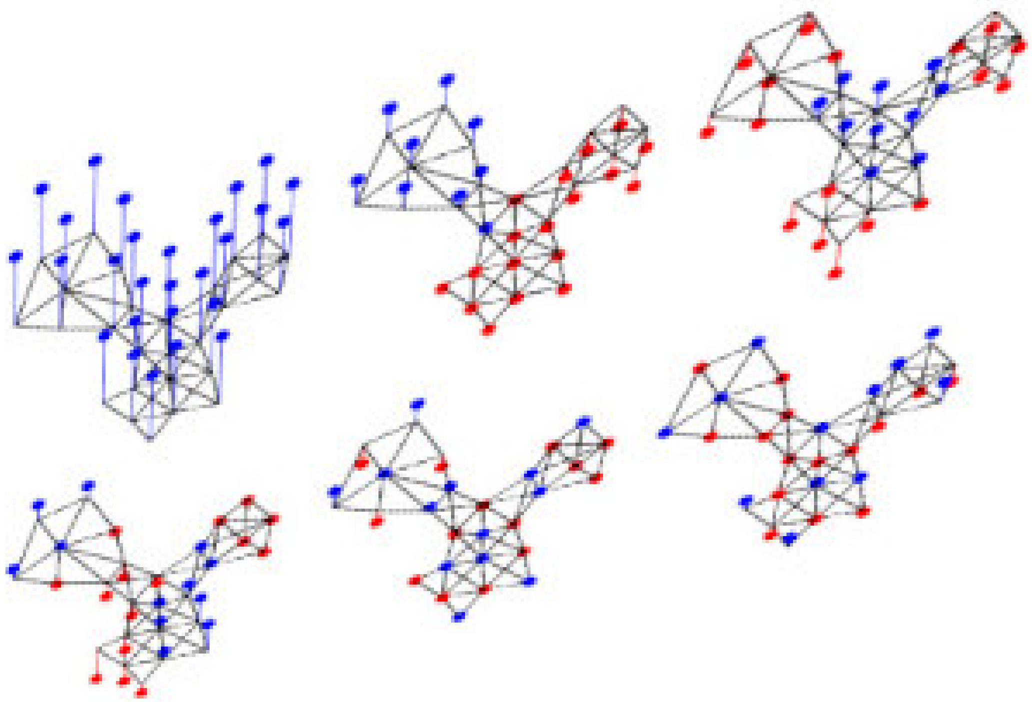
Current Approaches to Graph Signal Processing



Pavel Loskot

pavelloskot@intl.zju.edu.cn

ZJU-UIUC INSTITUTE
Zhejiang University-University of Illinois at Urbana-Champaign Institute
浙江大学伊利诺伊大学厄巴纳香槟校区联合学院

IARIA

SIGNAL 2021: ***The Sixth International Conference on Advances in Signal, Image and Video Processing***

May 30, 2021 to June 03, 2021 - Valencia, Spain

About Me

Pavel Loskot joined the ZJU-UIUC Institute as Associate Professor in January 2021. He received his PhD degree in Wireless Communications from the University of Alberta in Canada, and the MSc and BSc degrees in Radioelectronics and Biomedical Electronics, respectively, from the Czech Technical University of Prague. He is the Senior Member of the IEEE, Fellow of the HEA in the UK, and the Recognized Research Supervisor of the UKCGE.

In past 25 years, he was involved in numerous industrial and academic collaborative projects in the Czech Republic, Finland, Canada, the UK, Turkey, and China. These projects concerned mainly wireless and optical telecommunication networks, but also genetic circuits, air transport services, and renewable energy systems. This experience allowed him to truly understand the interdisciplinary workings, and crossing the disciplines boundaries.

His current research focuses on statistical signal processing and importing methods from Telecommunication Engineering and Computer Science to model and analyze systems more efficiently and with greater information power.

Objectives

1. Survey mainstream approaches to graph signal processing

- this tutorial deliberately does not add any new ideas or contributions to existing knowledge

2. Provide a starting point for researchers wishing to explore this area

- a few key ideas are identified that are exploited in many graph signal processing problems

Outline

1. Why graph signal processing
2. Key ideas in graph signal processing
3. Survey of problems and tasks in graph signal processing
4. Recommended readings on graph signal processing

Part 1:
Why graph signal processing?

Growing Interest in Complex Systems

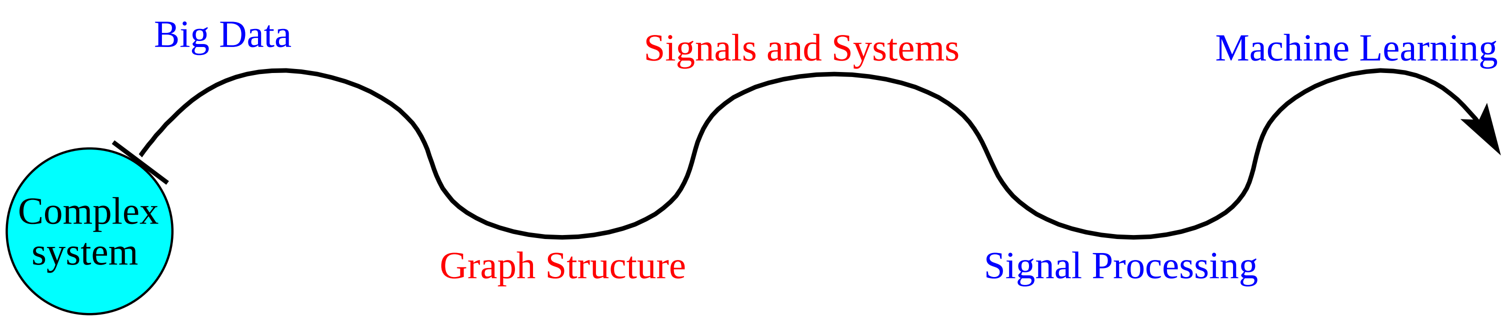
Networks seem to be suddenly appearing everywhere ...

Graphs

- mathematical models of networks
  → capture relations between pairs of objects
- many applications, well developed theory
  → Graph Theory, Network Science
  → less so in Signal Processing

Complex systems

- appear to be designed as networks
  → distributing and pooling resources
  must be a fundamental principle of Nature
- tools to work with graphs
  → understanding complex systems



