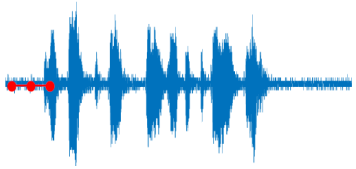
Graph Signal Processing

Introduction and Research at SPS

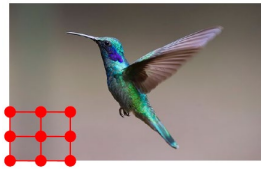
Geert Leus
g.j.t.leus@tudelft.nl

Why signal processing on graphs?

- Digital signal processing deals with signals in the Euclidean domain

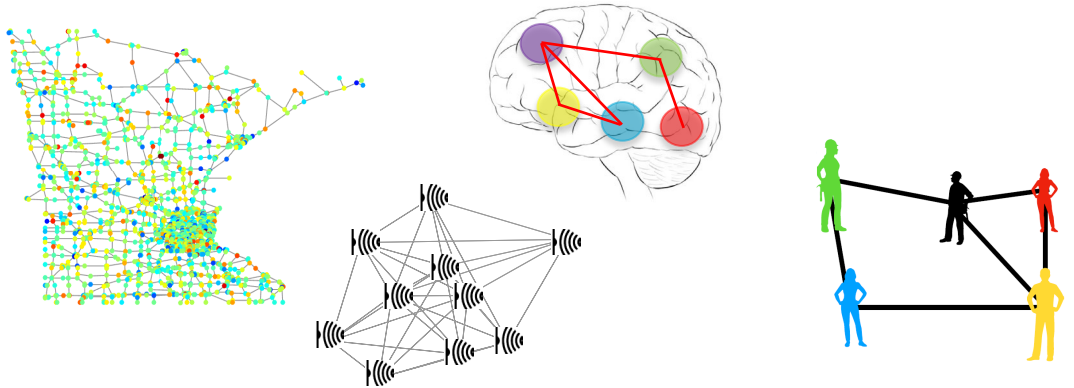


Audio: temporal regularity



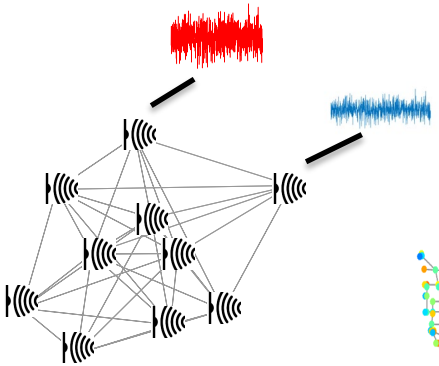
Images: spatial regularity

- More and more data comes from networks (graphs) with an irregular structure
  - Explicit networks:
    - Transportation
    - Brain (anatomical)
    - Sensors (proximity)
  - Implicit networks:
    - Social
    - Brain (functional)
    - Sensors (similarity)

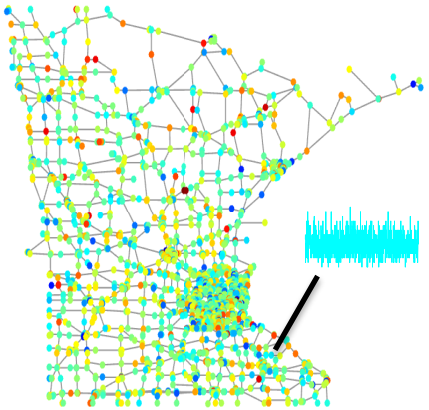


Why signal processing on graphs?

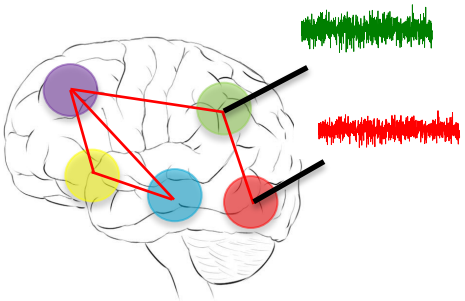
- Data can be viewed as signals on top of the graph: graph signals



