitions to have "soft" boundaries
- Investigate different graph partitioning methods
- Investigate usefulness for classification and function estimation

Summary

- The GHWT is a *multiscale transform* that is a generalization of the Haar Transform and the Walsh-Hadamard Transform
- It produces an overcomplete dictionary of orthonormal bases from which we can choose a basis most suitable for the task at hand

Future Work

- Allow for partitions to have "soft" boundaries
- Investigate different graph partitioning methods
- Investigate usefulness for classification and function estimation

Acknowledgements & References

Funding by

- NSF VIGRE DMS-0636297
- NDSEG Fellowship, 32 CFR 168a
- ONR grant N00014-12-1-0177

References

1. J. Irion, N. Saito: "The Generalized Haar-Walsh Transform," *Proceedings of 2014 IEEE Workshop on Statistical Signal Processing*, pp. 488–491, 2014.
2. J. Irion, N. Saito: "Hierarchical Graph Laplacian Eigen Transforms," *Japan SIAM Letters*, vol. 6, pp. 21–24, 2014.
3. R. Coifman, M. Wickerhauser: "Entropy-based algorithms for best basis selection," *IEEE Transactions on Information Theory*, vol. 38, no. 2, pp. 713–718, 1992.