Understanding graphs

Graphs as data structure

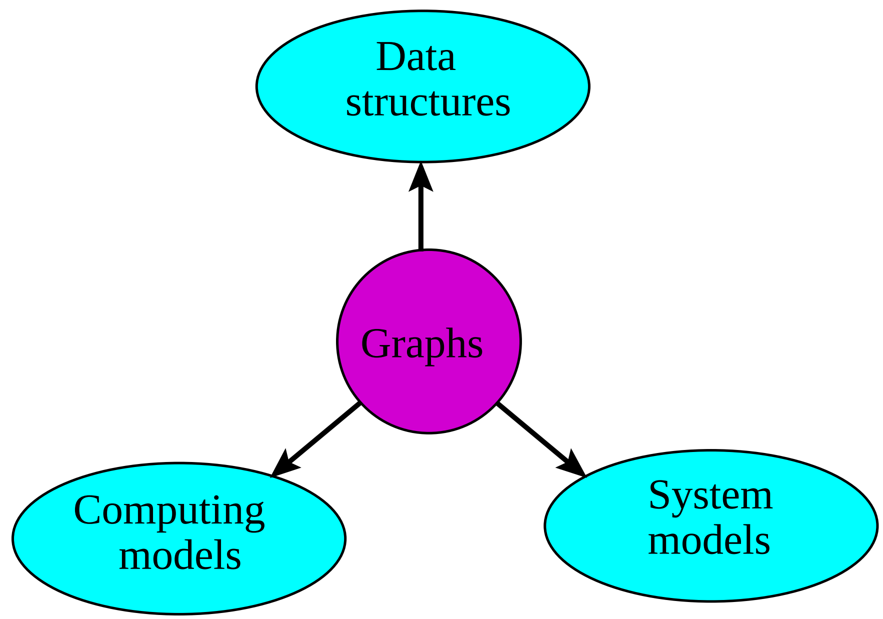
- discrete mathematical structure

Graphs as computing models

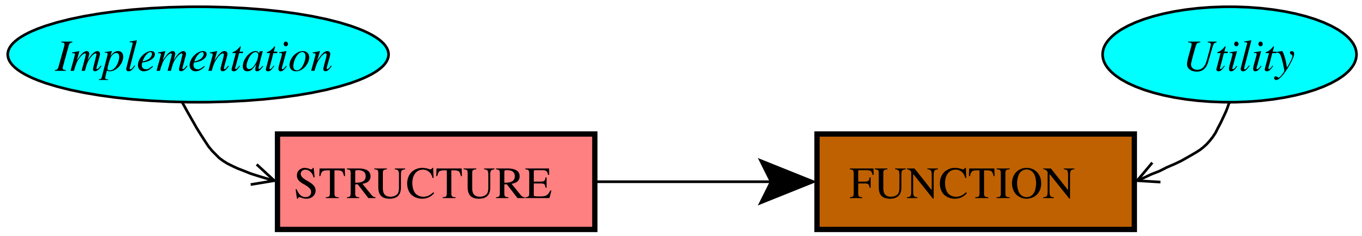
- Gaussian networks
- Bayesian networks
- Markov random fields
- hidden Markov models
- finite state machines
- data flows (Tensorflow, Spark)

Graphs as system model

- physical networks
- social networks
- flow networks
(telecommunications, transport, utilities)



Structure vs. function



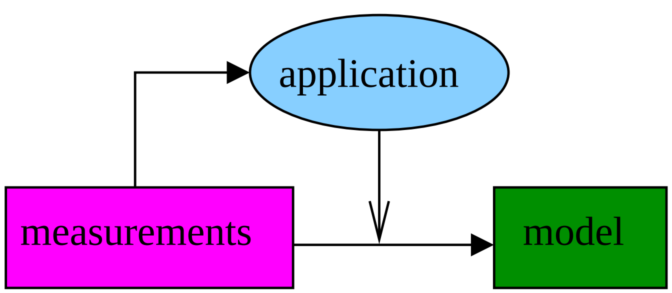
Analysis strategy

- known tuples *(structure,function)* to train ML classifier (e.g. Deep Learning)
- common in biochemistry to predict protein function
- can replace Deep Packet Inspection with cheaper and faster ML classifiers

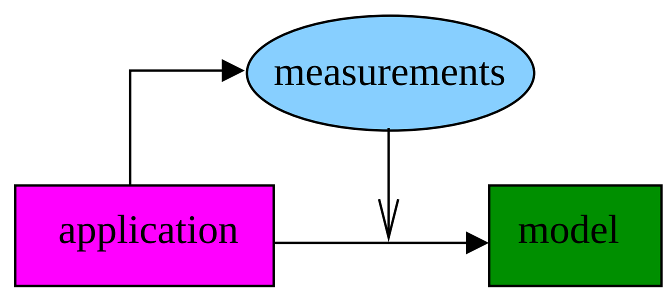
Synthesis strategy

- little explored Engineering territory

Reverse (data-driven) vs forward (application-driven) modeling

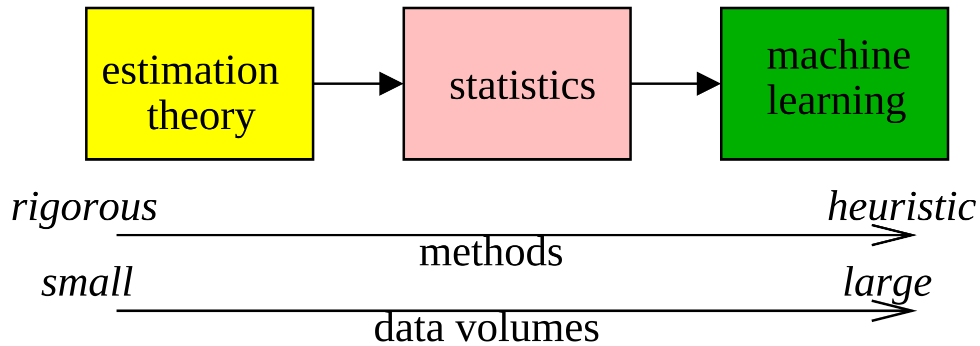


available measurements constrain possible applications



application determines required measurements

Data Processing



Signals and Systems

- concerned about creating and analyzing mathematical models

Signal processing

- models $\longleftrightarrow$ measurements (equivalence)
- numbers arranged into regular structures (scalars, vectors, matrices) $\rightarrow$ most algebraic operations defined for these regular data structures
- outcome is an algorithm

Machine Learning

- arbitrary data structures, mostly heuristic approaches
- the aim is discovery of patterns, relationships and knowledge

Computer programming

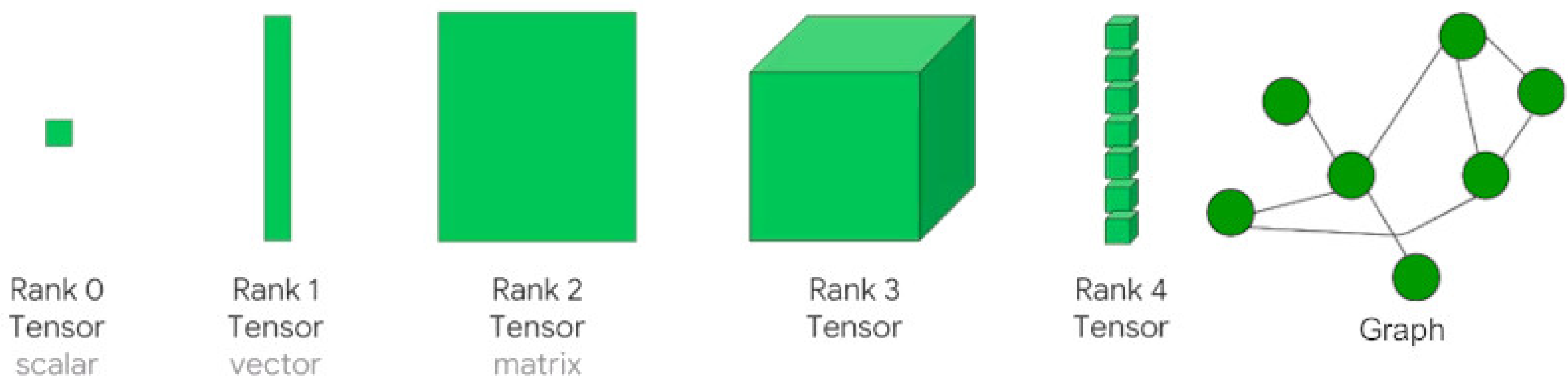
- different types of data structures and associated algorithms