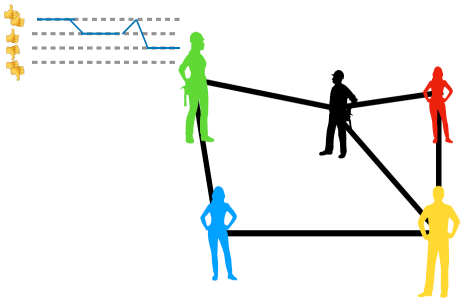
Temperature



Number of cars passing



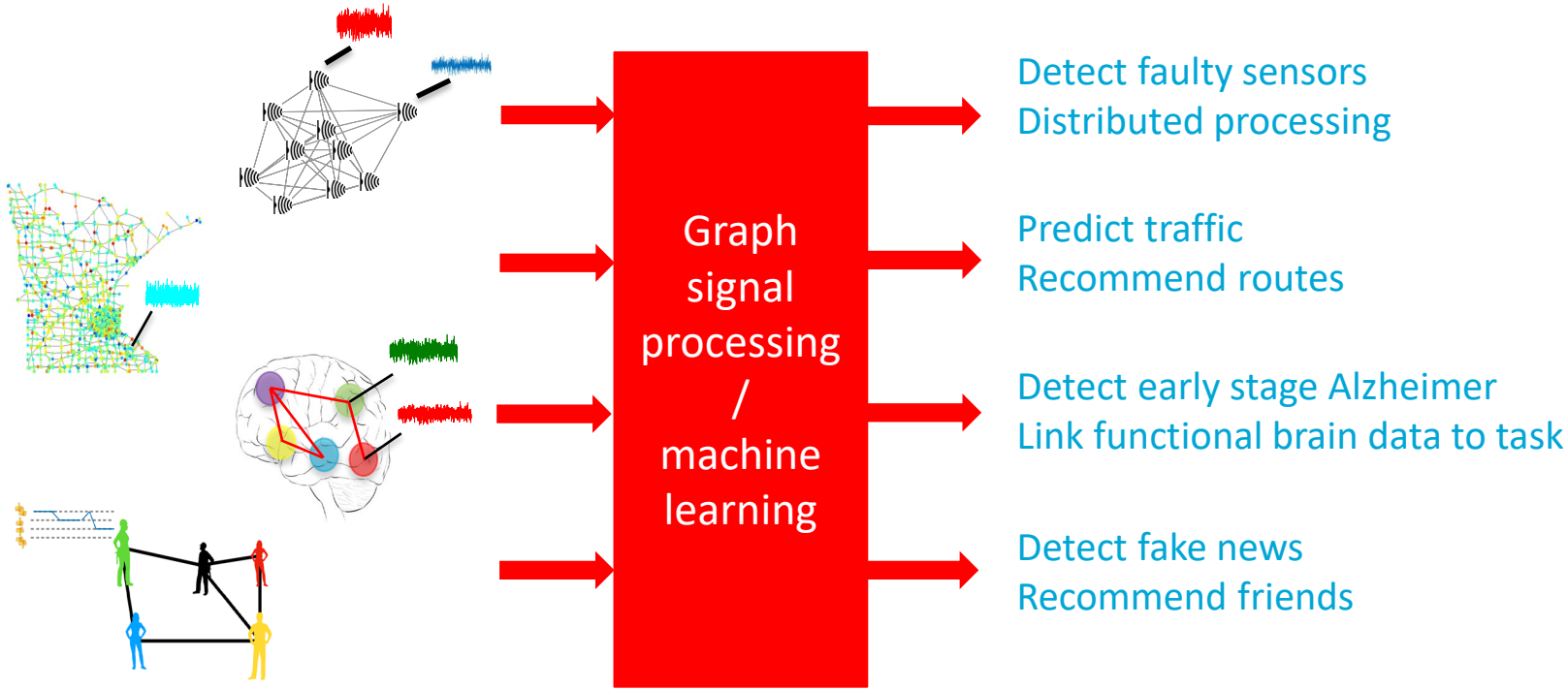
fMRI time series



Opinion profile

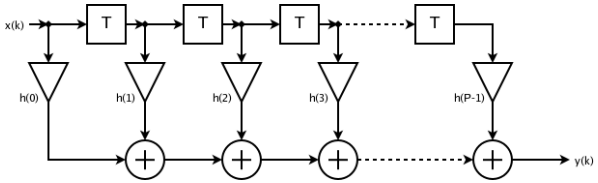
Why signal processing on graphs?

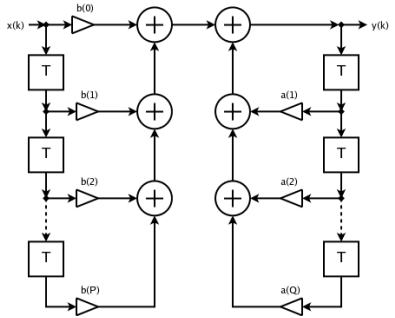
- Graph signal processing: Given the graph and the signal, perform inference tasks on the graph signal exploiting prior knowledge of the graph



Why signal processing on graphs?

- These applications require new tools that can exploit the structure
- Filters are the central tool for digital signal processing and machine learning





  - Linear systems with theoretical guarantees
  - Links with the spectral response
  - Can embed the data structure into the learning function
  - Reduce the number of learnable parameters in neural networks
  - Efficient implementations of neural networks

Goal: How can we extend filters for data that is linked to networks?

Overview

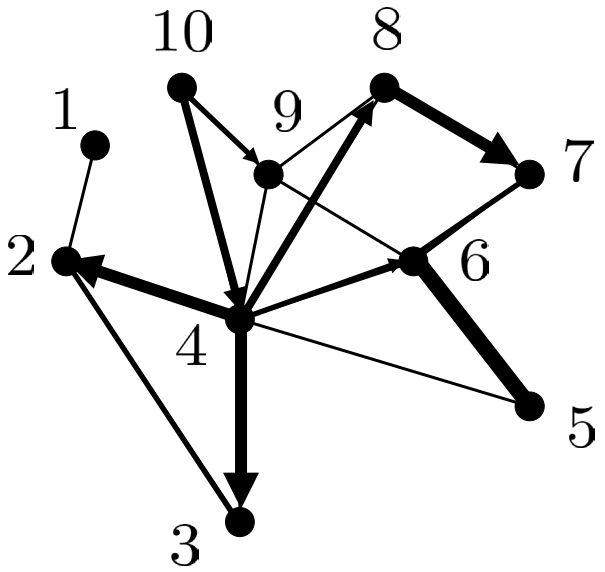
- Introduction to graph signal processing
  - Graphs and graph signals
  - Signal variability and the graph Fourier transform
  - Graph filters

- GSP research at the signal processing systems (SPS) group
  - Graph-time signal processing
  - Distributed optimization using graph filters
  - Other activities

Graphs

Datasets with irregular support can be represented using a graph

- Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$
- $\mathcal{V}$ is the set of nodes, $|\mathcal{V}| = N$
- $\mathcal{E}$ is the set of edges, $|\mathcal{E}| = M$
- $\mathbf{A} \in \mathbb{R}_{+}^{N \times N}$ is the adjacency matrix
- $\mathbf{L} = \mathbf{D} - \mathbf{A}$ is the Laplacian matrix
- $\mathcal{N}_i$ is the neighborhood of node i

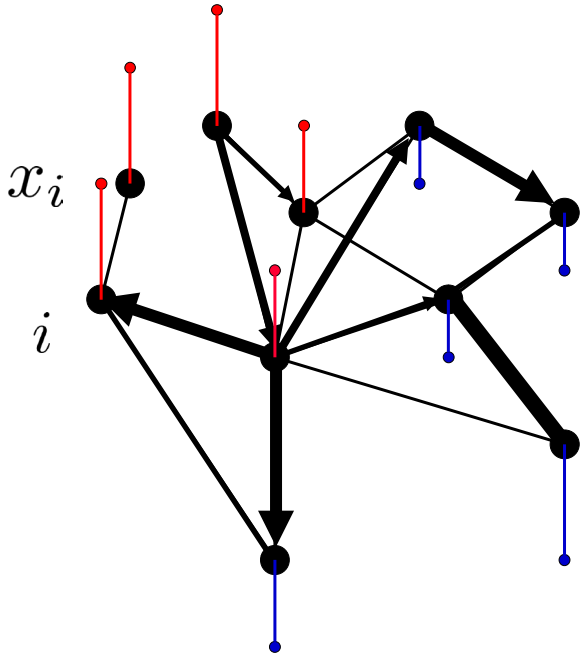


Graph signals

On top of the graph we can consider a graph signal

- Scalar value x_i on every node i
- Collected all values into a vector

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} \in \mathbb{R}^N$$



Graph shift operator

The adjacency and Laplacian are graph shift operators (GSOs)

- A GSO $\mathbf{S} \in \mathbb{R}^{N \times N}$ is any matrix capturing the graph structure
- Definition: $S_{ji} \neq 0$ implies that $i = j$ or $(i, j) \in \mathcal{E}$
- Other candidates for $\mathbf{S}$ are the normalized matrices

$\lambda_{\max}$: eigenvalue with maximum absolute value

Normalized adjacency $\mathbf{A}_n = \mathbf{A}/|\lambda_{\max}|$

Normalized Laplacian $\mathbf{L}_n = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$

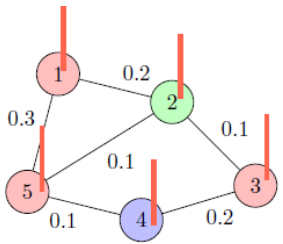
Random-walk Laplacian $\mathbf{L}_{rw} = \mathbf{D}^{-1}\mathbf{L}$

Signal diffusion over graphs

- Signal diffuses along the edges – graph signal shifting $\mathbf{x}^{(1)} = \mathbf{S}\mathbf{x}$

