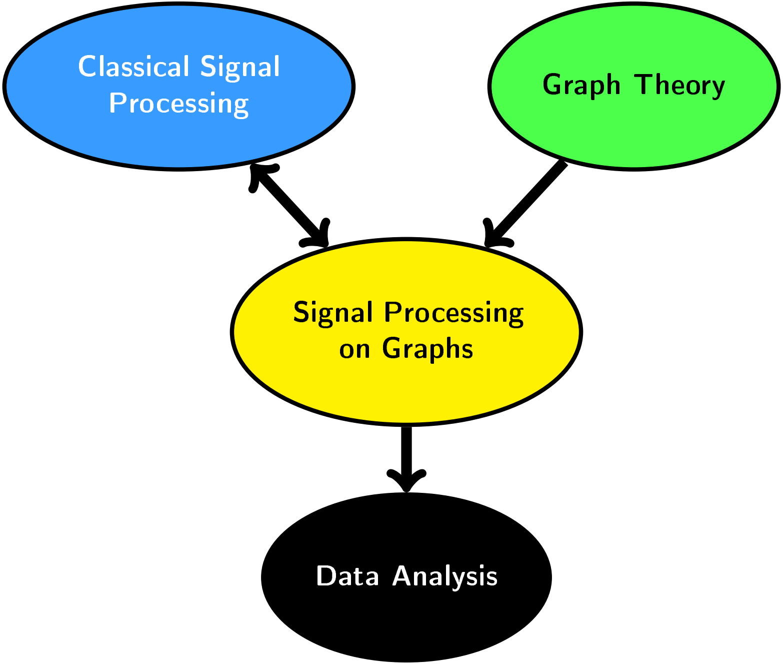
Multiscale Transforms for Signals on Graphs: Methods and Applications

Jeff Irion, Ph.D.

Adviser: Naoki Saito

Department of Mathematics
University of California, Davis

Bosch
April 7, 2016


Classical Signal Processing
Graph Theory
Signal Processing on Graphs
Data Analysis

What is a graph?

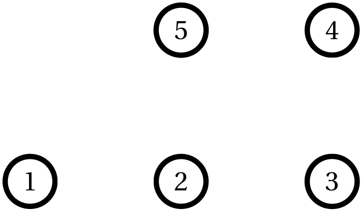
A graph G has:

- Vertices $V = V(G) = \{v_1, \ldots, v_N\}$
- Edges $E = E(G) = \{e_1, \ldots, e_{N'}\}$
- Edge weights, which we organize in a weight matrix $W = W(G) \in \mathbb{R}^{N \times N}$
    - w_{ij} denotes the edge weight between vertices i and j

What is a graph?

A graph G has:

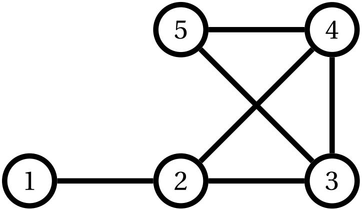
- Vertices $V = V(G) = \{v_1, \ldots, v_N\}$
- Edges $E = E(G) = \{e_1, \ldots, e_{N'}\}$
- Edge weights, which we organize in a weight matrix $W = W(G) \in \mathbb{R}^{N \times N}$
  - w_{ij} denotes the edge weight between vertices i and j



What is a graph?

A graph G has:

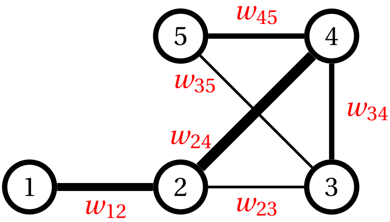
- Vertices $V = V(G) = \{v_1, \ldots, v_N\}$
- Edges $E = E(G) = \{e_1, \ldots, e_{N'}\}$
- Edge weights, which we organize in a weight matrix $W = W(G) \in \mathbb{R}^{N \times N}$
  - w_{ij} denotes the edge weight between vertices i and j



What is a graph?

A graph G has:

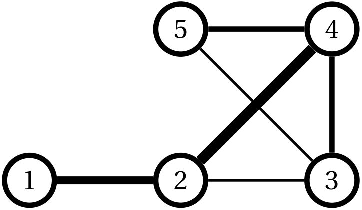
- Vertices $V = V(G) = \{v_1, \ldots, v_N\}$
- Edges $E = E(G) = \{e_1, \ldots, e_{N'}\}$
- Edge weights, which we organize in a weight matrix $W = W(G) \in \mathbb{R}^{N \times N}$
  - w_{ij} denotes the edge weight between vertices i and j



What is a signal on a graph?

A signal on a graph is a vector $\boldsymbol{f} \in \mathbb{R}^N$ whose values correspond to the vertices of the graph G.

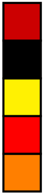
$$\boldsymbol{f} = \begin{array}{|c|} \hline 1.2 \\ \hline 0.7 \\ \hline 5.9 \\ \hline 2.2 \\ \hline 4.6 \\ \hline \end{array}$$

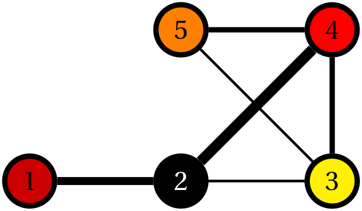


What is a signal on a graph?

A signal on a graph is a vector $\boldsymbol{f} \in \mathbb{R}^N$ whose values correspond to the vertices of the graph G.

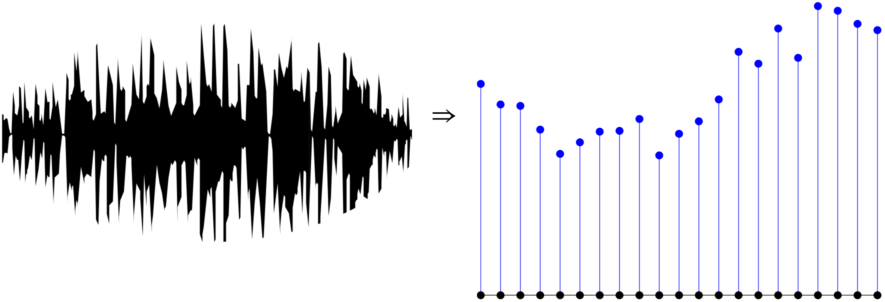
$$\boldsymbol{f} = \begin{bmatrix} 1.2 \\ 0.7 \\ 5.9 \\ 2.2 \\ 4.6 \end{bmatrix} =$$





Examples of Graph Signals (1/3)

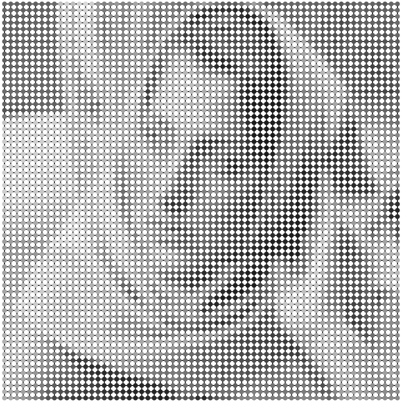
An audio signal:



Examples of Graph Signals (2/3)

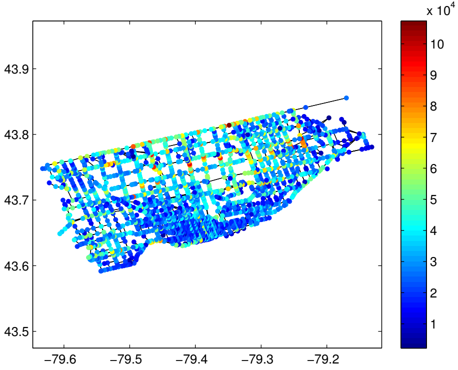
An image:

$\Rightarrow$



Examples of Graph Signals (3/3)

Traffic volume (Toronto):



Our Assumptions

We assume that the graph is

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

Motivation

Aims & Objectives:

1. Develop overcomplete multiscale transforms for signals on graphs
2. Develop a corresponding best-basis search algorithm
3. Investigate usefulness for approximation and data analysis

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 overcomplete multiscale transforms for signals on graphs
2. Develop a corresponding best-basis search algorithm
3. Investigate usefulness for approximation and data analysis

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!**

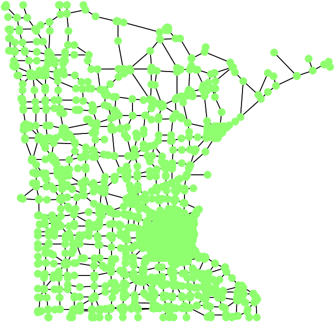
Recursive Partitioning

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

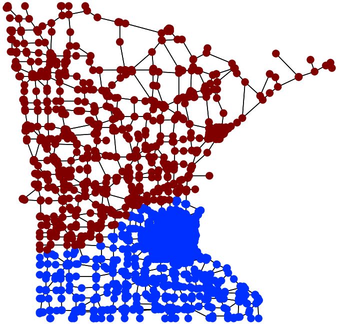
Our transform requires as input a recursive partitioning of the graph.



- 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

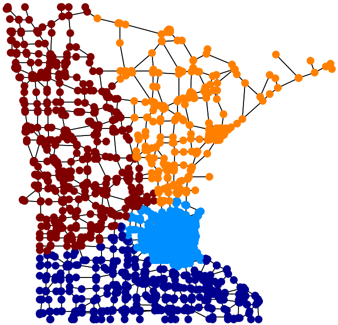
Our transform requires as input a recursive partitioning of the graph.



- 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

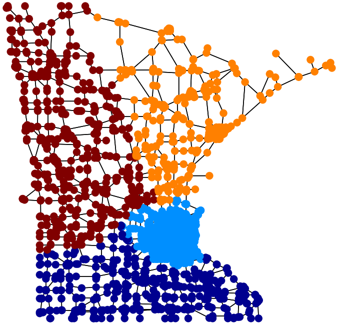
Our transform requires as input a recursive partitioning of the graph.



- 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.

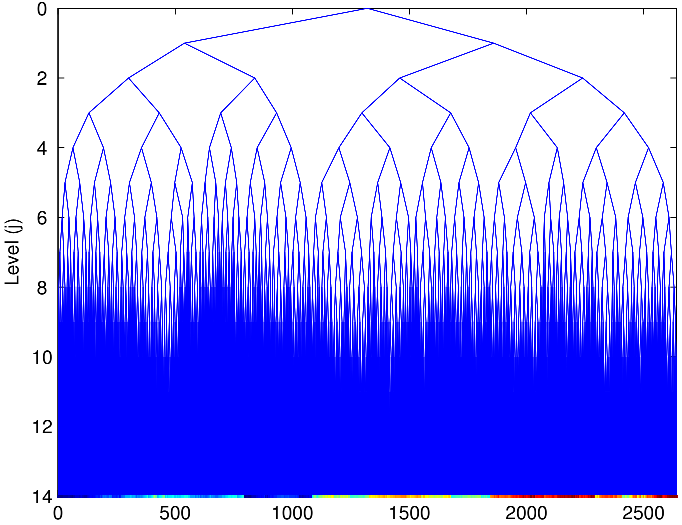
Recursive Partitioning

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.

Recursive Partitioning

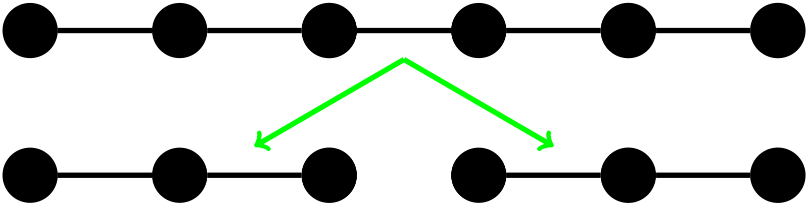


GHWT

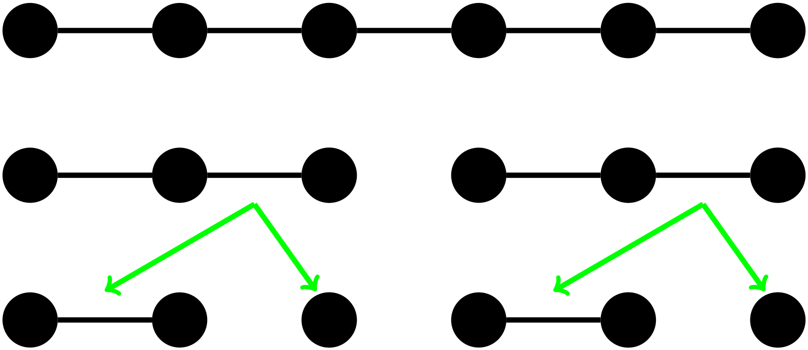
Given a recursive partitioning of the graph, the GHWT yields an ***overcomplete dictionary of orthonormal bases*** for signals on the graph.