Structured Signals

Sets of scalar numbers

- $V_1, V_2, V_3, \ldots$
- $V_1(t), V_2(t), V_3(t), \ldots$ scalar index $t \in \mathcal{R}$
- $V_1(\boldsymbol{x}), V_2(\boldsymbol{x}), V_3(\boldsymbol{x}), \ldots$ vector index $\boldsymbol{x} \in \mathcal{R}^K$

Discrete structures



- matrix vs. tensor: the latter has rank and additional constraints

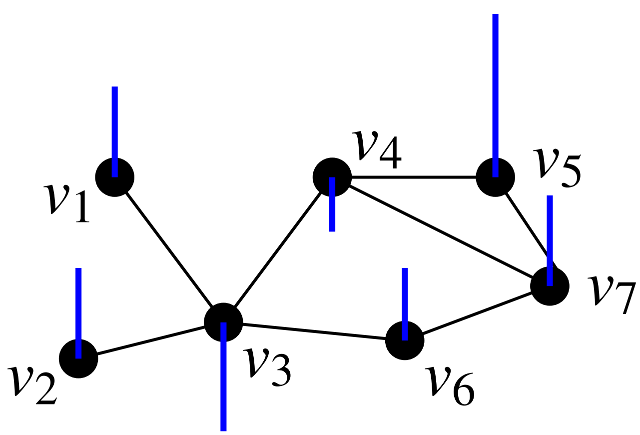
Main idea

- map scalars $V_1, V_2, V_3, \ldots$ to elements of discrete structures
- such mapping should reflect mutual relationships among $V_1, V_2, V_3, \ldots$

Graph Signals

Scalars at nodes

- example scenarios: social networks, gene interaction networks
  $\rightarrow$ nodes has properties/attributes/features
  $\rightarrow$ edges indicate pairwise relationships
- measurements at each network node $v \in V$, $|V| = N$
- graph signal: $\boldsymbol{v} = [v_1, v_2, \ldots, v_N]$ (special case, $N \rightarrow \infty$)



$$f:\ V \mapsto \mathcal{R}^{|V|}$$

Terminology

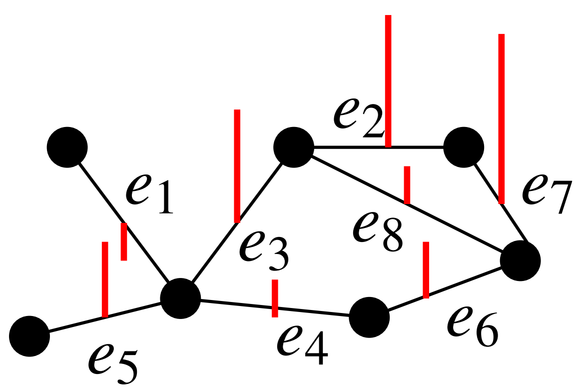
Graph	Network	System
vertex	node	component
edge	link	interaction

Graph Signals (cont.)

Scalars at edges

- example scenario: road and Internet traffic flows
  → edges has properties/attributes/features
  → edges are typically directed
- measurements at network edges $e \in E$
- graph signal: $\boldsymbol{e} = [e_1, e_2, \ldots, e_{|E|}]$

Node	Function	Flow balance
sink	absorbed flows	inflows > outflows
source	generates flows	inflows < outflows
router	mix and split flows	inflows = outflows



$$f : E \mapsto \mathcal{R}^{|E|}$$

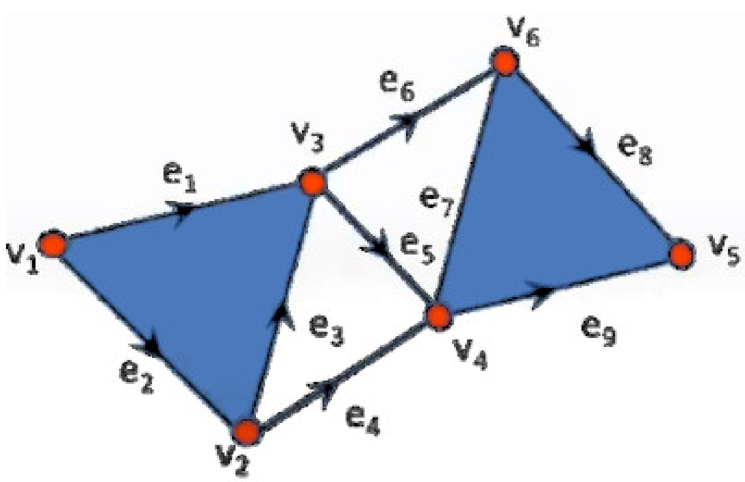
GRAPH SIGNALS (CONT.)

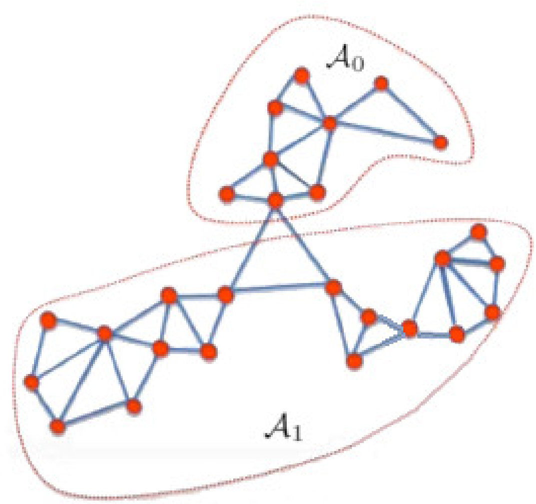
Scalars at simplexes

- k-simplex: a closed-path object with k edges and $k+1$ nodes
  → 0-simplexes are nodes
  → 1-simplexes are edges (pairwise relations or flows)
  → 2-simplexes are open/closed triangles (triplewise relations)

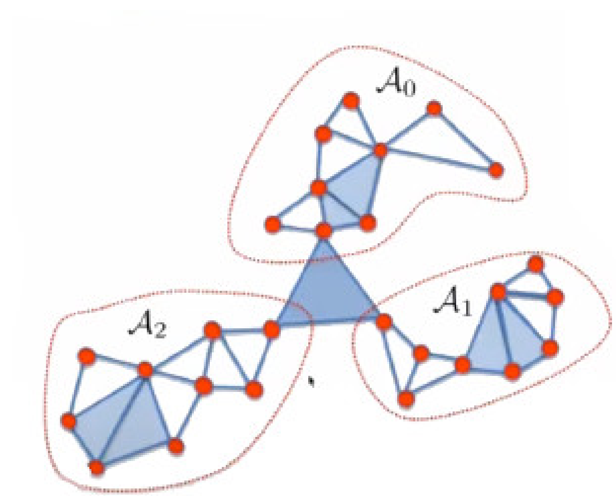
Simplical complexes of order k

- graph partitioning (a.k.a. graph cuts)
  → often aim to find the minimum cut





Simplicial complex of orde 0



Simplicial complex of order 1

Mathematical Representation of Graphs

Adjacency matrix

$$[\boldsymbol{A}]_{ij} = \begin{cases} 1 & e_{ij} = 1 \text{ (edge from node } v_i \text{ to } v_j) \\ 0 & e_{ij} = 0 \text{ (no edge between } v_i \text{ and } v_j) \end{cases}$$