Undirected graph

$$\mathbf{S} = \mathbf{L},\ \mathbf{x} = \mathbf{1}$$



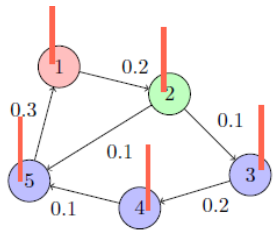
$$\mathbf{x}^{(1)} = \mathbf{L}\mathbf{x}, \quad x_i^{(1)} = \sum_{j \in \mathcal{N}_i} A_{ij}(x_i - x_j)$$

$$\mathcal{N}_2 = \{1, 3, 5\}$$

$$x_2^{(1)} = (0.2 + 0.1 + 0.1)x_2 - 0.2x_1 - 0.1x_3 - 0.1x_5 = 0$$

Directed graph

$$\mathbf{S} = \mathbf{A},\ \mathbf{x} = \mathbf{1}$$



$$\mathbf{x}^{(1)} = \mathbf{A}\mathbf{x}, \quad x_i^{(1)} = \sum_{j \in \mathcal{N}_i} A_{ij}x_j$$

$$\mathcal{N}_2 = \{1\}$$

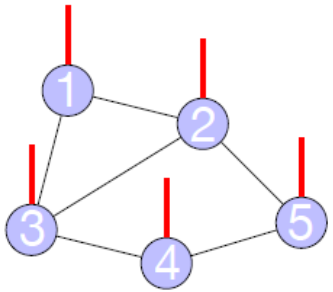
$$x_2^{(1)} = 0.2x_1 = 0.2$$

Signal variability

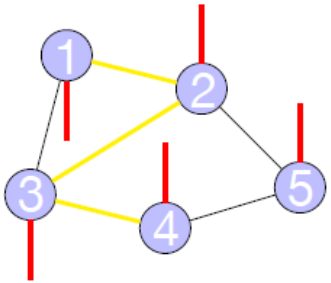
- Undirected graph $\mathcal{G}$ with GSO $\mathbf{S} = \mathbf{L}$ and signal $\mathbf{x}$

  How fast does the signal change over the graph?

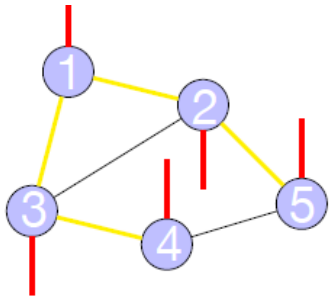
- Important for filtering
- Initial thought: count sign changes



Constant, No sign change



Slow-varying, 3 sign change



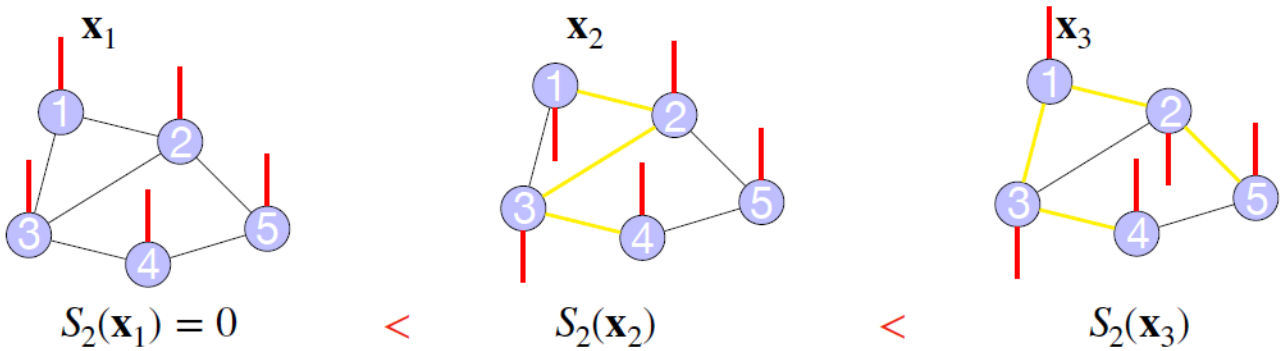
High-varying, 4 sign change

Signal variability

- Consider graph Laplacian quadratic form

$$S_2(\mathbf{x}) = \mathbf{x}^T \mathbf{L} \mathbf{x} = \sum_{i,j=1}^{N} A_{ij}(x_i - x_j)^2$$

- Quantifies how much the signal changes over the graph



Signal variability

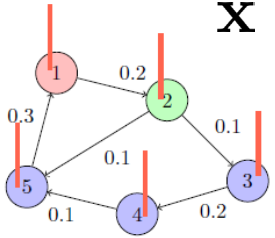
- Directed graph $\mathcal{G}$ with GSO $\mathbf{S} = \mathbf{A}$ and signal $\mathbf{x}$
- Consider the total variation

$$\mathrm{TV}(\mathbf{x}) = \|\mathbf{x} - \mathbf{A}_n\mathbf{x}\|_1 = \|\mathbf{x} - \mathbf{x}_n^{(1)}\|_1$$

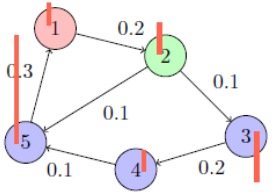
$$\mathbf{A}_n = \mathbf{A}/|\lambda_{\max}|$$

- Normalization for practical reasons
- TV high: signal changes substantially
- TV low: signal changes little

$\mathbf{x}$



$\mathbf{x}_n^{(1)}$



Graph Fourier transform

- Consider the eigendecomposition $\mathbf{S} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1}$

$$\mathbf{S} = \begin{pmatrix} | & & | \\ \mathbf{v}_1 & \dots & \mathbf{v}_N \\ | & & | \end{pmatrix} \begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_N \end{pmatrix} \begin{pmatrix} | & & | \\ \mathbf{v}_1 & \dots & \mathbf{v}_N \\ | & & | \end{pmatrix}^{-1}$$

graph modes graph frequencies

- Graph Fourier transform (GFT): $\hat{\mathbf{x}} = \mathbf{V}^{-1}\mathbf{x}$
- Inverse GFT: $\mathbf{x} = \mathbf{V}\hat{\mathbf{x}}$

How to order them? How to measure their variability?

Graph Fourier transform

- Undirected graph $\mathcal{G}$ with GSO $\mathbf{S} = \mathbf{L}$
- GSO is symmetric and positive semi-definite
  - Eigenvalue decomposition always exists
  - Eigenvectors are real-valued orthogonal (one is all-one vector)
  - Eigenvalues are real-valued and positive (one is zero)

$$\mathbf{L} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$$

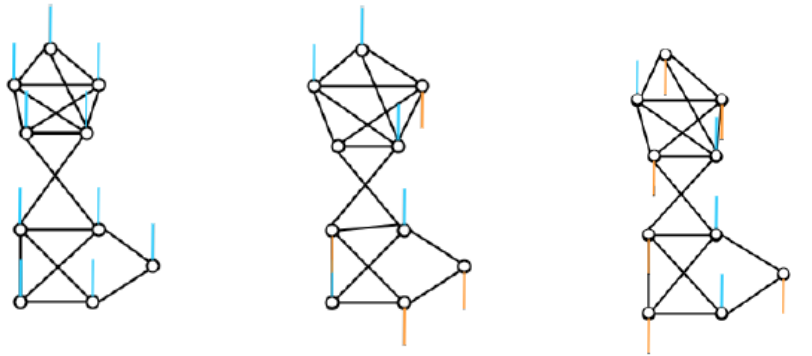
$$0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_N$$