GHWT on P_6

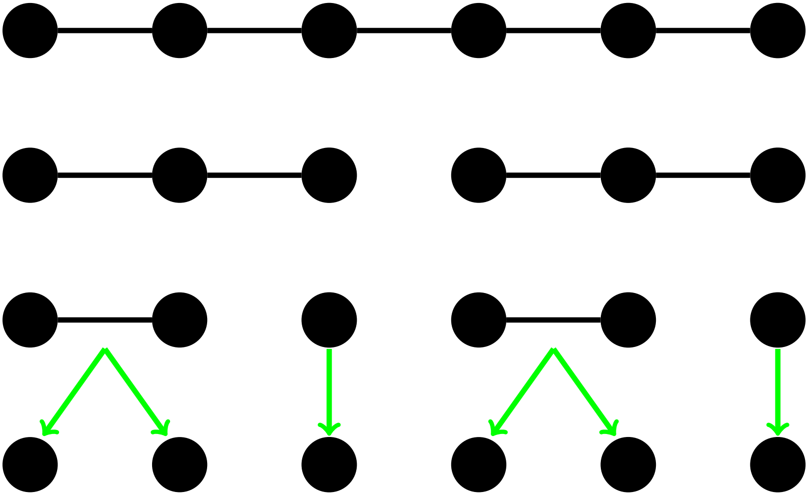
GHWT on P_6



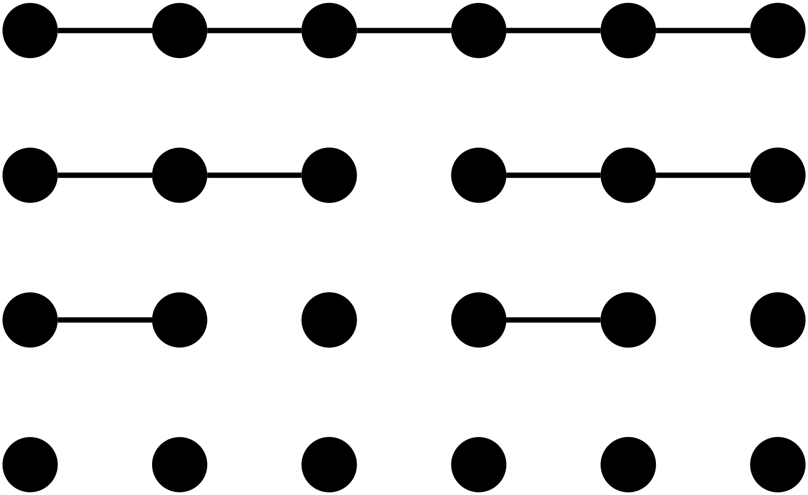
GHWT on P_6



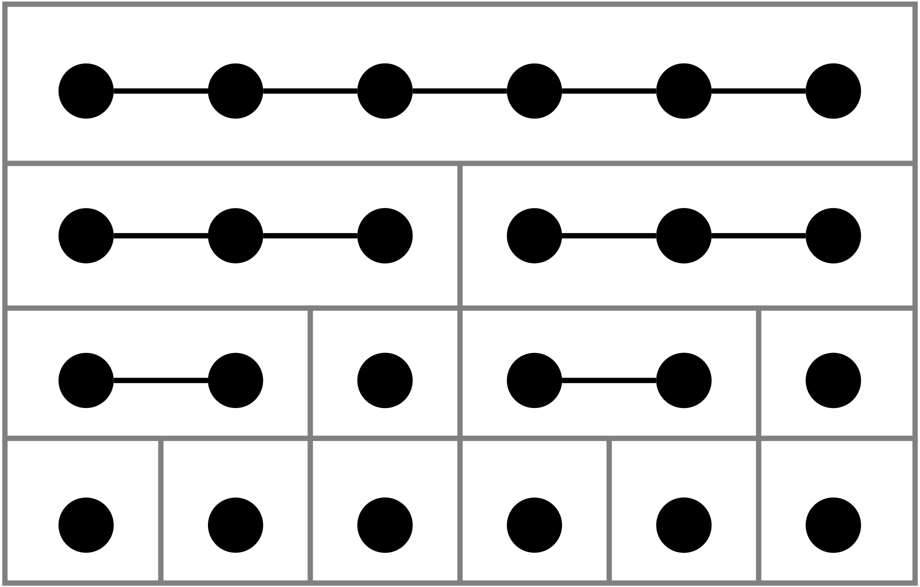
GHWT on P_6



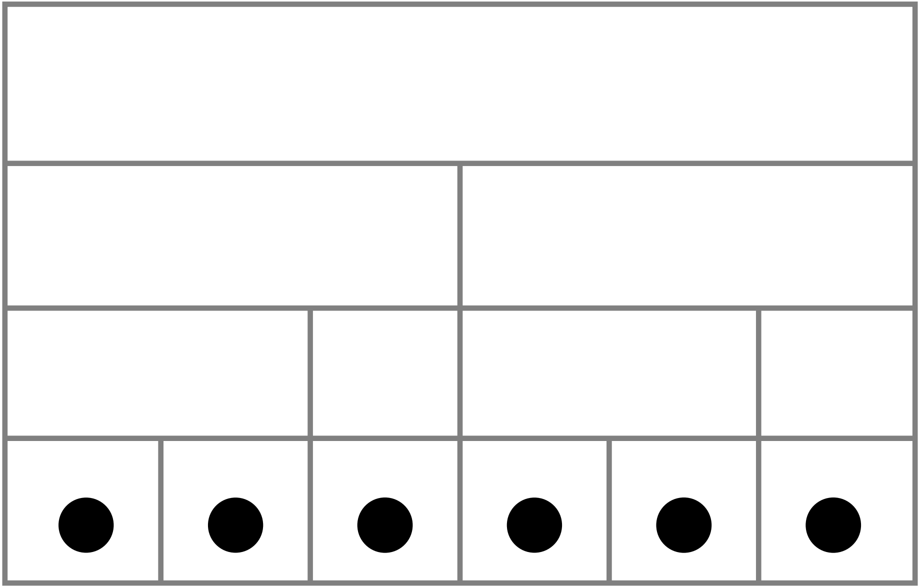
GHWT on P_6



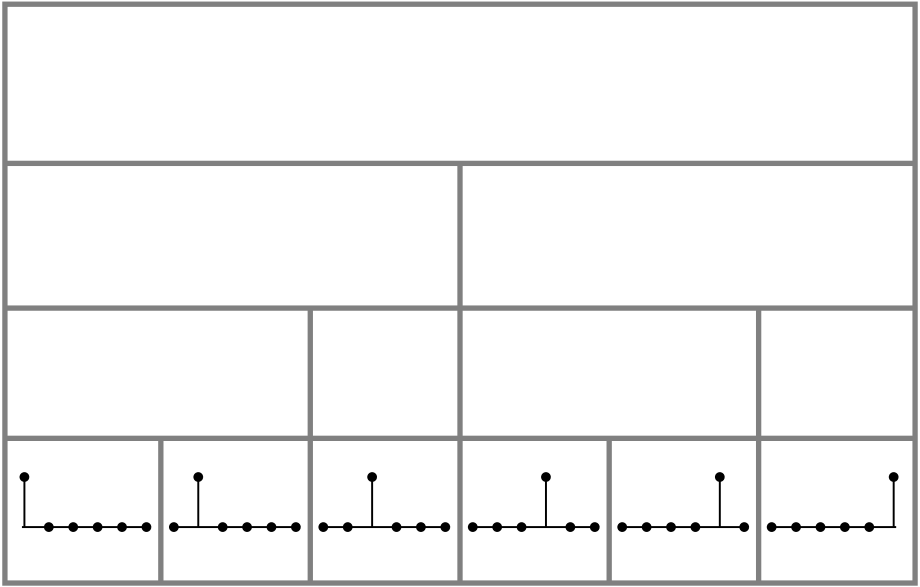
GHWT on P_6



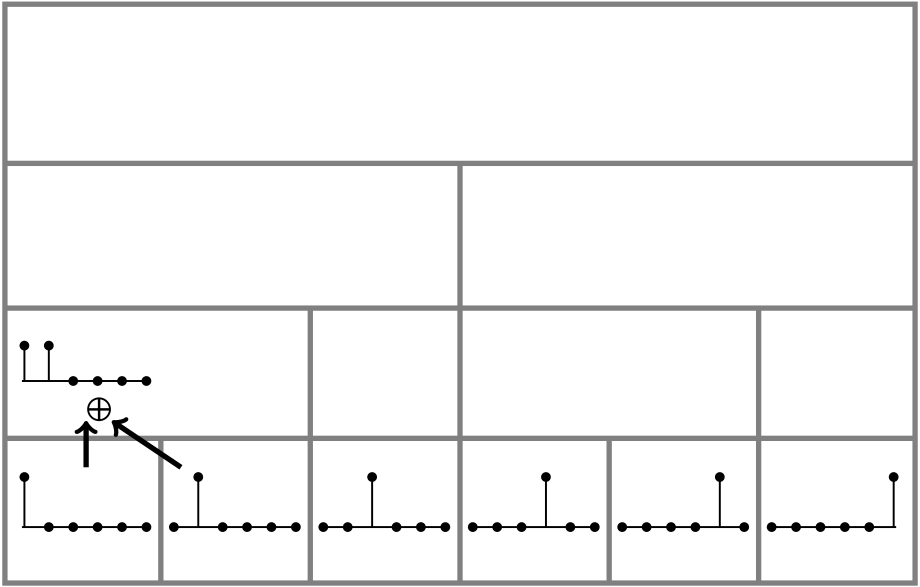
GHWT on P_6



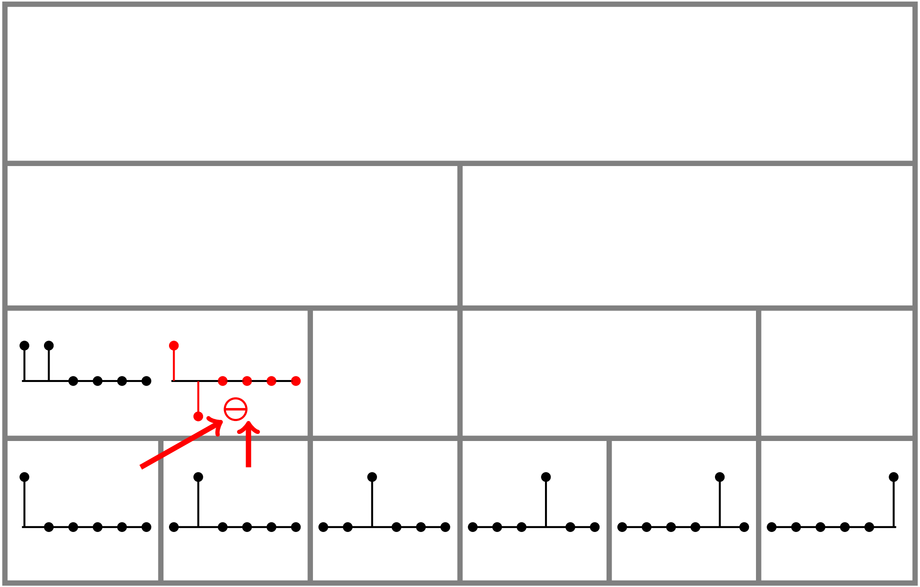
GHWT on P_6



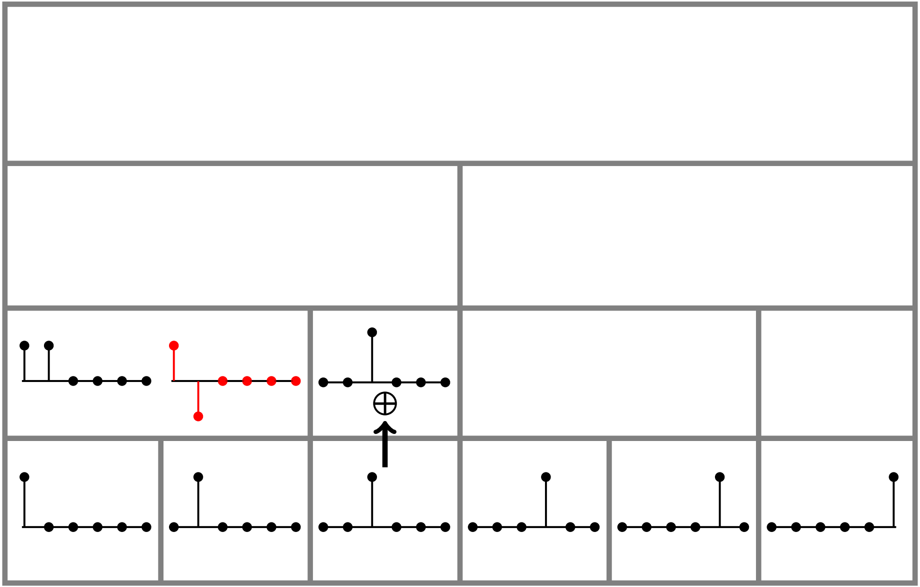
GHWT on P_6



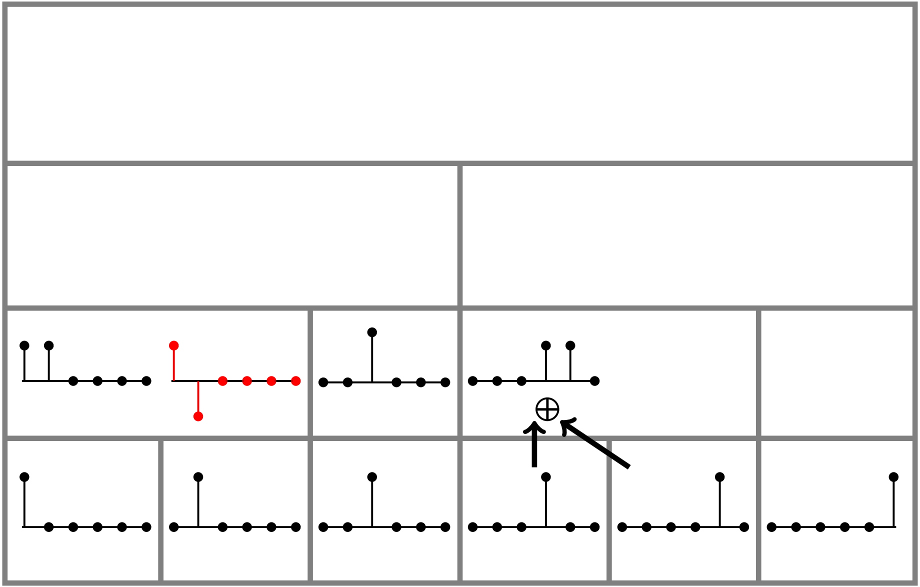
GHWT on P_6



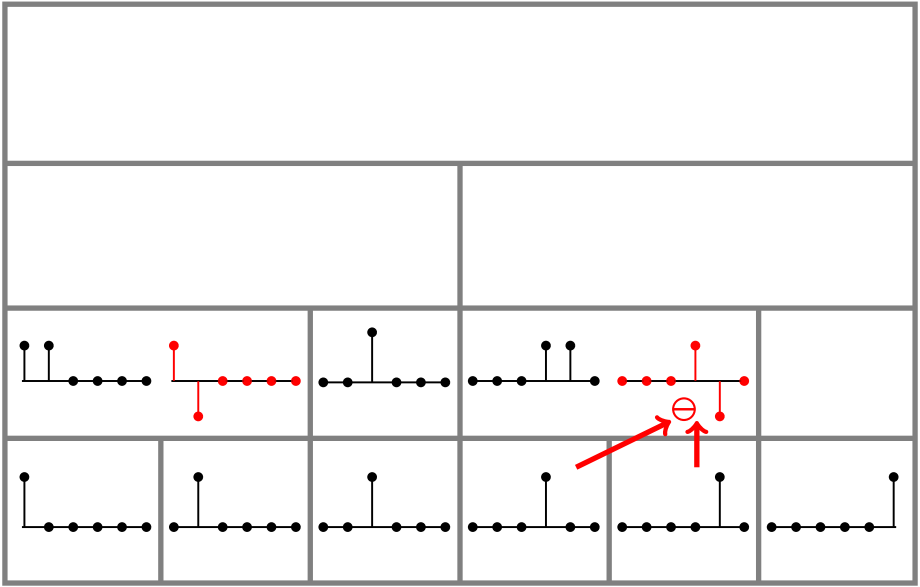
GHWT on P_6



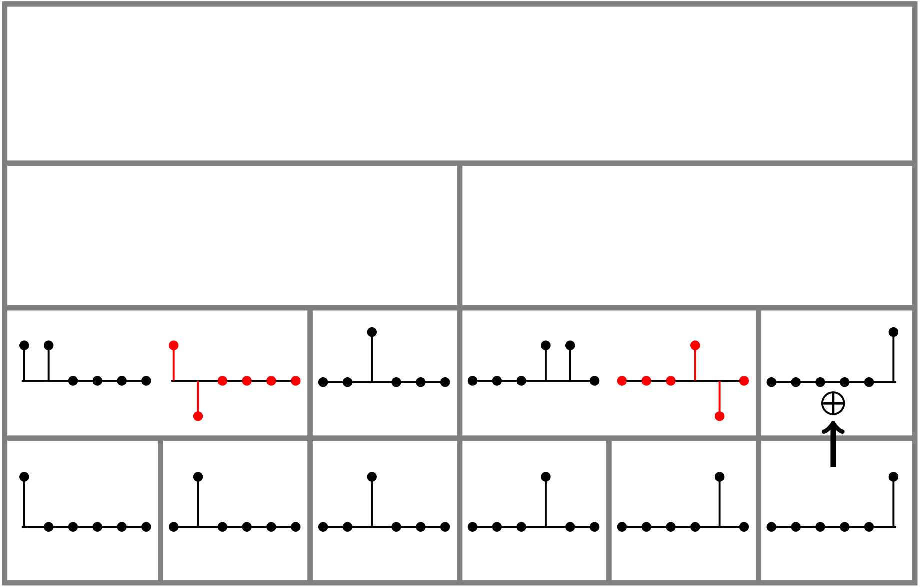
GHWT on P_6



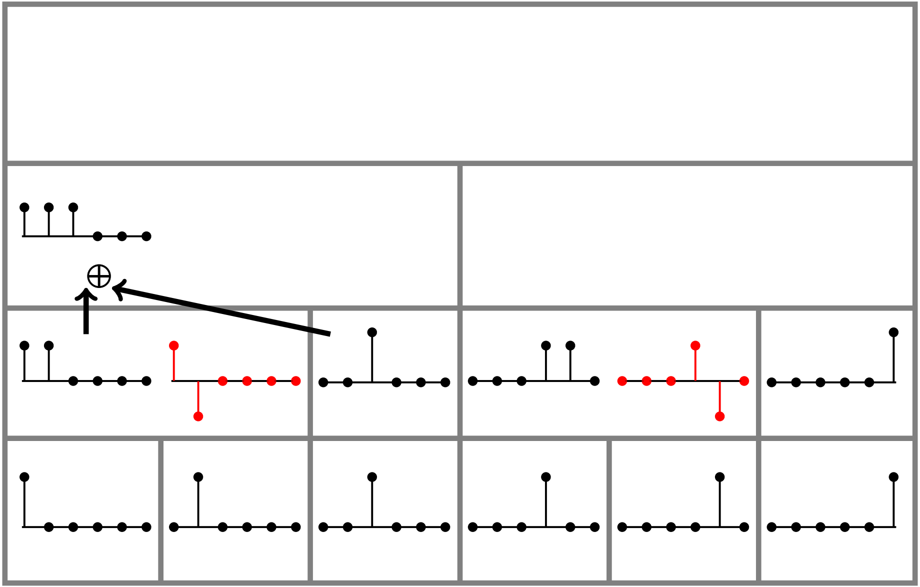
GHWT on P_6