Weight matrix

$$[\boldsymbol{W}]_{ij} = \begin{cases} w \in \mathcal{R} & i \neq j \text{ (implies fully connected graph)} \\ 0 & i = j \end{cases}$$

Degree matrix

$$[\boldsymbol{D}]_{ij} = \begin{cases} \deg(v_i) & i = j \\ 0 & i \neq j \end{cases}$$

Laplacian matrix

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$$

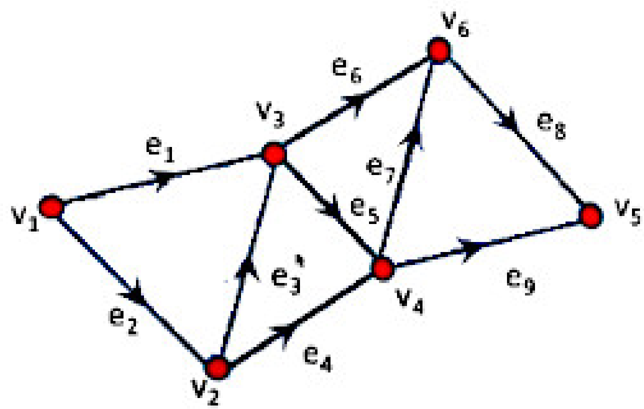
$$\boldsymbol{L} = \operatorname{diag}(\boldsymbol{W} \cdot \mathbf{1}) - \boldsymbol{W}, \quad \mathbf{1} = [1, \ldots, 1]^T$$

Mathematical Representation of Graphs (cont.)

Incidence matrix

$$[\boldsymbol{B}_1]_{ij} = \begin{cases} 1 & \text{edge } j \text{ contains node } v_i \\ 0 & \text{otherwise} \end{cases}$$

$$\boldsymbol{L} = \boldsymbol{B}_1\boldsymbol{B}_1^T, \quad \boldsymbol{A} = \boldsymbol{B}_1^T\boldsymbol{B}_1 - 2\boldsymbol{I}$$

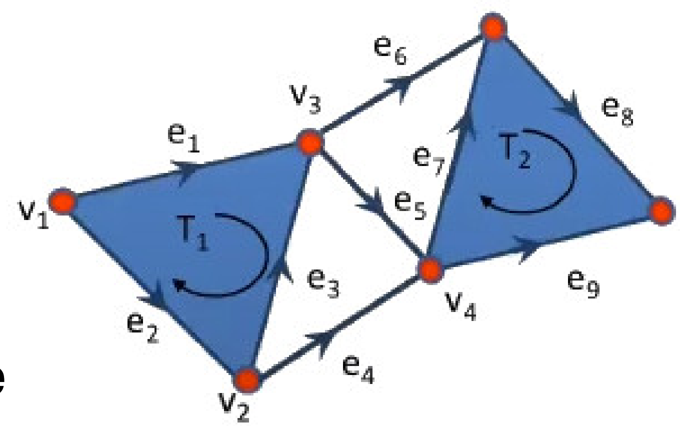


$$\begin{array}{cc} \mathbf{B}_1 = \begin{pmatrix} -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & -1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & 0 \end{pmatrix} & \text{V} \\ \text{E} & \end{array}$$

Higher-order incidence matrix

$$\boldsymbol{L}_2 = \boldsymbol{B}_2\boldsymbol{B}_2^T$$

$\rightarrow$ does it uniquely define the graph?



$$\begin{array}{cc} \mathbf{B}_2 = \begin{pmatrix} 1 & 0 \\ -1 & 0 \\ -1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & -1 \end{pmatrix} & \text{E} \\ \text{T} & \end{array}$$

Mathematical Representation of Graphs (cont.)

Undirected graphs

- $\boldsymbol{A}$, $\boldsymbol{W}$, $\boldsymbol{L}$ and $\boldsymbol{B}_1$ all uniquely define the graph
- $\boldsymbol{A}$, $\boldsymbol{W}$ and $\boldsymbol{L}$ are symmetric
- $\boldsymbol{L} \in \mathcal{R}^{N\times N}$ is positively semi-definite i.e. its SVD:

$$\boldsymbol{L} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$$

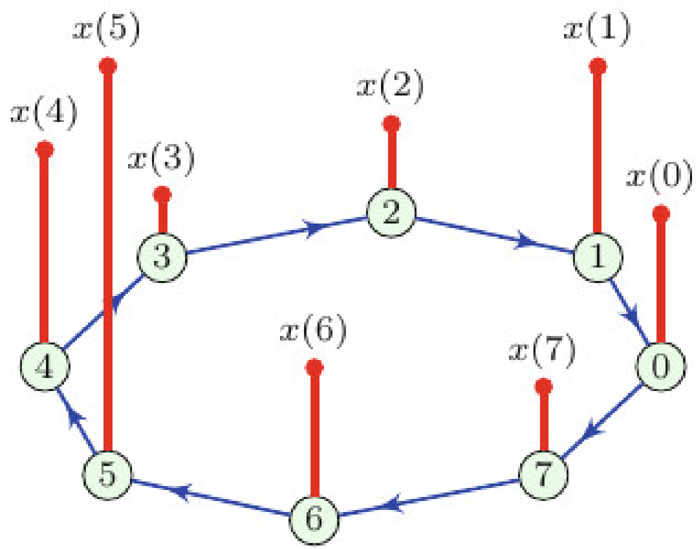
$$\begin{aligned} \boldsymbol{\Lambda} &= \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_N), \lambda_i \geq 0 \\ \boldsymbol{U}\boldsymbol{U}^T &= \boldsymbol{I}_N \text{ (identity matrix)} \end{aligned}$$

  - eigenvector $\boldsymbol{u}_i$ corresponding to eigenvalue λ_i: $\boldsymbol{L}\boldsymbol{u}_i = \lambda_i \boldsymbol{u}_i$
  - eigenvalues are obtained by solving the roots of $\det(\boldsymbol{L} - \lambda \boldsymbol{I}) = 0$
  - if $\lambda_1 < \lambda_2 < \cdots < \lambda_N$ and the graph consists of K connected components, then $\lambda_1 = \lambda_2 = \cdots = \lambda_K = 0$, and always, $\lambda_1 = 0$ and $\boldsymbol{u}_1 = [1, \ldots, 1]^T / \sqrt{N}$
- the number of walks of length K between nodes v_i and v_j is equal to $[\boldsymbol{A}^K]_{ij}$
- the number of walks of length at most K between v_i and v_j is $[\sum_{k=1}^K \boldsymbol{A}^k]_{ij}$
  $\rightarrow$ can be used to enumerate node neighbors at distance at most K steps