Graph Fourier transform

- Variability measured using Laplacian quadratic form $\mathbf{x}^T\mathbf{L}\mathbf{x}$
- Consider each eigenvector as a graph signal $\mathbf{x} = \mathbf{v}_i$

- Graph mode variability: $\mathbf{v}_i^T\mathbf{L}\mathbf{v}_i = \lambda_i\,, \quad 0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_N$

The variability increases with λ_i

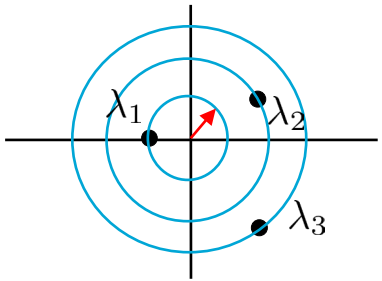


Graph Fourier transform

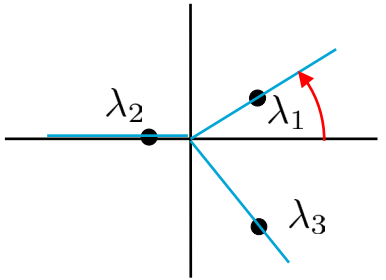
- Directed graph $\mathcal{G}$ with GSO $\mathbf{S} = \mathbf{A}$
- GSO is not necessarilly symmetric
  - Eigenvalue decomposition might not exist
  - Eigenvectors and eigenvalues can be complex-valued

$$\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1}$$

amplitude order



phase order



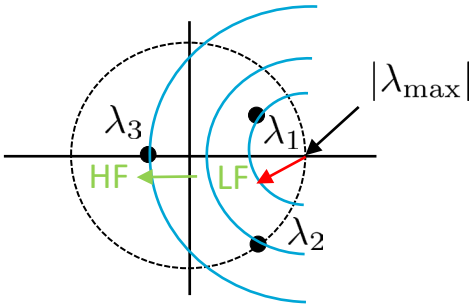
Graph Fourier transform

- Variability measured using total variation $\|\mathbf{x} - \mathbf{A}_n\mathbf{x}\|_1$
- Consider each eigenvector as a graph signal $\mathbf{x} = \mathbf{v}_i$

- Graph mode variability: $\|\mathbf{v}_i - \mathbf{A}_n\mathbf{v}_i\|_1 = \left|1 - \frac{\lambda_i}{|\lambda_{\max}|}\right| \|\mathbf{v}_i\|_1$

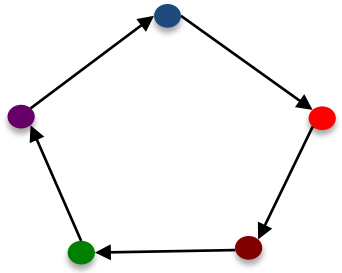
The variability increases with the distance to $(|\lambda_{\max}|, 0)$

variability order



Time-domain as a graph

- The DFT matrix and the traditional frequency grid obtained by the adjacency matrix of the cycle graph

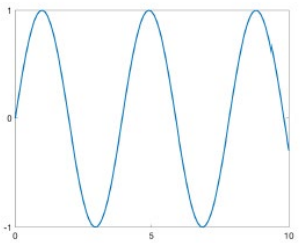


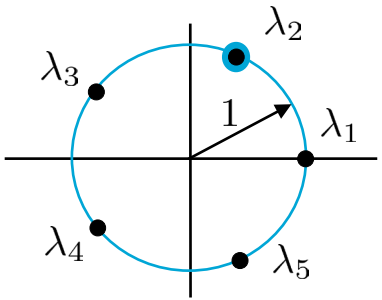
$$\mathbf{S} = \mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1}$$

$$\mathbf{V} = \frac{1}{N}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & e^{\frac{j2\pi}{5}} & e^{\frac{j2\pi 2}{5}} & e^{\frac{j2\pi 3}{5}} & e^{\frac{j2\pi 4}{5}} \\ 1 & e^{\frac{j2\pi*2}{5}} & e^{\frac{j2\pi 2*2}{5}} & e^{\frac{j2\pi 3*2}{5}} & e^{\frac{j2\pi 4*2}{5}} \\ 1 & e^{\frac{j2\pi*3}{5}} & e^{\frac{j2\pi 2*3}{5}} & e^{\frac{j2\pi 3*3}{5}} & e^{\frac{j2\pi 4*3}{5}} \\ 1 & e^{\frac{j2\pi*4}{5}} & e^{\frac{j2\pi 2*4}{5}} & e^{\frac{j2\pi 3*4}{5}} & e^{\frac{j2\pi 4*4}{5}} \end{bmatrix}$$

$$\mathbf{\Lambda} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & e^{\frac{j2\pi}{5}} & 0 & 0 & 0 \\ 0 & 0 & e^{\frac{j2\pi 2}{5}} & 0 & 0 \\ 0 & 0 & 0 & e^{\frac{j2\pi 3}{5}} & 0 \\ 0 & 0 & 0 & 0 & e^{\frac{j2\pi 4}{5}} \end{bmatrix}$$

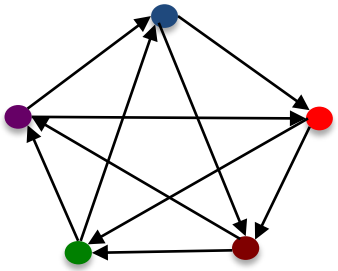
$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

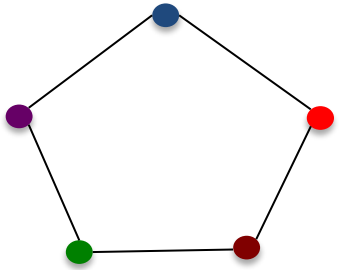




Time domain as a graph

- Any circulant graph (directed or not) in principle leads to the DFT as the matrix that diagonalises the GSO





$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

$$\mathbf{L} = \begin{bmatrix} 2 & -1 & 0 & 0 & -1 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 1 & 2 & -1 \\ -1 & 0 & 0 & -1 & 2 \end{bmatrix}$$

Graph filters

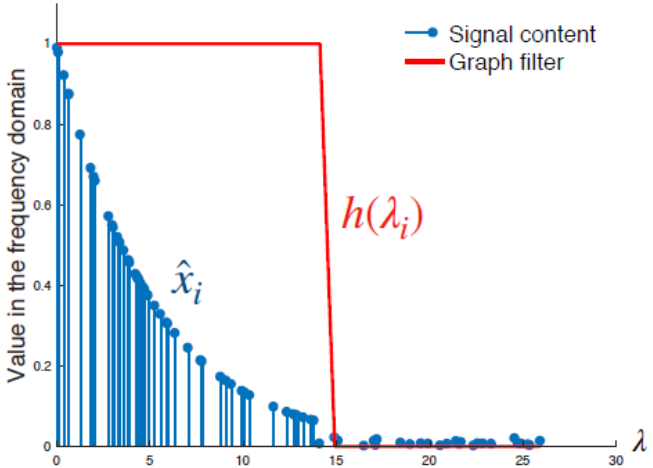
- (I)GFT: $\hat{\mathbf{x}} = \mathbf{V}^{-1}\mathbf{x} \;\Leftrightarrow\; \mathbf{x} = \mathbf{V}\hat{\mathbf{x}}$
- Graph filters can be used to modify the frequency content