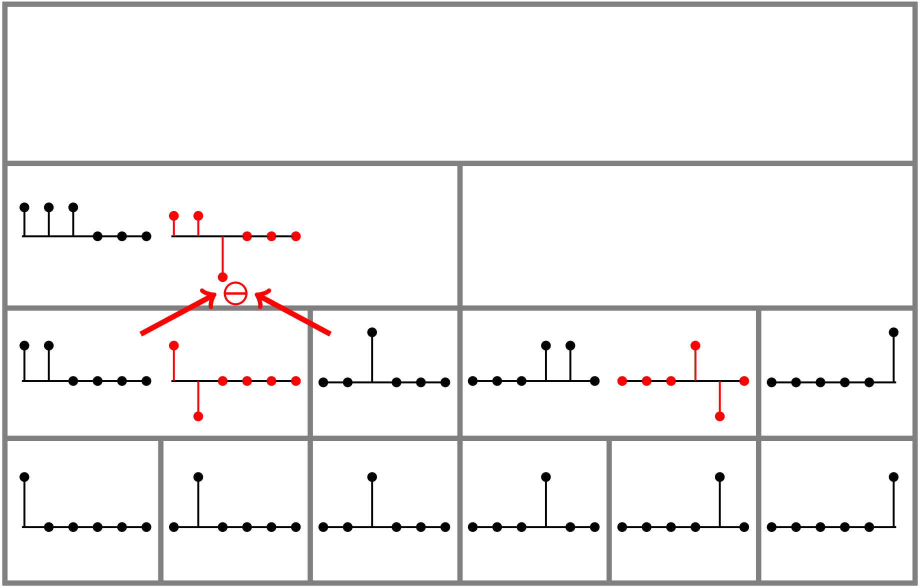
GHWT on P_6



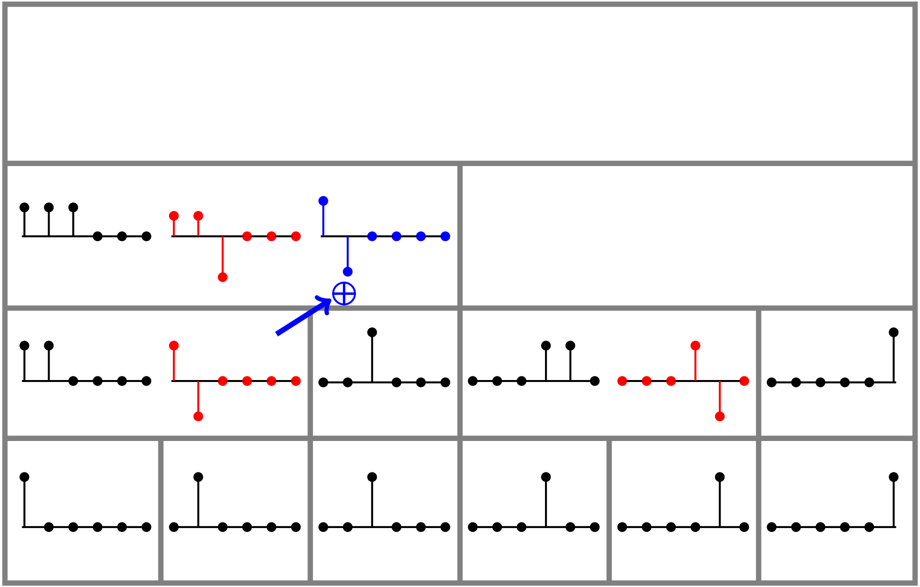
GHWT on P_6



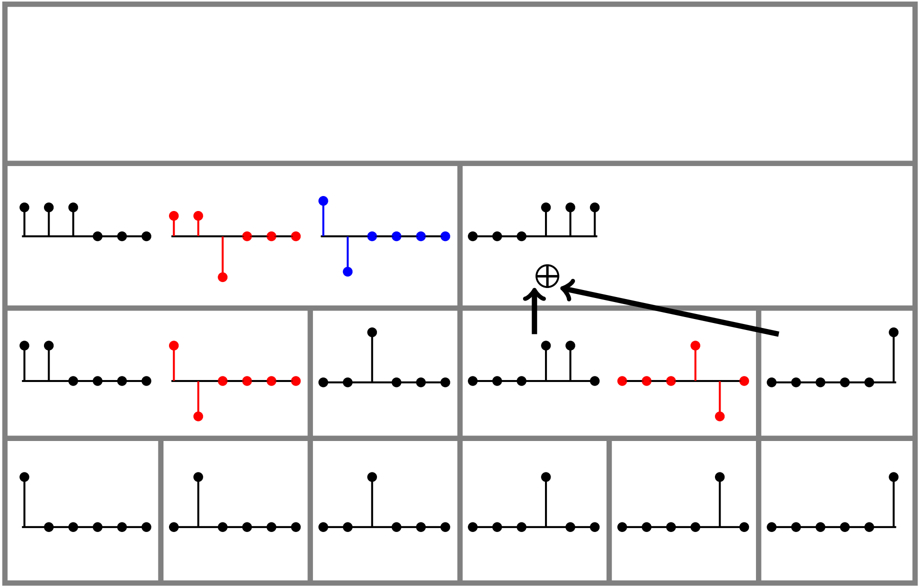
GHWT on P_6



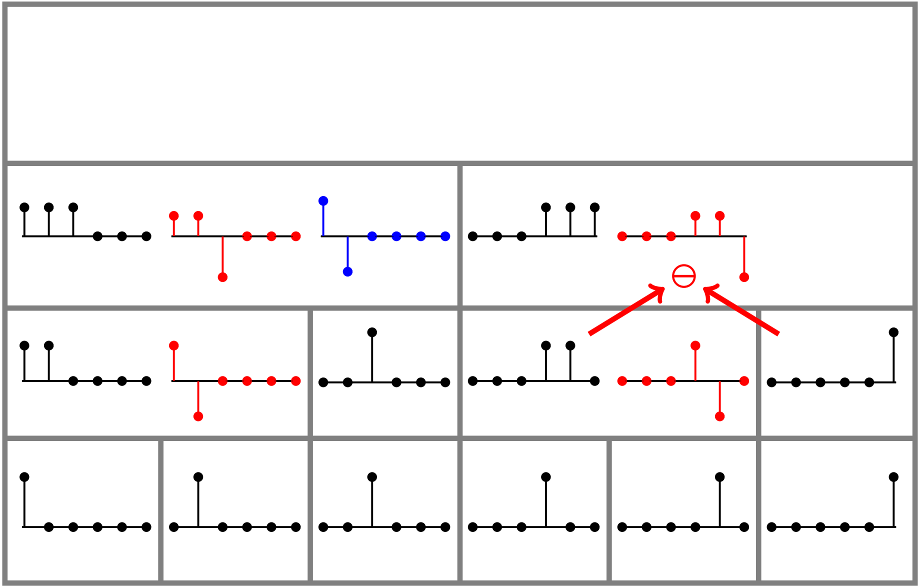
GHWT on P_6



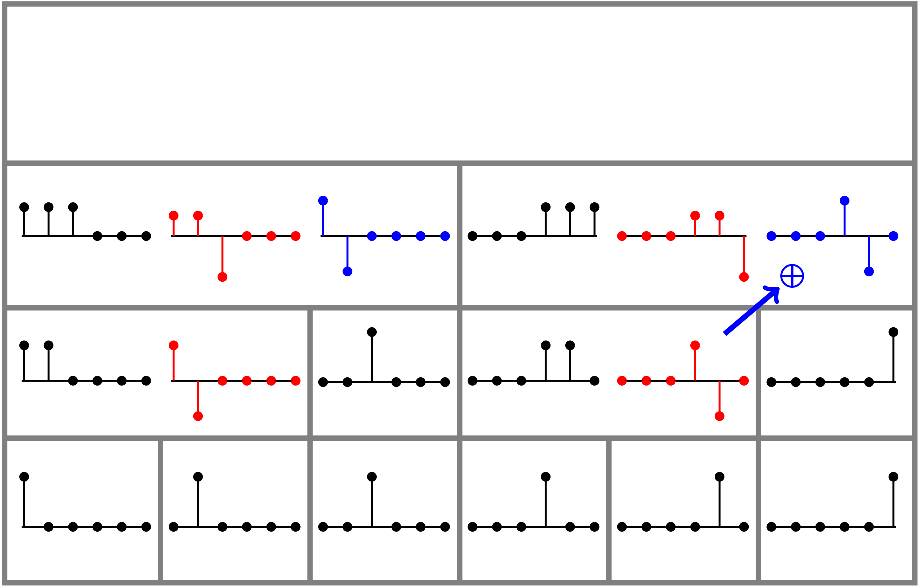
GHWT on P_6



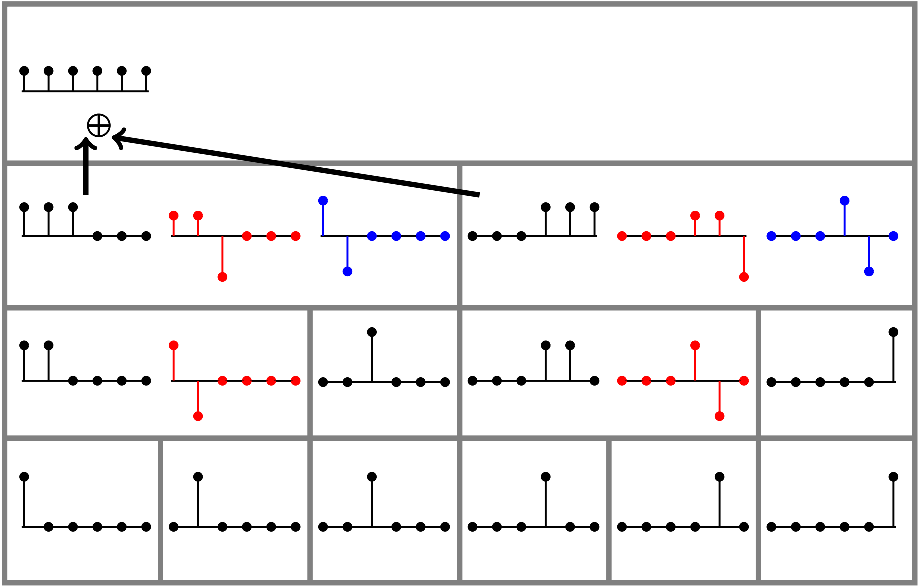
GHWT on P_6



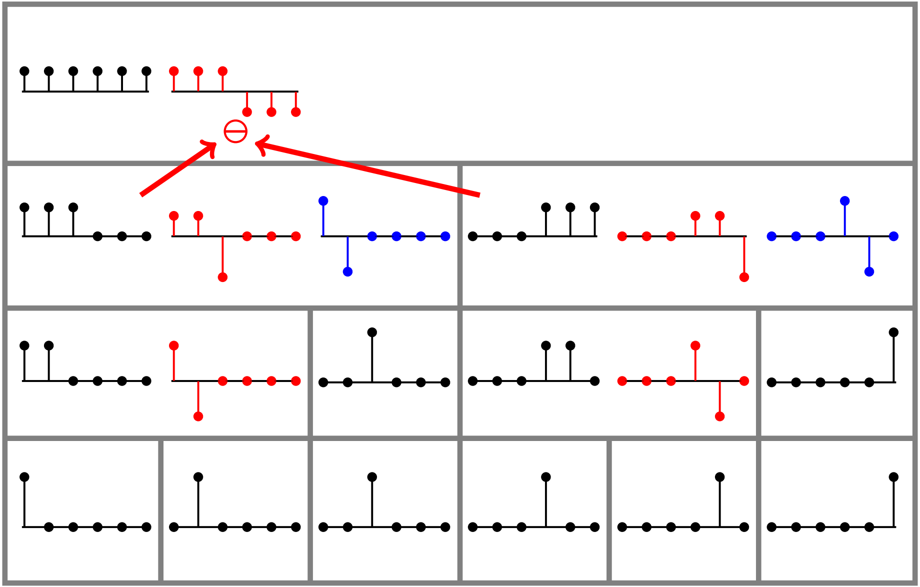
GHWT on P_6



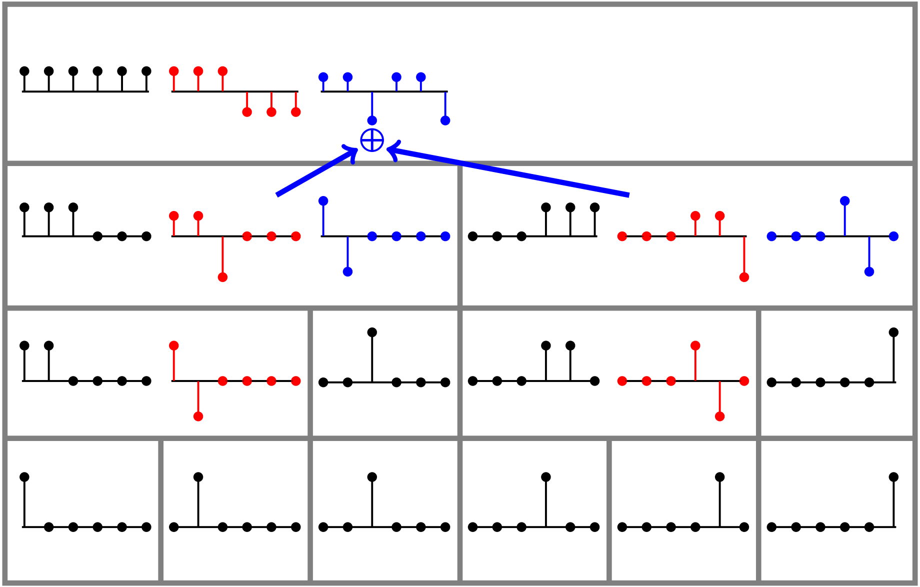
GHWT on P_6



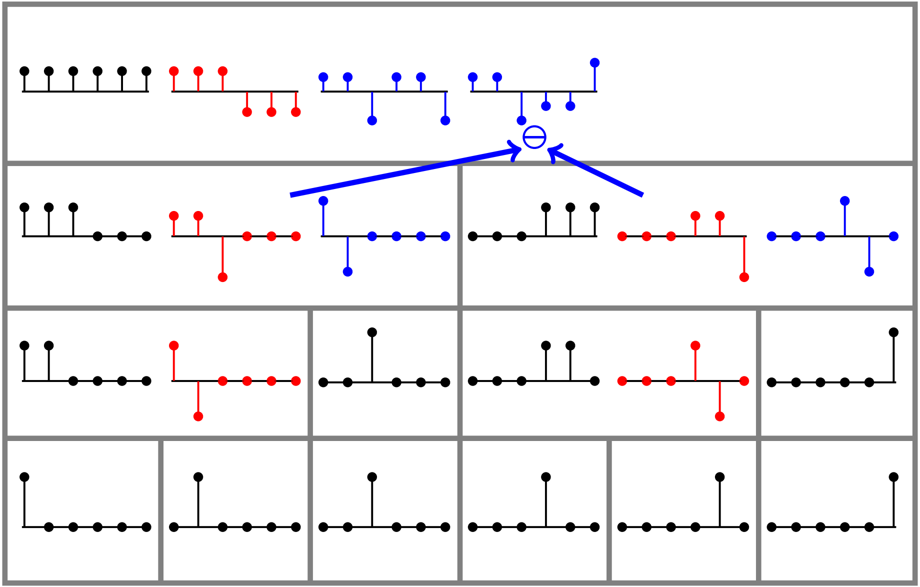
GHWT on P_6



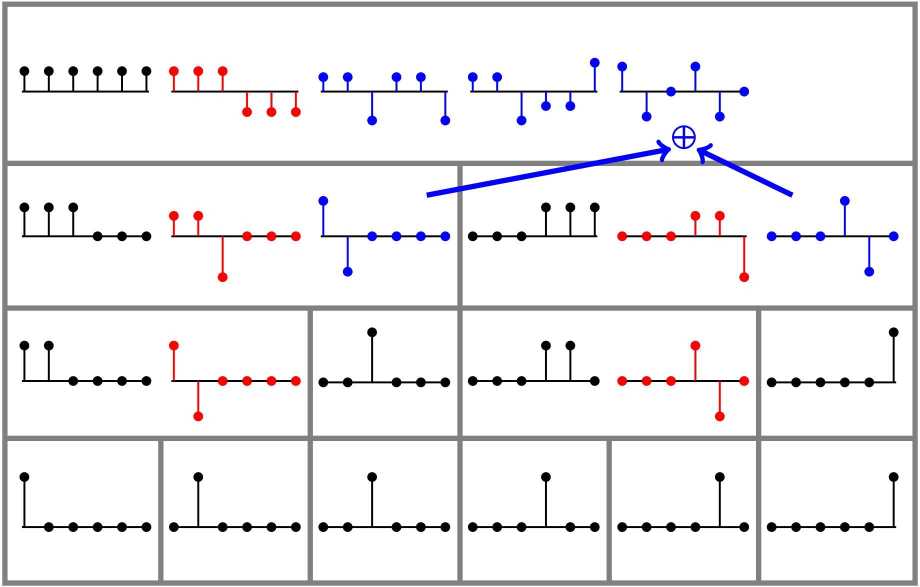
GHWT on P_6



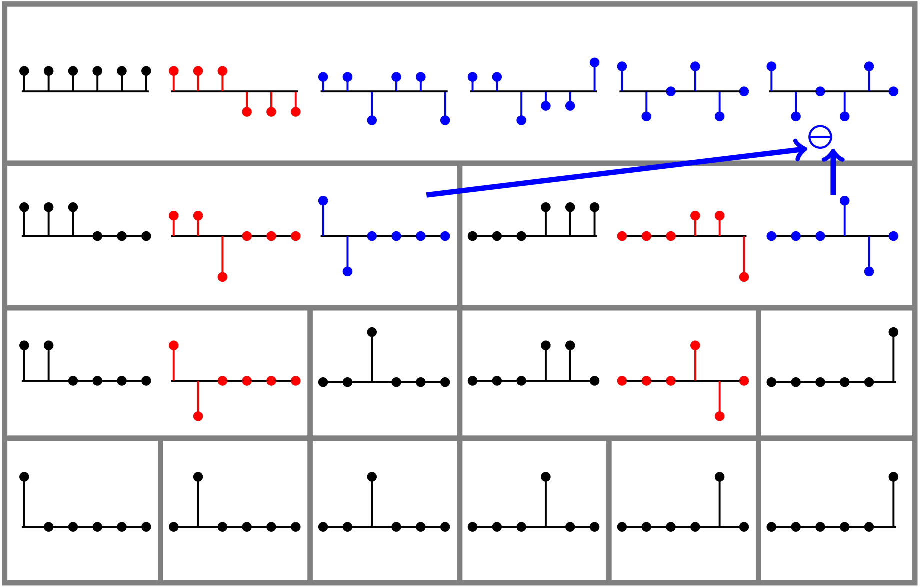
GHWT on P_6



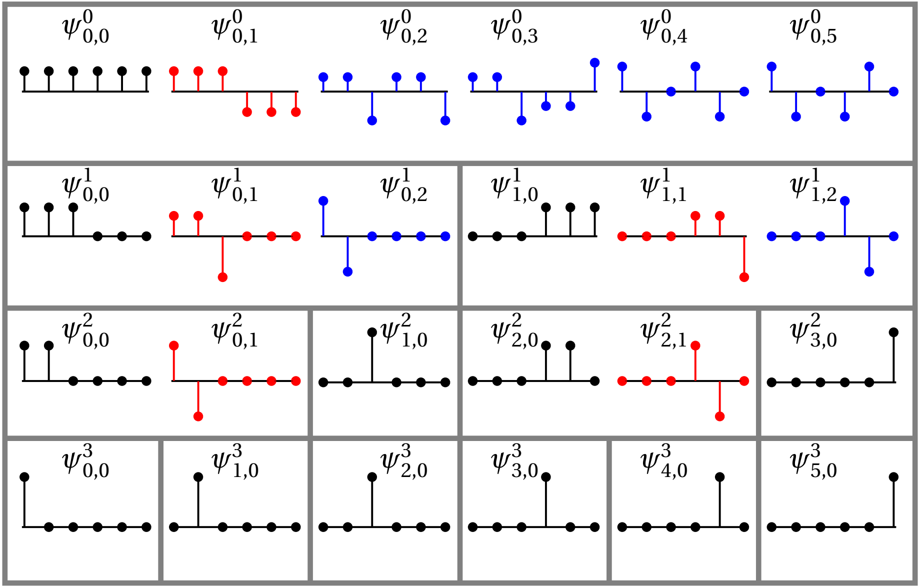
GHWT on P_6



GHWT on P_6