Part 2:

Key ideas in graph signal processing

Circular Graph



$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 1 \\ 1 & 0 & 1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & 1 & 0 & 1 & 0 \\ 0 & \cdots & 0 & 1 & 0 & 1 \\ 1 & 0 & \cdots & 0 & 1 & 0 \end{bmatrix}$$

$$\boldsymbol{A}\boldsymbol{u}_k = \lambda_k \boldsymbol{u}_k \quad \Longleftrightarrow \quad [\boldsymbol{u}_i]_{n-1} = \lambda_k [\boldsymbol{u}_k]_n \text{ (delay operator!)}$$

$$\Rightarrow \quad [\boldsymbol{u}_k]_n = \frac{1}{\sqrt{N}} \mathrm{e}^{\mathrm{j}2\pi nk/N}, \quad \lambda_k = \mathrm{e}^{-\mathrm{j}2\pi k/N} \text{ (this is DFT)}$$

Note that

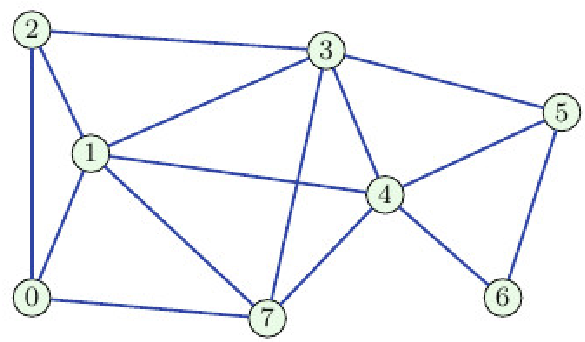
$$\boldsymbol{A}^n = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{-1}\boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{-1}\cdots\boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{-1} = \boldsymbol{U}\boldsymbol{\Lambda}^n\boldsymbol{U}^{-1}$$

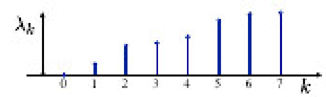
so a $(N-1)$-degree polynomial

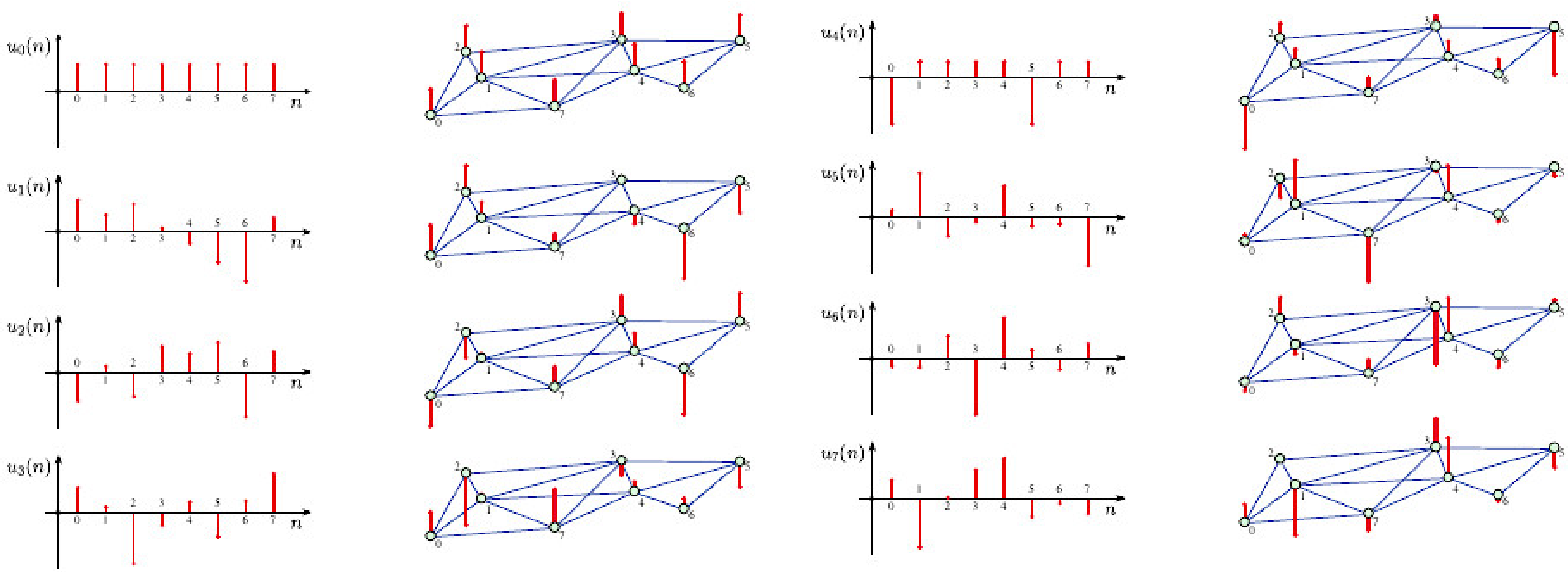
$$f(\boldsymbol{A}) = h_0\boldsymbol{A}^0 + h_1\boldsymbol{A} + \cdots + h_{N-1}\boldsymbol{A}^{N-1} = \boldsymbol{U} f(\boldsymbol{\Lambda})\boldsymbol{U}^{-1}$$

Eigen-Decomposition of a General Graph

Example







$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$$

Filtering of Graph Signals

Linear combining

$$\boldsymbol{x}(t) = \boldsymbol{A}\boldsymbol{x}(t-1), \quad vx(t-1) = [x_1(t-1), x_2(t-1), \ldots, x_N(t-1)]^T$$

i.e., if $\mathcal{N}_i$ denotes direct neighbors of node i,

$$x_i(t) = \sum_{x_j \in \mathcal{N}_i} x_j(t-1)$$

Delay of graph signal

$$\boldsymbol{x}(t) = \boldsymbol{A}\boldsymbol{x}(t-1) = \boldsymbol{A}^t\,\boldsymbol{x}(0), \quad t = 1, 2, \ldots$$

Classical linear-time invariant filter

$$y(t) = h(t) * x(t) = \sum_{k=0}^{M-1} h(k)\,x(t-k)$$

Linear-time invariant filter for graph signals

$$\boldsymbol{y}(t) = h(0)\boldsymbol{x}(t) + h(1)\boldsymbol{x}(t-1) + \cdots + h(M-1)\boldsymbol{x}(t-M+1) = \underbrace{\sum_{k=0}^{M-1} h(k)\boldsymbol{A}^{t-k}}_{H(\boldsymbol{A})}\,\boldsymbol{x}(0) = H(\boldsymbol{A})\,\boldsymbol{x}(0)$$

- but, $\boldsymbol{A}$ may not be invertible, so ‘shift’ via $\boldsymbol{A}$ cannot be undone

Fourier Transform of Graph Signals

Recall

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{-1}, \quad \boldsymbol{x} = [x_1, x_2, \ldots, x_N]^T$$

- eigenvector $\boldsymbol{u}_k$ corresponds to eigenvalue λ_k- for circular graphs,