$$\hat{y}_n = h(\lambda_n)\hat{x}_n$$

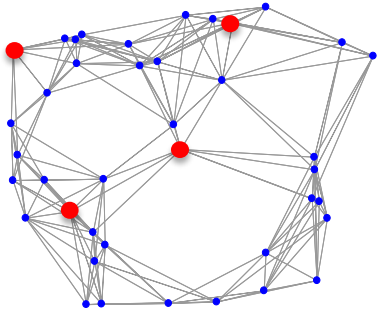
$$\begin{cases} \hat{\mathbf{y}} = h(\mathbf{\Lambda})\hat{\mathbf{x}} \\ h(\mathbf{\Lambda}) = \mathrm{diag}\{h(\lambda_n)\} \end{cases}$$

$$\mathbf{y} = \mathbf{V}h(\mathbf{\Lambda})\mathbf{V}^{-1}\mathbf{x} = \mathbf{H}\mathbf{x}$$

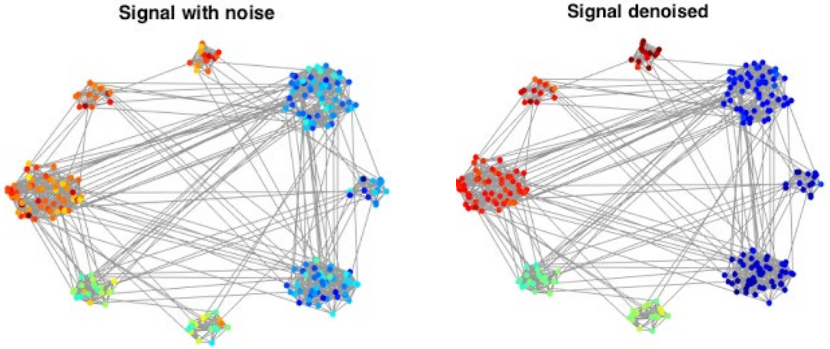
Shift invariance: $\mathbf{HS} = \mathbf{SH}$



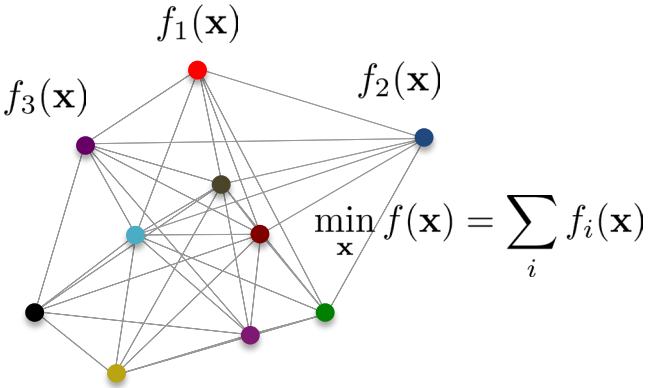
Applications of graph filters



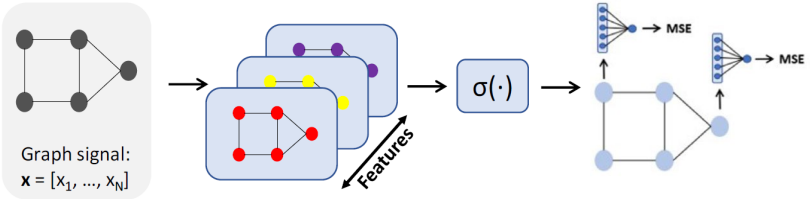
Interpolation



Denoising



Distributed optimization

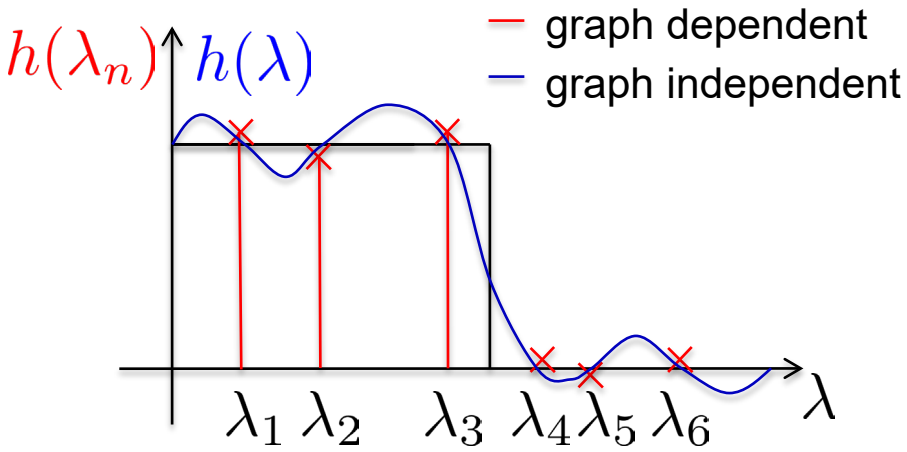


Deep learning

Graph filter design

$$\mathbf{y} = \mathbf{V}h(\mathbf{\Lambda})\mathbf{V}^{-1}\mathbf{x} = \mathbf{H}\mathbf{x} \quad \Leftrightarrow \quad \mathbf{H}\mathbf{S} = \mathbf{S}\mathbf{H}$$

- Graph-dependent vs. graph-independent (universal) filter design



[Shuman'11, DCOSS]
[Sandryhaila'13, TSP]
[Shuman'13, SPM]

Graph filter implementation

$$\mathbf{y} = \mathbf{V}h(\mathbf{\Lambda})\mathbf{V}^{-1}\mathbf{x} = \mathbf{H}\mathbf{x} \quad \Leftrightarrow \quad \mathbf{H}\mathbf{S} = \mathbf{S}\mathbf{H}$$

- Freqency-domain vs. vertex-domain implementation
  - No fast GFT implementations
  - Need for parameterized filters in the vertex-domain
    - Parameterization need to be eligible
      - Polynomial functions of $\mathbf{S}$
      - Rational polynomial functions of $\mathbf{S}$
    - Parameterization should improve implementation efficiency
    - Distributed implementation possible?