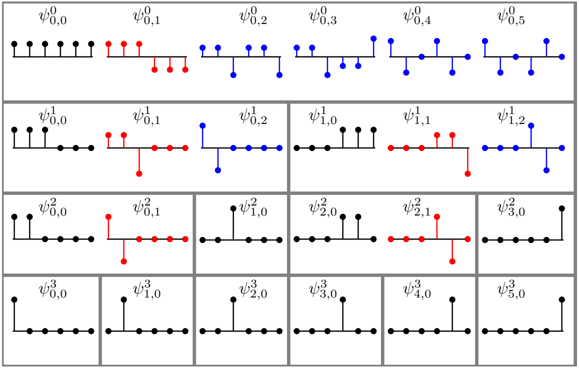


GHWT on P_6


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

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 on P_6 – Fine-to-Coarse Dictionary

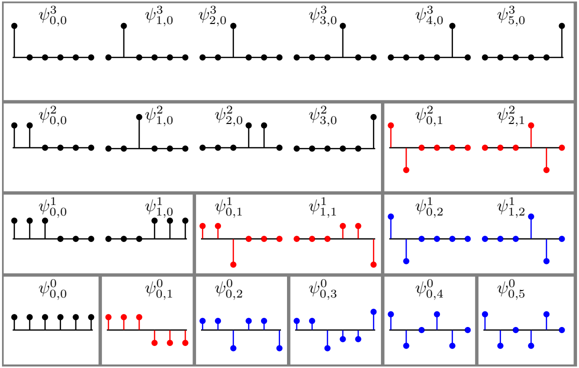
Note that the basis vectors with tag ℓ on level j were used to generate those with tags 2ℓ and $2\ell+1$ on level $j-1$.

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

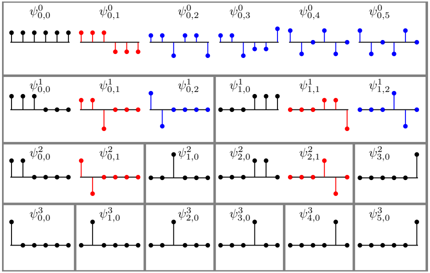
GHWT on P_6 – Fine-to-Coarse Dictionary

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

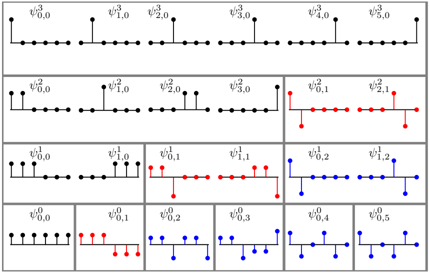
Using this fact, we can reorganize the basis vectors by their tags to yield the *fine-to-coarse dictionary*:



GHWT on P_6



(a) Coarse-to-fine dictionary



(b) Fine-to-coarse dictionary

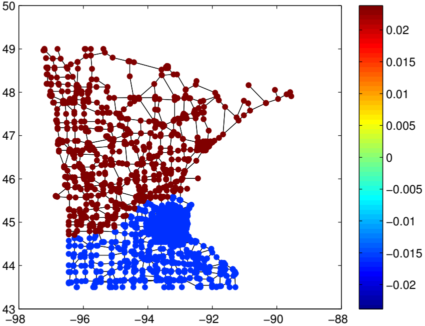
GHWT on MN Road Network

- $j = 0$ 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}

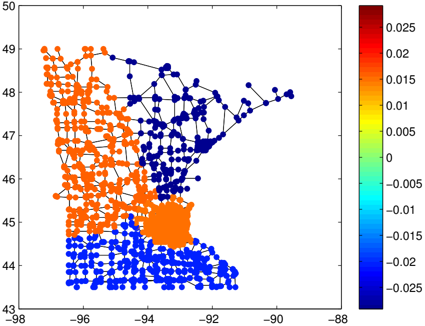
Level $j = 0$, Region $k = 0$, $l = 1$



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}

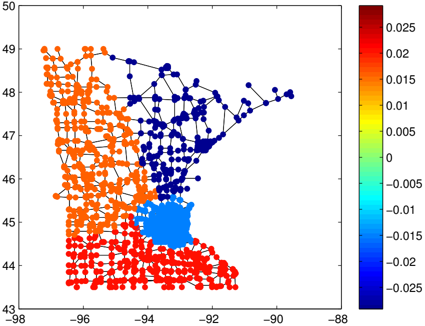
Level $j = 0$, Region $k = 0$, $l = 2$



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}

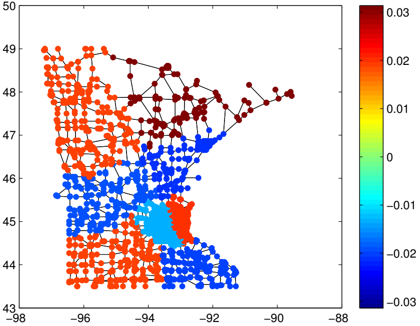
Level $j = 0$, Region $k = 0$, $l = 3$



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}

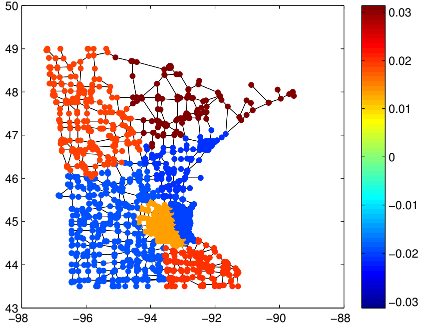
Level $j = 0$, Region $k = 0$, $l = 4$



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}

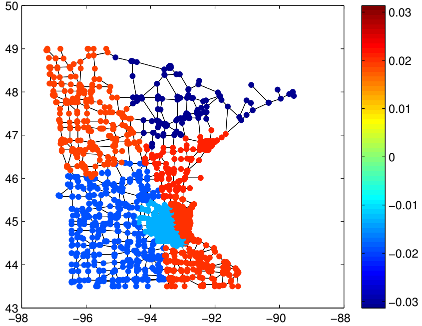
Level $j = 0$, Region $k = 0$, $l = 5$



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}

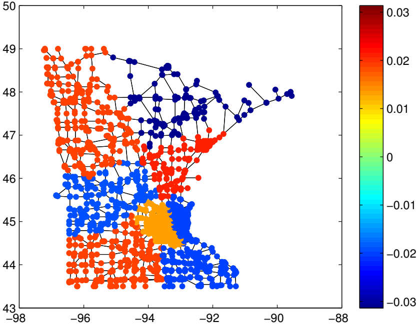
Level $j = 0$, Region $k = 0$, $l = 6$



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}

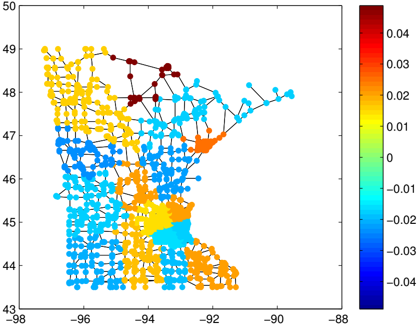
Level $j = 0$, Region $k = 0$, $l = 7$



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}

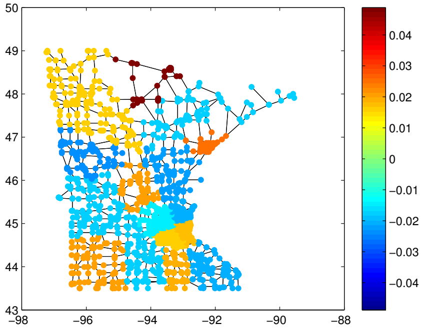
Level $j = 0$, Region $k = 0$, $l = 8$



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}

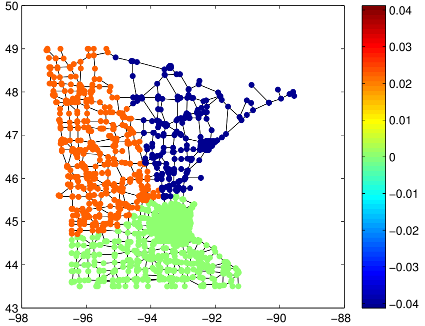
Level $j = 0$, Region $k = 0$, $l = 9$



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}

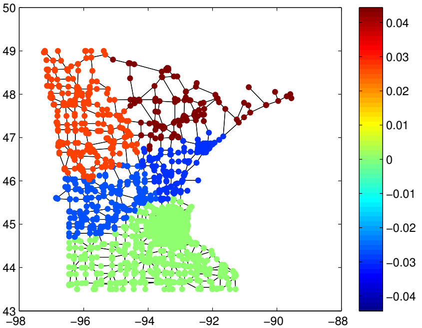
Level $j = 1$, Region $k = 0$, $l = 1$



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}

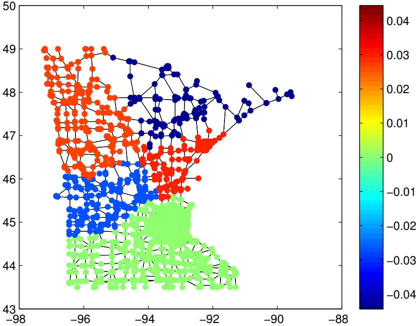
Level $j = 1$, Region $k = 0$, $l = 2$



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}

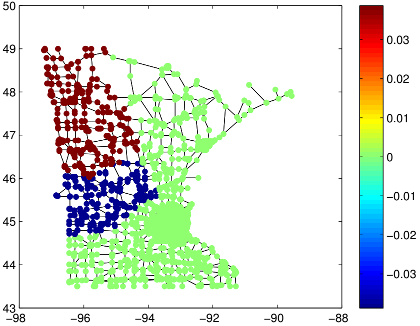
Level $j = 1$, Region $k = 0$, $l = 3$



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}

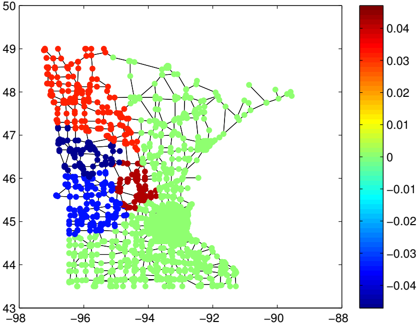
Level $j = 2$, Region $k = 0$, $l = 1$



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}

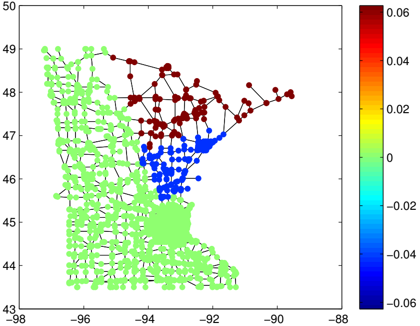
Level $j = 2$, Region $k = 0$, $l = 2$



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}

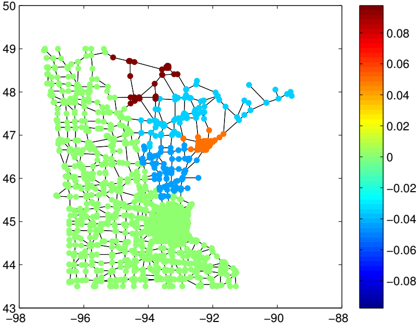
Level $j = 2$, Region $k = 1$, $l = 1$



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}

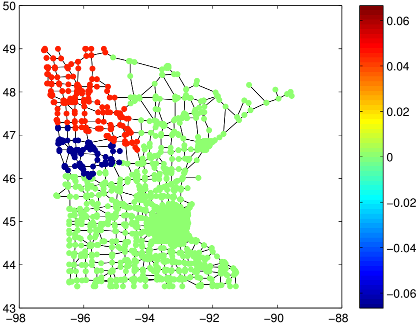
Level $j = 2$, Region $k = 1$, $l = 2$



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}

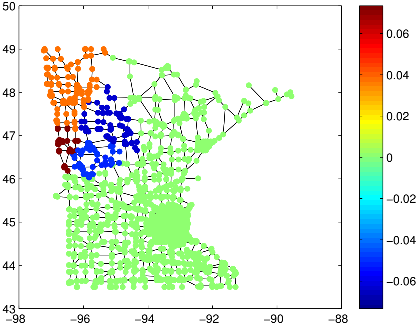
Level $j = 3$, Region $k = 0$, $l = 1$



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}

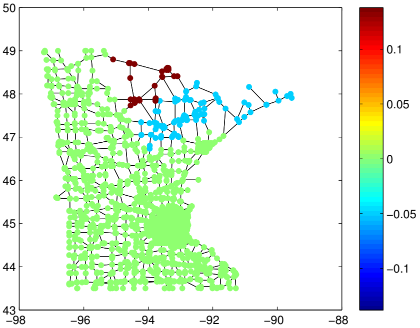
Level $j = 3$, Region $k = 0$, $l = 2$



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}

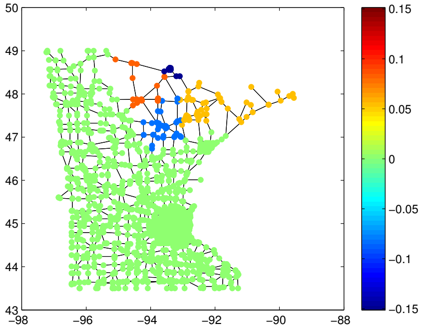
Level $j = 3$, Region $k = 2$, $l = 1$



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.)

1. Background

2. Motivation

3. Overcomplete Multiscale Transforms
   - Recursive Partitioning
   - GHWT
   - Best Basis Algorithm

4. Matrix Data Analysis

5. Conclusion

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

$$\mathscr{J}(\boldsymbol{x}) = \|\boldsymbol{x}\|_p := \left(\sum_{i=1}^{N} |x_i|^p \right)^{1/p} \qquad 1 \le p < 2$$