Define

$$\hat{\boldsymbol{x}} = \boldsymbol{U}^{-1}\boldsymbol{x}, \quad \text{i.e.,} \quad \boldsymbol{x} = \boldsymbol{U}\hat{\boldsymbol{x}}, \quad \hat{\boldsymbol{x}} = [\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_N]^T$$

$$\Rightarrow \quad \hat{x}_k = \sum_{n=1}^{N} x_n [\boldsymbol{u}_k]_n \quad \text{and} \quad x_n = \sum_{k=1}^{N} \hat{x}_k [\boldsymbol{u}_k]_n$$

Polynomial filter

$$\boldsymbol{y} = H(\boldsymbol{A})\boldsymbol{x} = H(\boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{-1})\boldsymbol{x} = \boldsymbol{U}H(\boldsymbol{\Lambda})\boldsymbol{U}^{-1}\boldsymbol{x}$$

$$\Rightarrow \quad \underbrace{\boldsymbol{U}^{-1}\boldsymbol{y}}_{\hat{\boldsymbol{y}}} = H(\boldsymbol{\Lambda})\underbrace{\boldsymbol{U}^{-1}\boldsymbol{x}}_{\hat{\boldsymbol{x}}} \quad \Rightarrow \quad \hat{y}(k) = \left(h(0)\lambda_k^0 + \cdots + h(M-1)\lambda_k^{M-1}\right)\hat{x}(k)$$

Filtering of Graph Signals (cont.)

Task

- design a graph filter $\boldsymbol{h} = [h_0, h_1, \ldots, h_{M-1}]^T$ having the desired frequency response G_k at frequency λ_k, for $k = 0, 1, \ldots, N-1$

Frequency domain filtering

1. obtain GDFT of graph signal: $\hat{\boldsymbol{x}} = \boldsymbol{U}^{-1}\boldsymbol{x}$
2. frequency domain filtering: $\hat{\boldsymbol{y}} = G(\boldsymbol{\Lambda})\hat{\boldsymbol{x}}$
3. recover the output graph signal: $\boldsymbol{y} = \boldsymbol{U}\hat{\boldsymbol{y}}$

Vertex domain filtering

1. if $N \geq M$, solve the linear set of equations for filter impulse response $\boldsymbol{h}$

$$H(\lambda_k) = h_0 + h_1\lambda_k^1 + \cdots + h_{M-1}\lambda_k^{M-1} = G_k,\ \forall k = 0, 1, \ldots, N-1$$

$$\Rightarrow \quad \boldsymbol{V}_\lambda \boldsymbol{h} = \boldsymbol{G}_\lambda \quad \Rightarrow \quad \boldsymbol{h} = \boldsymbol{V}_\lambda^{-1}\boldsymbol{G}_\lambda \quad \text{(invert Vandermonde matrix)}$$

Laplacian Spectral Domain

Let

$$y = Lx \quad \leftrightarrow \quad y(t) = 2x(t) - x(t-1) - x(t+1)$$

$$L = U\Lambda U^{-1} \quad \text{and} \quad \hat{x} = U^{-1}x \quad \leftrightarrow \quad x = U\hat{x}$$

Graph signal smoothness

$$Lu = \lambda u \quad \Rightarrow \quad u^T Lu = \lambda u^T u = \lambda$$

$$u^T Lu = \frac{1}{2}\sum_{i,j=0}^{N} [A]_{i,j}\,(u(i) - u(j))^2 = \lambda$$

i.e. eigenvectors with small eigenvalues represent low-pass components of graph signal x

Ideal low-pass filter

$$H(\lambda) = \begin{cases} 1 & \forall \lambda < \lambda_c \\ 0 & \text{otherwise} \end{cases} \qquad \text{(frequency response)}$$

General filter

$$y = \sum_{m=0}^{M-1} h_m L^m x$$

Fourier Analysis of Graph Signals

Attempted definitions of

- convolution in vertex and spectral domains
- Z-transform for graph signals
- Parseval's theorem for graph signals
- shift in frequency domain

... often not intuitive extensions, but more '*can be considered as*'

Other weaknesses

- signal shift using $\boldsymbol{A}$ or $\boldsymbol{L}$ may not be invertible
- since graph signal is non-periodic (what would a periodic extension be?), Discrete-time Fourier Transform and not DFT should be assumed
- sampling should make the spectrum periodic

... indicative that other different approaches should be attempted

Smoothing Filters of Graph Signals

De-noising application

- input: noisy graph signal $\boldsymbol{x}$
- output: smoothed output graph signal $\boldsymbol{y}$
- solution:

$$\min J = \frac{1}{2}||\boldsymbol{y}-\boldsymbol{x}||_2^2 + \alpha \boldsymbol{y}^T \boldsymbol{L} \boldsymbol{y}$$

$$\Rightarrow \quad \frac{\partial}{\partial \boldsymbol{y}^T} J \overset{!}{=} 0 \quad \Rightarrow \quad \boldsymbol{y} = (\mathbf{1} + 2\alpha \boldsymbol{L})^{-1}\, \boldsymbol{x} \quad \leftrightarrow \quad \hat{\boldsymbol{y}} = \underbrace{(\mathbf{1} + 2\alpha \boldsymbol{\Lambda})^{-1}}_{H(\boldsymbol{\Lambda})}\, \hat{\boldsymbol{x}}$$

- alternative formulation:

$$\min J = \frac{1}{2}||\boldsymbol{y}-\boldsymbol{x}||_2^2 + \alpha \boldsymbol{y}^T \boldsymbol{L} \boldsymbol{y} + \beta \underbrace{\boldsymbol{y}^T \boldsymbol{L}^2 \boldsymbol{y}}_{||\boldsymbol{L}\boldsymbol{y}||_2^2}$$

$$\Rightarrow \quad \frac{\partial}{\partial \boldsymbol{y}^T} J \overset{!}{=} 0 \quad \Rightarrow \quad \boldsymbol{y} = \left(\mathbf{1} + 2\alpha \boldsymbol{L} + 2\beta \boldsymbol{L}^2\right)^{-1} \boldsymbol{x}$$

- it is also possible to consider sparsity instead of smoothness

Graph Sampling and Compression

Task

- represent or reconstruct graph signal from small number of its samples

K-sparse graph signal

$$\hat{\boldsymbol{x}} = [\hat{x}(0), \hat{x}(1), \ldots, \hat{x}(K-1), \underbrace{0, \ldots, 0}_{N-K}\,]^T \quad \Rightarrow \quad x(n) = \sum_{k=0}^{K-1} \hat{x}(k)\,\boldsymbol{u}_k(n)$$