FIR graph filters

- Polynomial of the GSO:

  Finite impulse response (FIR) graph filter

$$\mathbf{y} = \mathbf{H}\mathbf{x} \qquad \mathbf{H} = \sum_{k=0}^{K} \phi_k \mathbf{S}^k$$

[Sandryhaila'13, TSP]

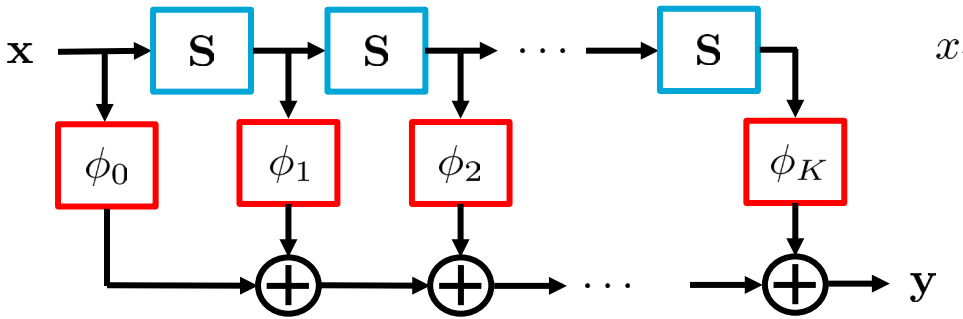
- The graph filter is eligible (diagonizable by the GFT)
- The frequency response is given by

$$h(\lambda_n) = \sum_{k=0}^{K} \phi_k \lambda_n^k$$

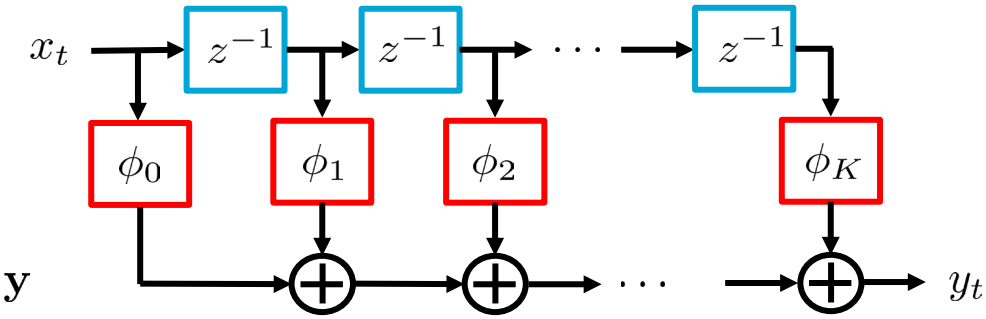
FIR graph filters

- The GSO is the dual of a time delay in temporal FIR filters

graph FIR



temporal FIR



- In both cases the output is a sum of shifted signals
- The graph FIR carries the notion of convolution in the graph domain

FIR properties

$$\mathbf{y} = \sum_{k=0}^{K} \phi_k \mathbf{S}^k \mathbf{x}$$

- Number of parameters $K+1$
- No stability issues
- Efficient and distributed implementation
- Implementation cost of $\mathcal{O}(KM)$
- Least squares design – for (un)directed graphs, graph-(in)dependent
- Orthogonal polynomials design – for (un)directed graphs, graph-(in)dependent
- Good approximation requires high filter orders, which leads to ill-conditioning

ARMA graph filters

- Rational polynomial of the GSO:

  Autoregressive moving average (ARMA) graph filter

$$\mathbf{y} = \mathbf{H}\mathbf{x} \qquad \mathbf{H} = \left(\mathbf{I} + \sum_{p=1}^{P} \psi_p \mathbf{S}^p\right)^{-1} \left(\sum_{q=0}^{Q} \varphi_q \mathbf{S}^q\right)$$

- The graph filter is eligible (diagonizable by the GFT)
- The frequency response is given by

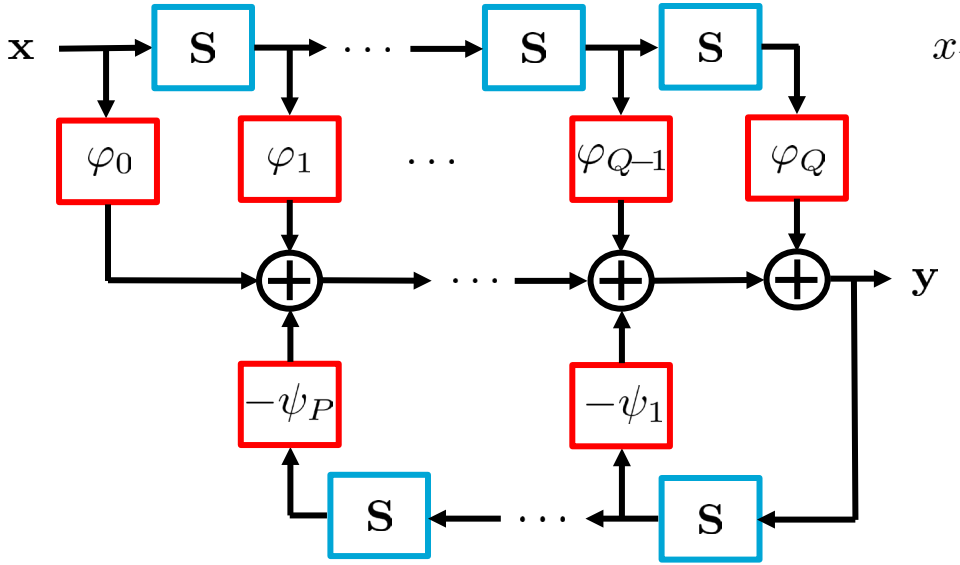
$$h(\lambda_n) = \frac{\sum_{q=0}^{Q} \varphi_q \lambda_n^q}{1 + \sum_{p=1}^{P} \psi_p \lambda_n^p}$$

Isufi, Loukas, Simonetto, Leus, *Autoregressive moving average graph filtering*, IEEE TSP, 2017

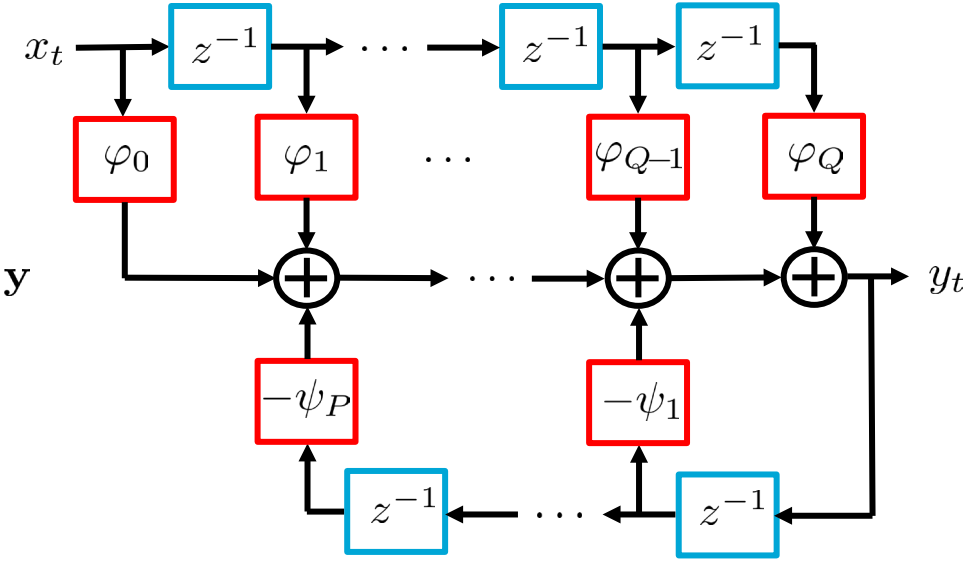
ARMA graph filters

- The GSO is the dual of a time delay in temporal ARMA filters

graph ARMA



temporal ARMA



- Feedback is easy in temporal ARMA but difficult in graph ARMA

ARMA properties

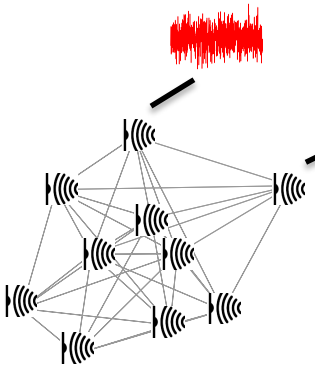
$$\mathbf{y} = \left(\mathbf{I} + \sum_{p=1}^{P} \psi_p \mathbf{S}^p \right)^{-1} \left(\sum_{q=0}^{Q} \varphi_q \mathbf{S}^q \right) \mathbf{x}$$