Best Basis Algorithm

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

Best Basis Algorithm

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

Best Basis Algorithm

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

Best Basis Algorithm

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

Best Basis Algorithm

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

Best Basis Algorithm

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

Best Basis Algorithm

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

Best Basis Algorithm

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

Best Basis Algorithm

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

Best Basis Algorithm

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

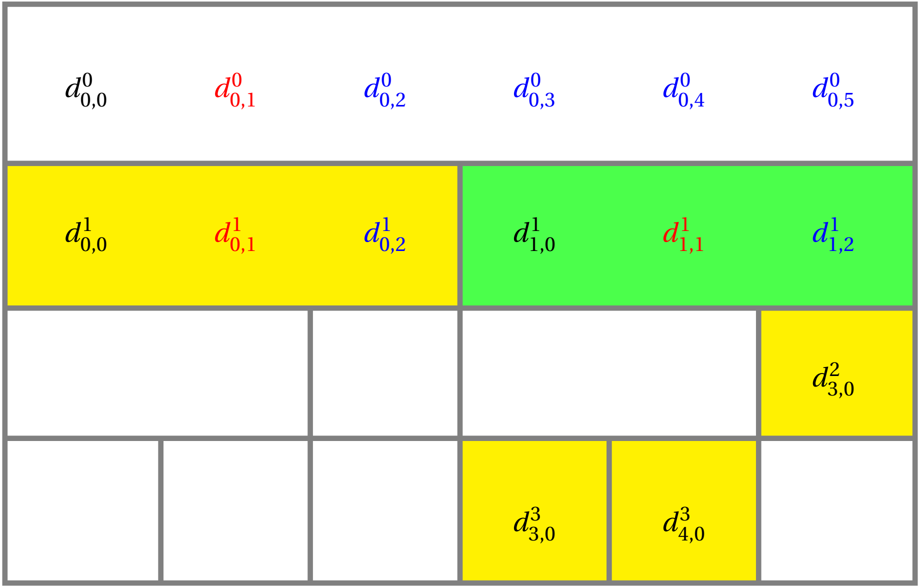
Best Basis Algorithm

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

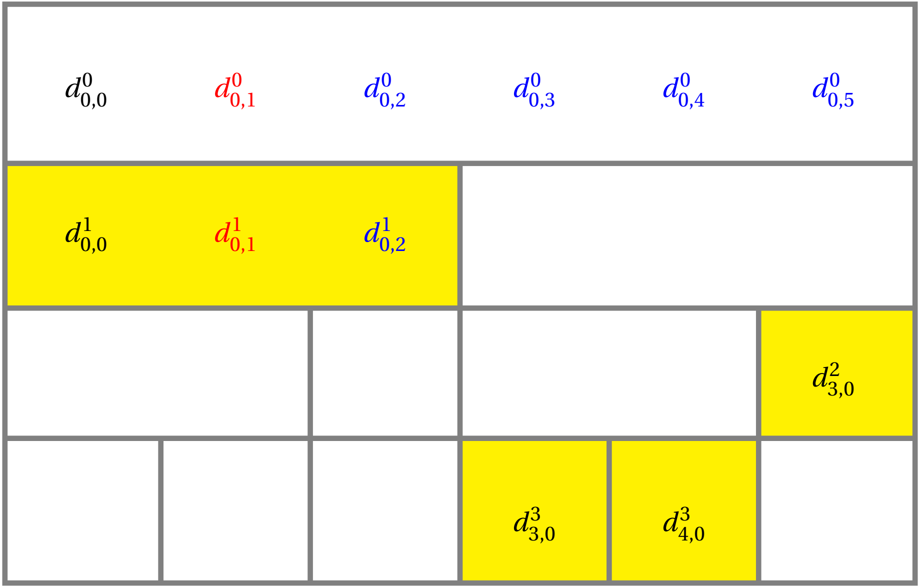
Best Basis Algorithm

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

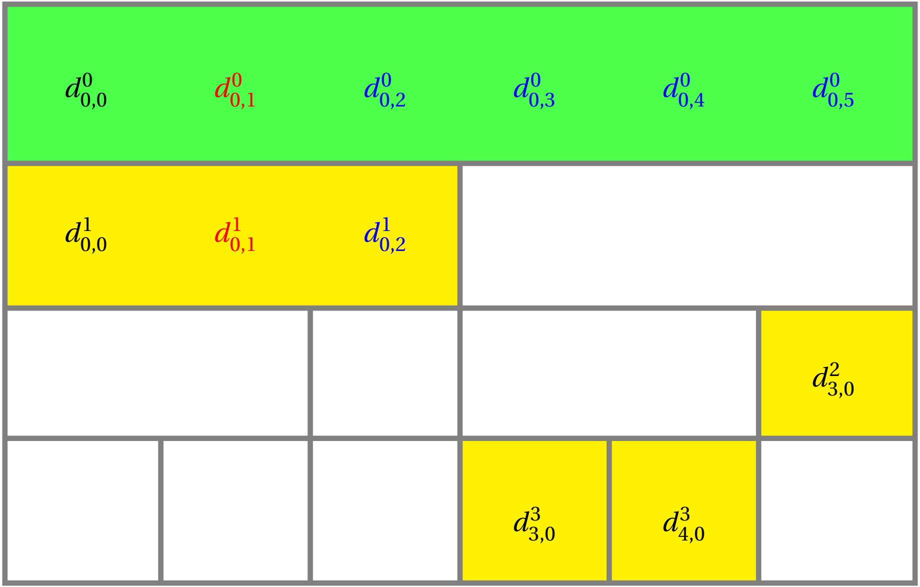
Best Basis Algorithm



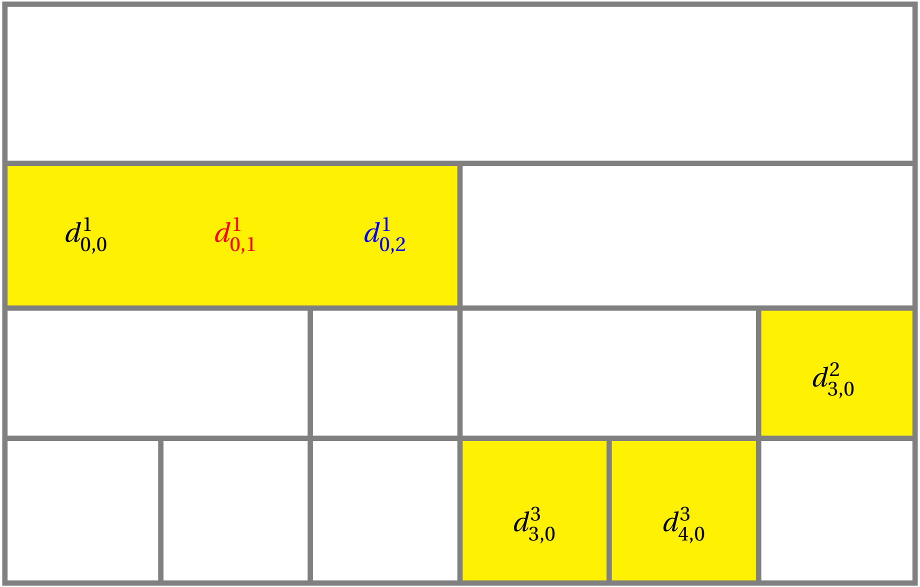
Best Basis Algorithm



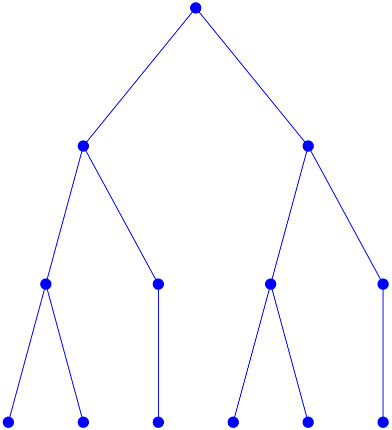
Best Basis Algorithm



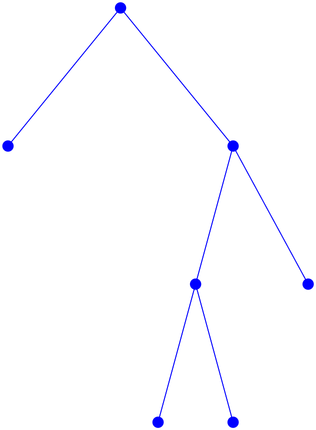
Best Basis Algorithm



Comparison to Decision Trees



(a) Full set of coefficients arranged as a tree.



(b) The "pruned" tree of best basis coefficients.

Best Basis Algorithm

Proposition

Suppose that $\mathscr{J}$ is a cost functional such that for all sequences $\{x_i\}$ and $\{y_i\}$ and integers $\alpha < \beta < \gamma$,

$$\begin{aligned} \textit{if} \quad & \mathscr{J}\left(\{x_i\}_{i\in[\alpha,\beta)}\right) \le \mathscr{J}\left(\{y_i\}_{i\in[\alpha,\beta)}\right) \\ \textit{and} \quad & \mathscr{J}\left(\{x_i\}_{i\in[\beta,\gamma)}\right) \le \mathscr{J}\left(\{y_i\}_{i\in[\beta,\gamma)}\right), \\ \textit{then} \quad & \mathscr{J}\left(\{x_i\}_{i\in[\alpha,\gamma)}\right) \le \mathscr{J}\left(\{y_i\}_{i\in[\alpha,\gamma)}\right). \end{aligned}$$

Given a signal $\boldsymbol{f}$ on a graph G and a hierarchical tree for the graph, the set of expansion coefficients returned by the best basis algorithm is the set that ***minimizes $\mathscr{J}$ over all choosable sets of coefficients*** *in the dictionary (or dictionaries) considered.*

The Minnesota road network ($N = 2640$) has **over 10^{453} choosable bases!**

Best Basis Algorithm

Proposition

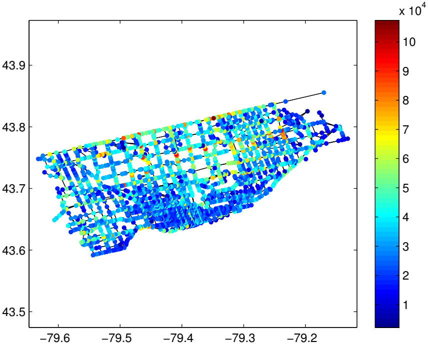
Suppose that $\mathscr{J}$ is a cost functional such that for all sequences $\{x_i\}$ and $\{y_i\}$ and integers $\alpha < \beta < \gamma$,