- assume M samples available at nodes $n_1, n_2, \ldots, n_{n_M}$, $K \leq M < N$

$$\underbrace{\begin{bmatrix} y(n_1) \\ y(n_2) \\ \vdots \\ y(n_M) \end{bmatrix}}_{\boldsymbol{y}\ \text{(measurements)}} = \underbrace{\begin{bmatrix} \boldsymbol{u}_0(n_1) & \boldsymbol{u}_1(n_1) & \cdots & \boldsymbol{u}_{N-1}(n_1) \\ \boldsymbol{u}_0(n_2) & \boldsymbol{u}_1(n_2) & \cdots & \boldsymbol{u}_{N-1}(n_2) \\ \vdots & & & \vdots \\ \boldsymbol{u}_0(n_M) & \boldsymbol{u}_1(n_M) & \cdots & \boldsymbol{u}_{N-1}(n_M) \end{bmatrix}}_{\boldsymbol{M}\ \text{(measurement matrix)}} \underbrace{\begin{bmatrix} \hat{x}(0) \\ \hat{x}(1) \\ \vdots \\ \hat{x}(K-1) \end{bmatrix}}_{\hat{\boldsymbol{x}}_{(K)}}$$

- recovery from $M > K$ samples

$$\hat{\boldsymbol{x}}_{(K)} = (\boldsymbol{M}^T\boldsymbol{M})^{-1}\boldsymbol{M}^T\boldsymbol{y} \quad \Rightarrow \quad \boldsymbol{x} = \boldsymbol{U}\,[\hat{\boldsymbol{x}}_{(K)}^T, 0\cdots 0]^T$$

- this procedure can be used if positions of $(N-K)$ zeros in $\hat{\boldsymbol{x}}$ are known

Graph Sampling and Compression (cont.)

K-sparse graph signal

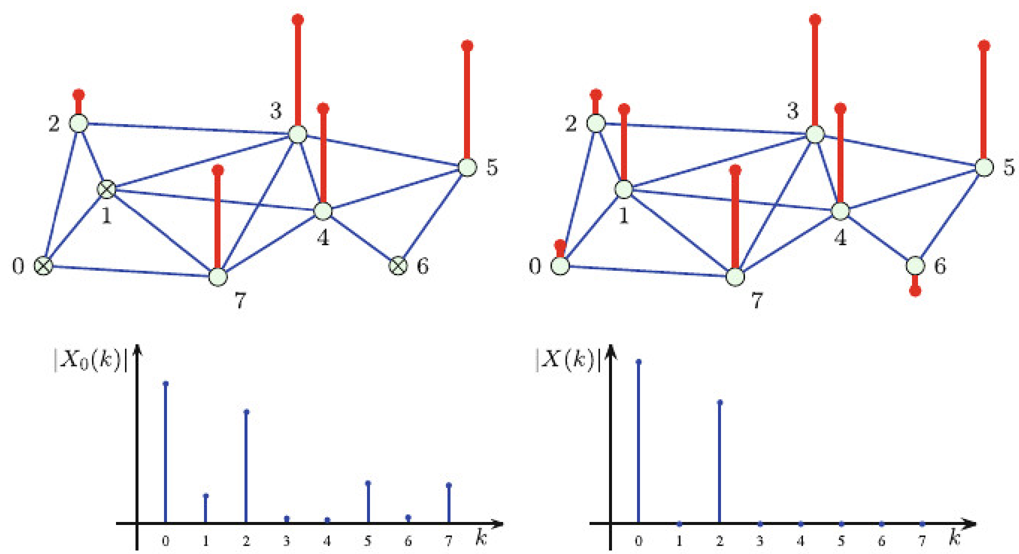
- neither positions nor the number of zeros in $\hat{\boldsymbol{x}}$ is known
- reconstruction:

$$\min \|\boldsymbol{X}\|_0 \quad s.t. \quad \boldsymbol{y} = \boldsymbol{M}\boldsymbol{X}$$

- possible solution:
  1. estimate zeros and their positions as $(N-K)$ largest values in $\boldsymbol{X} = \boldsymbol{M}^T\boldsymbol{y}$
  2. knowing K zero positions, use previous procedure to estimate $\boldsymbol{x}$

Example

- signal measured at vertices $2,3,4,5,7$: $\boldsymbol{y} = [0.224, 1.206, 1.067, 1.285, 1.116]$
- it is estimated that there are $(N-K) = 2$ non-zero components in $\boldsymbol{X} = \boldsymbol{M}^T\boldsymbol{y}$



Graph Sampling and Compression (cont.)

Aggregate sampling

- measurements using combining matrix $\boldsymbol{C}$

$$\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} = \underbrace{\boldsymbol{C}\boldsymbol{U}}_{\boldsymbol{M}}\,\hat{\boldsymbol{x}}$$

- reconstruction: see previous methods
- can assume sampling at one vertex only

$$y_0 = [\boldsymbol{x}]_n, \quad y_1 = [\boldsymbol{A}\boldsymbol{x}]_n, \quad \ldots, \quad y_{N-1} = [\boldsymbol{A}^{N-1}\boldsymbol{x}]_n$$

then

$$\boldsymbol{C} = \begin{bmatrix} 0 \cdots 1 \cdots 0 \\ \mathrm{row}_n(\boldsymbol{A}) \\ \vdots \\ \mathrm{row}_n\left(\boldsymbol{A}^{N-1}\right) \end{bmatrix}$$

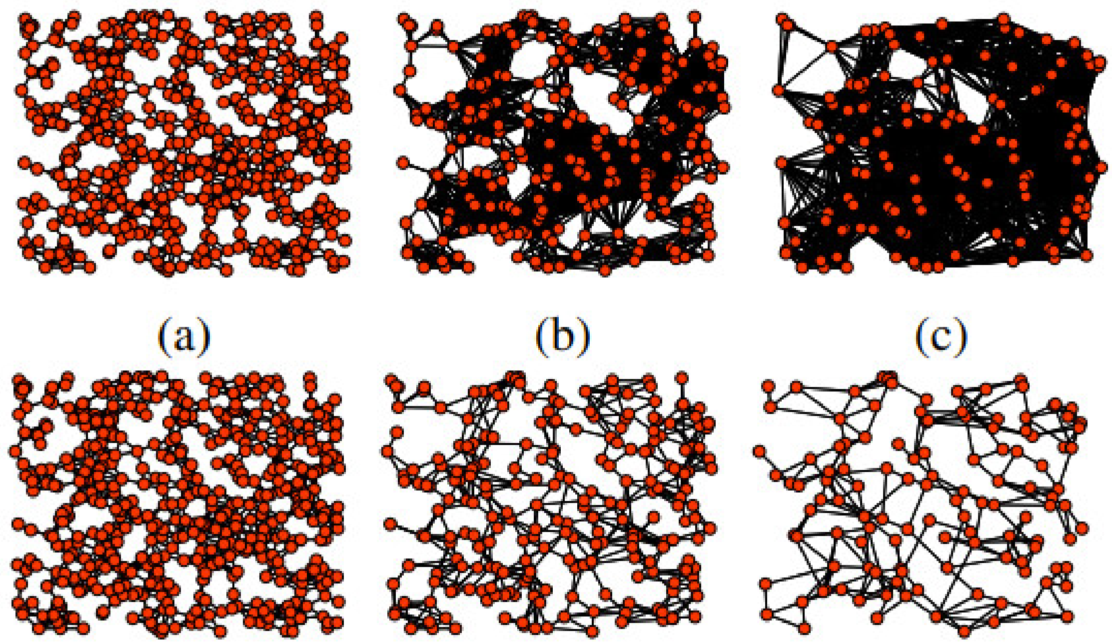
Graph Sampling and Compression (cont.)