- Number of parameters $P + Q + 1$
- Stability guaranteed by invertibility
- Efficient and distributed implementation
- Implementation cost of $\mathcal{O}(PM)$ per iteration
- (Iterative) least squares design – for (un)directed graphs, graph-(in)dependent
- Good approximation for low filter orders, so less ill-conditioning
- Exact solution for many GSP problems

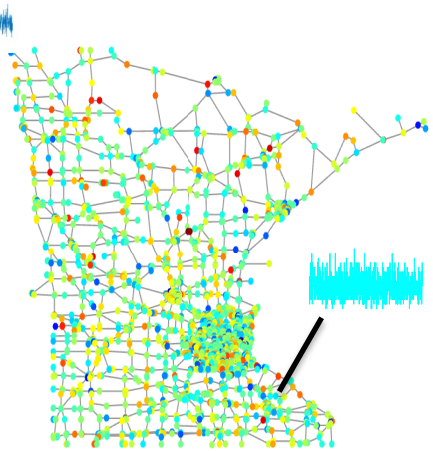
GSP Research at SPS

Time-varying data on graphs

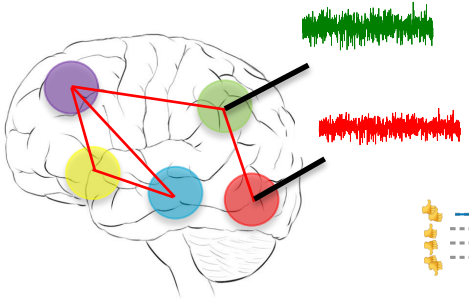
- So far we only considered a static signal (one scalar value per node)
- But signals on nodes are often time varying



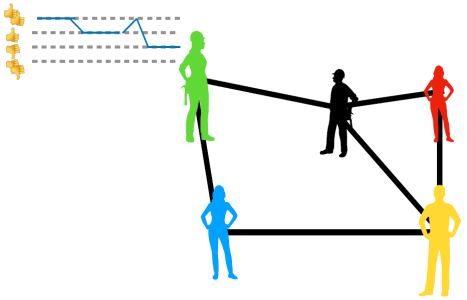
Temperature



Number of cars passing



fMRI time series

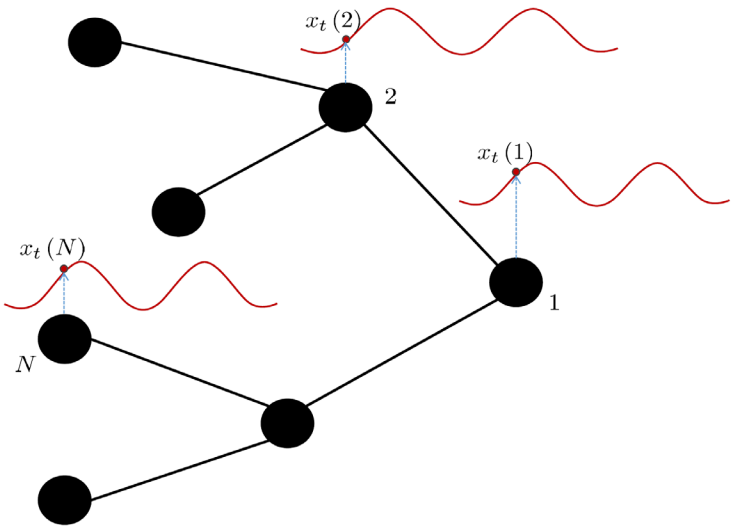


Opinion profile

- Data now have a graph depency as well as a temporal dependency

Time-varying data on graphs

- This leads to a time series over the nodes: graph-time signals



Applications:

- Forecasting future values
- Interpolate graph-time signals
- Detect anomalous patterns
- Classify graph-time signals

- Goal: How to exploit the joint graph-time coupling?

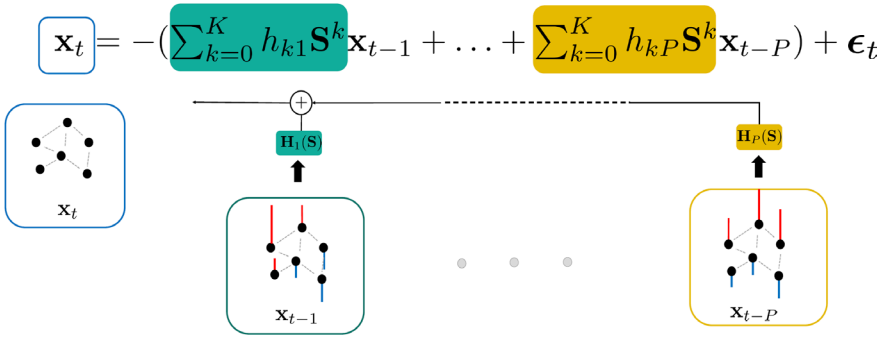
Graph-time signal processing

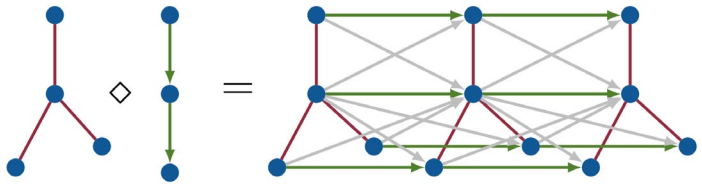
Graph recursive models

- Recursive time-aware filters
- Low complexity
- Scalable
- #parameters independent of graph

Product graphs

- Join the two graphs
- Interpretable
- Ready to use graph methods
- #parameters independent of graph

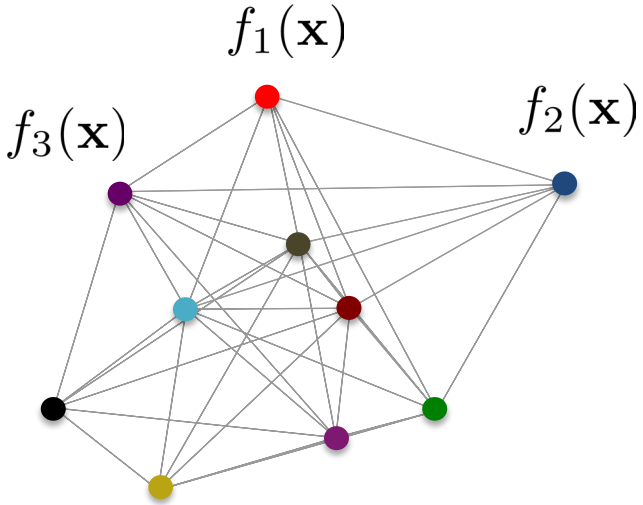




Distributed optimization

- Goal is to solve the following problem distributively

$$\min_{\mathbf{x}} f(\mathbf{x}) = \sum_{i=1}^{N} f_i(\mathbf{x})$$



- Requires data exchanges within the network: graph diffusions

Distributed optimization

- We assume input data is available through $\mathbf{y}$
- We focus on problems of the form

input data

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \sum_{i=1}^{N} f_i(\mathbf{y}; \mathbf{x})$$