$$\begin{aligned} \textit{if} \quad & \mathscr{J}\left(\{x_i\}_{i\in[\alpha,\beta)}\right) \le \mathscr{J}\left(\{y_i\}_{i\in[\alpha,\beta)}\right) \\ \textit{and} \quad & \mathscr{J}\left(\{x_i\}_{i\in[\beta,\gamma)}\right) \le \mathscr{J}\left(\{y_i\}_{i\in[\beta,\gamma)}\right), \\ \textit{then} \quad & \mathscr{J}\left(\{x_i\}_{i\in[\alpha,\gamma)}\right) \le \mathscr{J}\left(\{y_i\}_{i\in[\alpha,\gamma)}\right). \end{aligned}$$

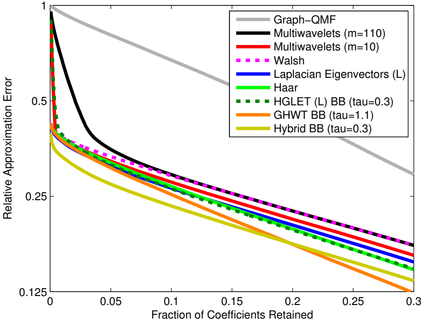
Given a signal $\boldsymbol{f}$ on a graph G and a hierarchical tree for the graph, the set of expansion coefficients returned by the best basis algorithm is the set that ***minimizes $\mathscr{J}$ over all choosable sets of coefficients*** *in the dictionary (or dictionaries) considered.*

The Minnesota road network ($N = 2640$) has **over 10^{453} choosable bases!**

Experimental Result: Approximation



(a) 24 hour traffic volume data on the Toronto road network



(b) n-term nonlinear approximation error

Motivation

There are many examples of data in matrix format:

- Images
- Ratings/Reviews
  - Rows → Netflix users
  - Columns → movies
  - $A(i, j)$ → user i's rating of movie j on a 1-5 scale
- Spatiotemporal data
  - Rows → sensors
  - Columns → times
  - $A(i, j)$ → sensor i's temperature reading at time j

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

Method

1. Use the matrix data to recursively partition the rows and the columns
   - Given a matrix $A \in \mathbb{R}^{N_R \times N_C}$, Dhillon (2001) views the rows and columns as the two sets of nodes in a bipartite graph.

     A_{ij} denotes the weight between the node for row i and the node for column j.

$$W = \begin{bmatrix} \mathbf{0} & A \\ A^T & \mathbf{0} \end{bmatrix}$$

2. Use the GHWT and best-basis algorithm to analyze the matrix
   i. Analyze along the rows and extract the best basis
   ii. Analyze the row best basis coefficients along the columns and extract the best basis

Method

1. Use the matrix data to recursively partition the rows and the columns
   - Given a matrix $A \in \mathbb{R}^{N_R \times N_C}$, Dhillon (2001) views the rows and columns as the two sets of nodes in a bipartite graph.

     A_{ij} denotes the weight between the node for row i and the node for column j.

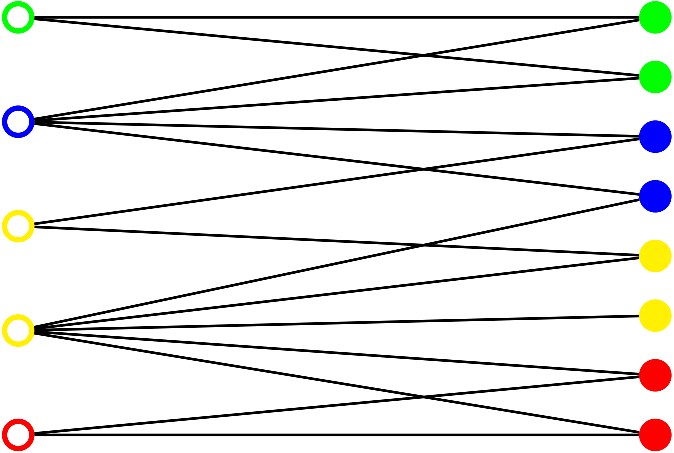
$$W = \begin{bmatrix} \mathbf{0} & A \\ A^T & \mathbf{0} \end{bmatrix}$$

2. Use the GHWT and best-basis algorithm to analyze the matrix
   i. Analyze along the rows and extract the best basis
   ii. Analyze the row best basis coefficients along the columns and extract the best basis

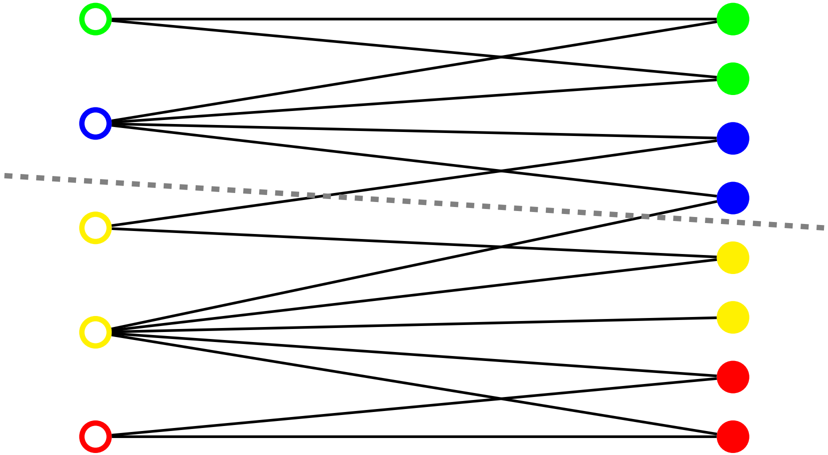
Matrix Partitioning à la Dhillon (2001)

$$
\begin{bmatrix}
* & * & & & & & & \\
* & * & * & * & & & & \\
& & * & & * & & & \\
& & & * & * & * & * & * \\
& & & & & & * & *
\end{bmatrix}
$$

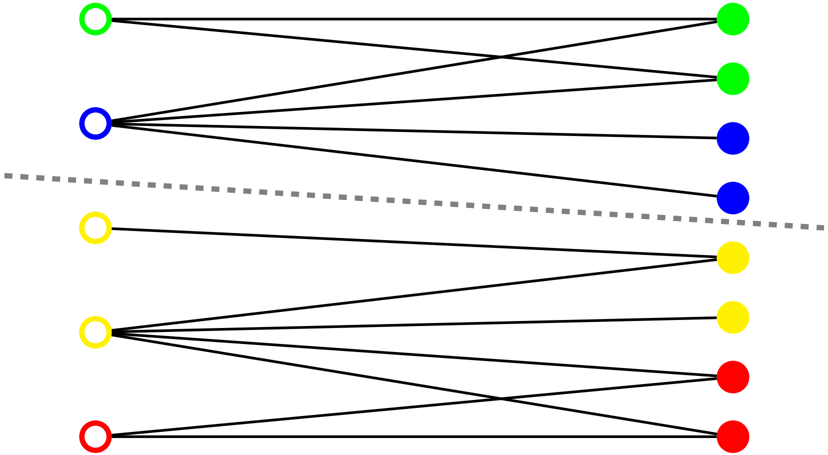
Matrix Partitioning à la Dhillon (2001)



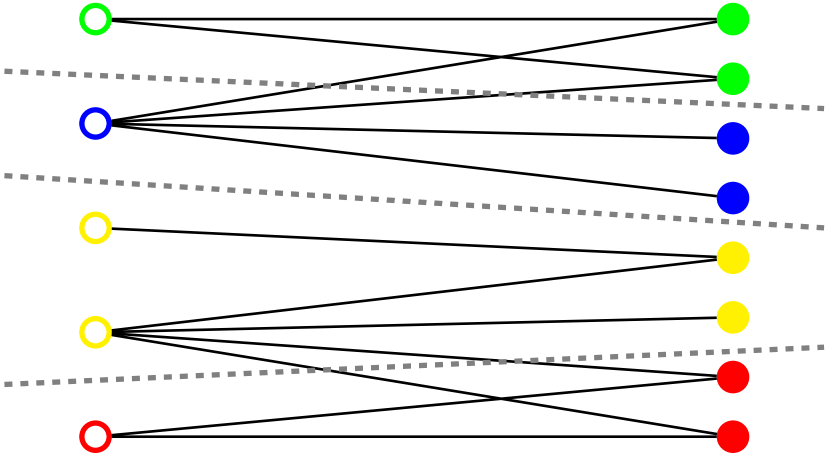
Matrix Partitioning à la Dhillon (2001)