- We assume solution is a linear transformation of the input data

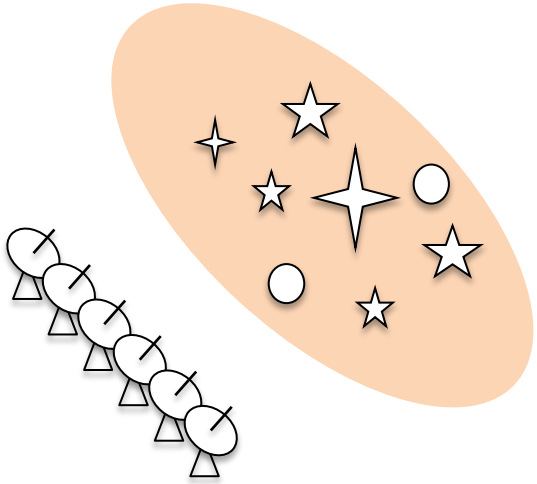
$$\mathbf{x}^* = \tilde{\mathbf{H}}\mathbf{y}$$

- Can be relaxed using graph CNNs

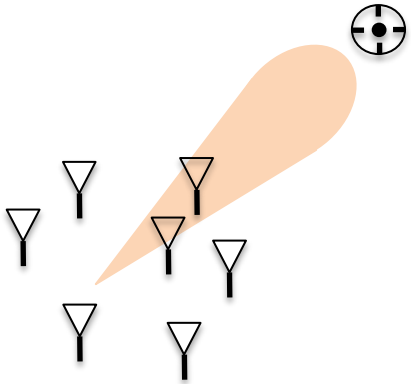
Distributed optimization

Average consensus

Distributed imaging



Distributed beamforming



Leveraging graph filters

- All these applications have a global solution of the form

$$\mathbf{x}^* = \tilde{\mathbf{H}}\mathbf{y}$$

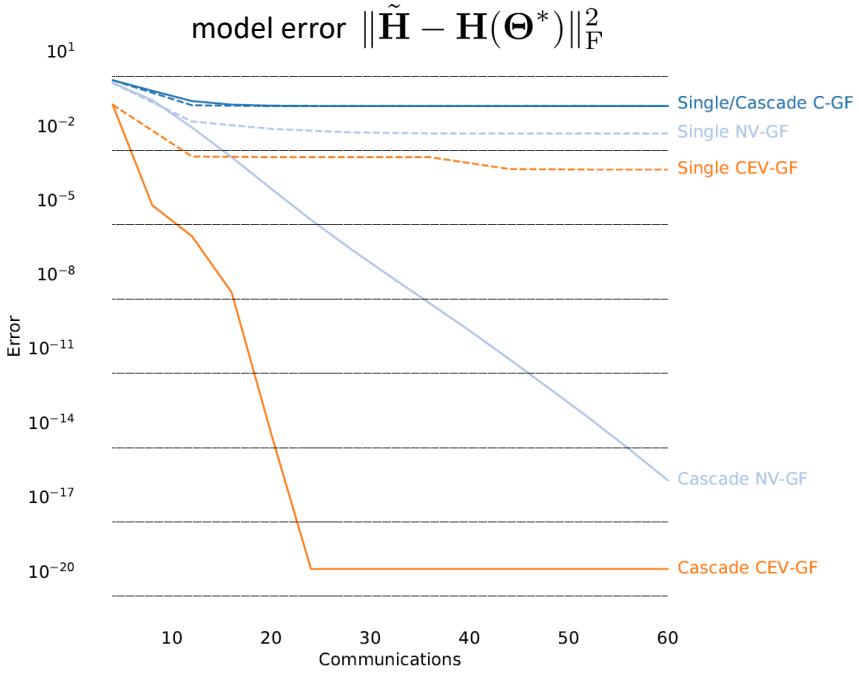
- Implement the known linear operation in a distributed manner
- Graph filters are distributed by nature

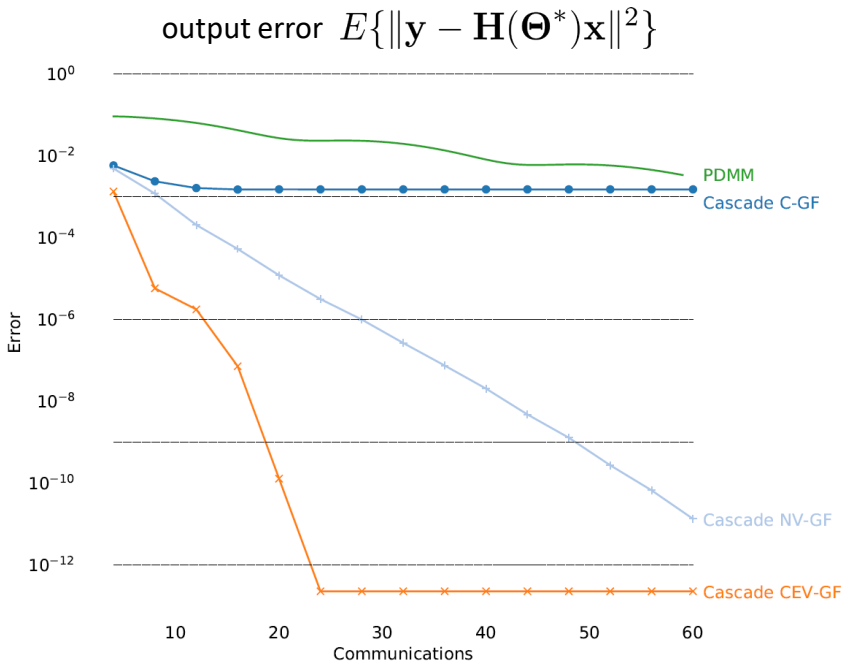
- Idea: Fit a graph filter to the operator $\tilde{\mathbf{H}}$

$$\min_{\boldsymbol{\Theta}} \|\tilde{\mathbf{H}} - \mathbf{H}(\boldsymbol{\Theta})\|_{\mathrm{F}}^2$$

Simulation results

- Graph: $N = 100$
- Algorithm: cascaded implementation of different filter types





Other activities at SPS

- Data-driven topology identification
  - Estimating the graph from data
  - On-line adaptive topology identification from streaming data
- Signal processing on higher-order graphs
  - Simplicial complexes (Hodge filter)
  - Hyper graphs (Graph Volterra filter)
  - Prediction and topology identification
- Array processing for (functional) ultrasound, radar, and wireless
  - Array design and compressive sensing
  - DoA estimation and imaging algorithms