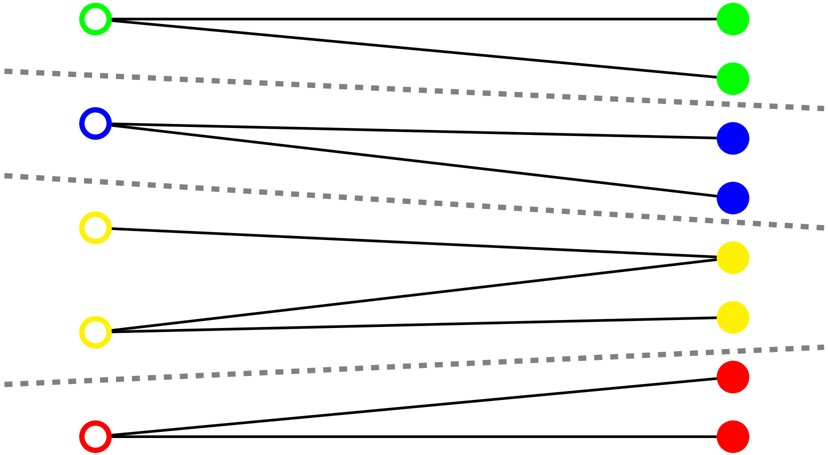
Matrix Partitioning à la Dhillon (2001)



Matrix Partitioning à la Dhillon (2001)



Matrix Partitioning à la Dhillon (2001)



Example 1

Dataset: the Science News database (1153×1042)

- Columns → article abstracts from 8 fields: Anthropology; Astronomy; Behavioral Sciences; Earth Sciences; Life Sciences; Math & CS; Medicine; Physics
- Rows → (appropriately chosen) words
- $A(i, j)$ → the relative frequency of word i in abstract j ⇒ all column sums are 1
- 10.1% sparsity

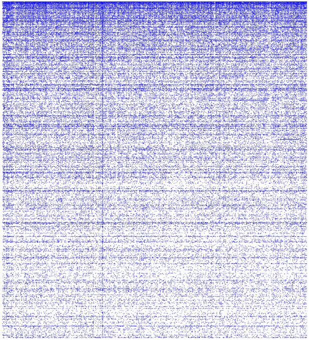


Figure: Science News database (original order).

Example 1

Dataset: the Science News database (1153×1042)

- Columns → article abstracts from 8 fields: Anthropology; Astronomy; Behavioral Sciences; Earth Sciences; Life Sciences; Math & CS; Medicine; Physics
- Rows → (appropriately chosen) words
- $A(i, j)$ → the relative frequency of word i in abstract j ⇒ all column sums are 1
- 10.1% sparsity

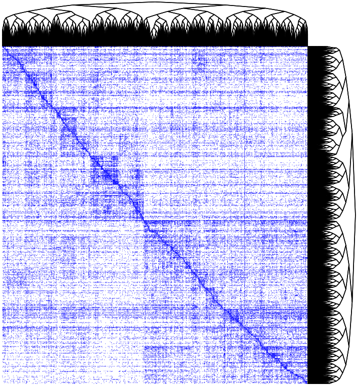


Figure: Science News database (reordered rows and columns).

Example 1

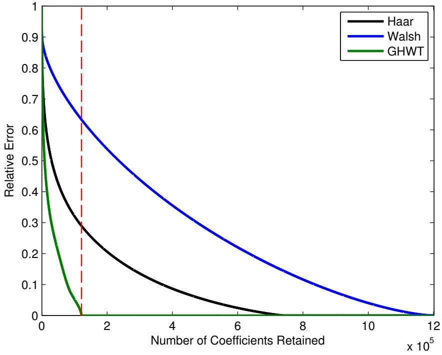


Figure: Haar basis vs. Walsh basis vs. GHWT best basis approximation results. The vertical line denotes the number of nonzero entries in the matrix (**10.1%**).

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

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

Example 1

Cost functional: 1-norm

The GHWT best basis is almost exactly the canonical basis, but the rows and columns that it combines provide insight.

Combined Rows:

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

Combined Columns:

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

Example 1

The **0.1-quasinorm** also combines the words

- "orbiting" and "extrasolar"
- "tornado," "tornadoe," and "meteorologist"

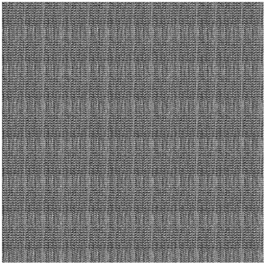
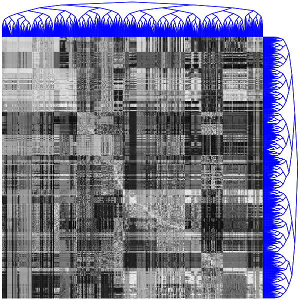
along with 8 pairs of documents, 1 group of three, and 1 group of four.

The **0.01-quasinorm** combines 1 additional pair of documents.

The **0.001-quasinorm** returns the canonical basis.

Example 2

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

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

Example 2

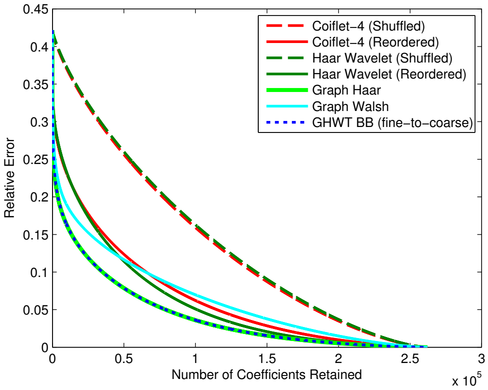


Figure: Approximation results. The "shuffled" and "reordered" results are for the cases that the shuffled image (middle figure on previous page) and reordered image (figure on the right) was analyzed, respectively.

- Cost functional: **1-norm**
- Total number of orthonormal bases searched: $< 6.37 \times 10^{173}$
- The GHWT best basis nearly matches the Haar basis
- The GHWT best basis performs much better than the Coiflet and Haar bases, which are fixed and therefore cannot account for the geometry of the data

Example 2

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

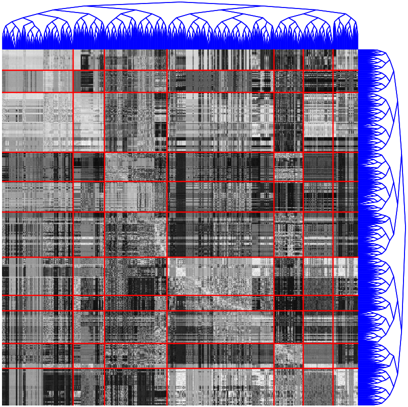


Figure: The coarse-to-fine row and column best bases for "Barbara" using the **0.5-quasinorm** as our cost functional.

Example 2

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

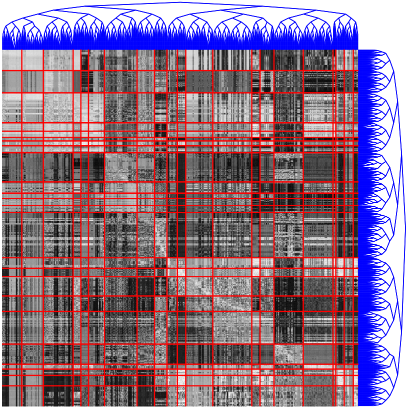


Figure: The coarse-to-fine row and column best bases for “Barbara” using the **0.1-quasinorm** as our cost functional.

Example 2

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

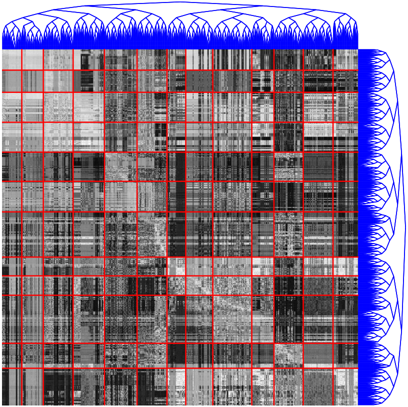
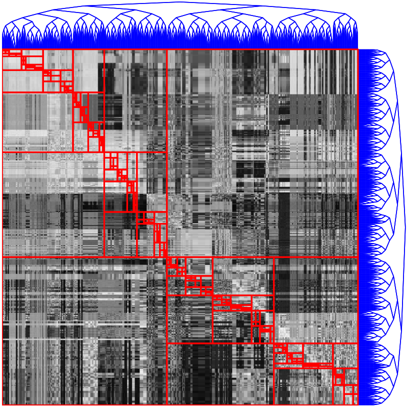


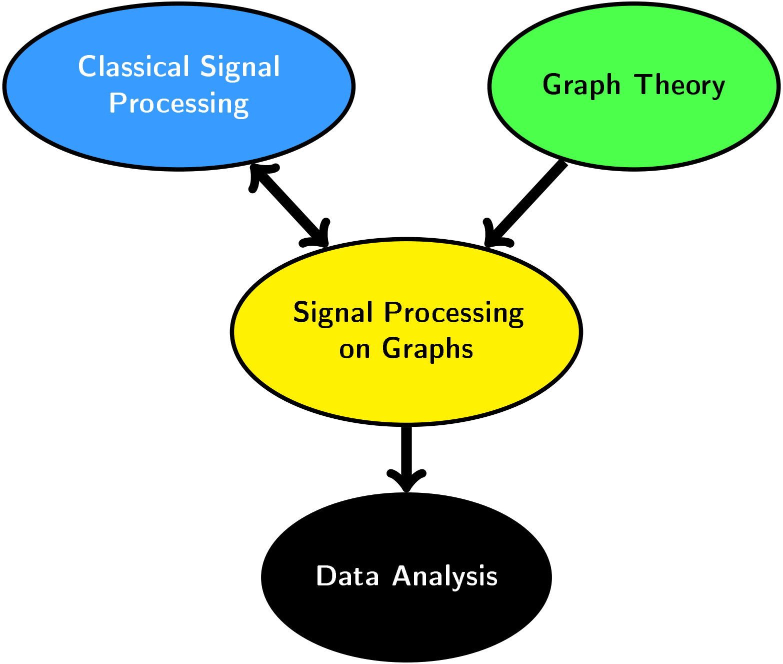
Figure: The coarse-to-fine row and column best bases for "Barbara" using the 0.1-quasinorm as our cost functional; regions with fewer than $[N_R/20] = [N_C/20] = 26$ nodes were not considered in the best basis search.

Example 2

Future work: instead of searching for the tensor best basis, search among all combinations of row and column bases.



- We have **developed**
  - the GHWT
  - the corresponding best basis algorithm
- We have **proven**
  - the best basis guarantee
- We have **demonstrated**
  - the effectiveness of the GHWT for approximation
  - using the GHWT for matrix data analysis
- In other work, we have
  - developed the HGLET (another overcomplete multiscale transform)
  - proven approximation bounds for the HGLET and GHWT
  - denoised signals on graphs with the HGLET and GHWT
  - used the HGLET to simultaneously segment, denoise, and compress classical 1-D signals


Classical Signal Processing
Graph Theory
Signal Processing on Graphs
Data Analysis

Publications:

- J. Irion & N. Saito: "Hierarchical graph Laplacian eigen transforms," *JSIAM Letters*, vol. 6, pp. 21–24, 2014.
- J. Irion & N. Saito: "The generalized Haar-Walsh transform," *Proc. 2014 IEEE Workshop on Statistical Signal Processing*, pp. 488-491, 2014.
- J. Irion & N. Saito, "Applied and computational harmonic analysis on graphs and networks," *Wavelets and Sparsity XVI* (M. Papadakis, V.K. Goyal, D. Van De Ville, eds.), *Proc. SPIE*
- J. Irion, "Multiscale Transforms for Signals on Graphs: Methods and Applications," Ph.D. dissertation, University of California, Davis, 2015.
- J. Irion & N. Saito, "Efficient Approximation and Denoising of Graph Signals Using the Multiscale Basis Dictionaries," submitted for publication, 2016.

Funding by

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