Task

- sub-sample large graph signals to reduce storage requirements and data processing complexity
- downsampled graph to retain certain similarity to original graph
  → connectivity distribution
  → spectral properties

Methods

- random selection of nodes (equi-probable, proportional to the node degrees or ranks) is satisfactory in very large graphs
- random selection of edges tends to produce disconnected graphs
- random walk and maximum spanning tree based
- many other methods in literature



(a) (b) (c)
(d) (e) (f)

Incorporation of a spectral sparsification step into the graph reduction. (a)-(c) Repeated largest eigenvector downsampling and Kron reduction of a sensor network graph. (d)-(f)

Flows on Graphs

Diffusion

$$\frac{\partial}{\partial t}\boldsymbol{x}(t) = -\boldsymbol{L}\boldsymbol{x}(t)$$

- backward difference approximation

$$\boldsymbol{x}(t+1) - \boldsymbol{x}(t) = -\alpha\boldsymbol{L}\boldsymbol{x}(t+1) \quad \Rightarrow \quad \boldsymbol{x}(t+1) = (\boldsymbol{I}+\alpha\boldsymbol{L})^{-1}\boldsymbol{x}(t),\ t = 0,1,\ldots$$

- forward difference approximation

$$\boldsymbol{x}(t+1) - \boldsymbol{x}(t) = -\alpha\boldsymbol{L}\boldsymbol{x}(t) \quad \Rightarrow \quad \boldsymbol{x}(t+1) = (\boldsymbol{I}-\alpha\boldsymbol{L})\boldsymbol{x}(t),\ t = 0,1,\ldots$$

Alternative interpretation

- minimize the smoothness of $\boldsymbol{x}$

$$\min_{\boldsymbol{x}} \boldsymbol{x}^T\boldsymbol{L}\boldsymbol{x} \quad \Rightarrow \quad \frac{\partial}{\partial \boldsymbol{x}}\boldsymbol{x}^T\boldsymbol{L}\boldsymbol{x} = 2\boldsymbol{x}^T\boldsymbol{L}$$

$$\Rightarrow \quad \boldsymbol{x}(t+1) = \boldsymbol{x}(t) - \alpha\boldsymbol{L}\boldsymbol{x}(t) = (\boldsymbol{I}-\alpha\boldsymbol{L})\boldsymbol{x}(t) \quad \text{(steepest descent)}$$
$$\Rightarrow \quad \hat{\boldsymbol{x}}(t+1) = (\boldsymbol{I}-\alpha\boldsymbol{\Lambda})\hat{\boldsymbol{x}}(t) \quad \text{(spectral domain)}$$

Building Graph Models

Graph model

- what do node represent?
- when to add edges between nodes?

Node modeling

- obvious cases: social networks, agent networks, routers
- less obvious cases: time series data, complex systems
- challenges:
  → merging/clustering nodes
  → time-varying scenarios

Edge modeling

- obvious cases: social networks, flow networks
- challenges in other cases:
  → weight metric selection
  → no edge can also mean missing data
  → time-varying scenarios

Building Graph Models (cont.)

Euclidean distance metric

- used when mutual geographical location $\boldsymbol{r}$ of nodes matters, i.e., the distance $r_{nm} = ||\boldsymbol{r}_n - \boldsymbol{r}_m||_2$
- edge weight metrics with parameters $\alpha \geq 0$, $r_0 > 0$ and $c \geq 0$

$$W_{nm} = \begin{cases} \mathrm{e}^{-\alpha\,(r_{nm})^c} & r_{nm} \leq r_0 \\ 0 & r_{nm} > r_0 \end{cases}$$

- these weights can produce exponentially weighted moving average of neighboring nodes for linear graph model

$$\boldsymbol{x}(t+1) = (\boldsymbol{I} + \boldsymbol{W})\,\boldsymbol{x}(t)$$

Other edge metrics

- any similarity metric can be considered
- total variance, correlation (Pearson, Spearman)
- binary (relationship exists/does not exist)
- learning/extrapolating relationships and weights

Learning Graphs from Data

Task

- given measurements $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, ..., $\boldsymbol{x}_N$ at N nodes, construct a graph representing the data
- exploit smoothness of the graph signal rather than internal correlations

Define

- data matrix $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N]$
- graph signal smoothness

$$\mathrm{tr}\left\{\boldsymbol{X}^T \boldsymbol{L} \boldsymbol{X}\right\} = \frac{1}{2}\sum_{n,m}[\boldsymbol{W}]_{n,m}\|\boldsymbol{x}_n - \boldsymbol{x}_m\|^2 = \frac{1}{2}\mathrm{tr}\{\boldsymbol{W}\boldsymbol{\Delta X}\}, \quad [\boldsymbol{\Delta X}]_{n,m} = \|\boldsymbol{x}_n - \boldsymbol{x}_m\|^2$$

Optimization problem

$$\min_{\boldsymbol{W}} \ \mathrm{tr}\{\boldsymbol{W}\boldsymbol{\Delta X}\} + \alpha h(\boldsymbol{W}) \quad \textit{s.t.} \quad \mathrm{diag}(\boldsymbol{W}) = \boldsymbol{0},\ \boldsymbol{W} = \boldsymbol{W}^T$$

- regularization $h(\boldsymbol{W})$: $\|\boldsymbol{W}\|_1$, $\|\boldsymbol{W}\|_2^2$, $\log\det \boldsymbol{W}$
- additional constraint $\mathrm{diag}(\boldsymbol{W}\boldsymbol{1}) = \boldsymbol{I}$ (weights normalization), or as objective

$$\min_{\boldsymbol{W}} \ \mathrm{tr}\{\boldsymbol{W}\boldsymbol{\Delta X}\} + \alpha h(\boldsymbol{W}) - \beta \boldsymbol{1}^T \log \boldsymbol{W}\boldsymbol{1} \quad \textit{s.t.} \quad \mathrm{diag}(\boldsymbol{W}) = \boldsymbol{0},\ \boldsymbol{W} = \boldsymbol{W}^T$$

Learning Graphs from Data (cont.)

Noisy data

- learn the graph *and* de-noise the measurements
- optimization problem

$$\min_{\boldsymbol{W},\boldsymbol{Y}} \|\boldsymbol{Y}-\boldsymbol{X}\|_2^2+\alpha\,\mathrm{tr}\{\boldsymbol{W}\boldsymbol{\Delta}\boldsymbol{Y}\}+\beta\,h(\boldsymbol{W}) \quad \textit{s.t.}\ \mathrm{diag}(\boldsymbol{W})=\boldsymbol{0},\ \boldsymbol{W}=\boldsymbol{W}^T,\ [\boldsymbol{\Delta}\boldsymbol{Y}]_{n,m}=\left\|\boldsymbol{y}_n-\boldsymbol{y}_m\right\|^2$$

Gaussian assumption

- graph signal as Gaussian-Markov random field
- optimization problem (gdet is generalized determinant) is defined as maximum-likelihood estimation of $\boldsymbol{L}$

$$\max_{\boldsymbol{L}} \log\mathrm{gdet}\,\boldsymbol{L}-\mathrm{tr}\{\boldsymbol{C}\boldsymbol{L}\}-\alpha h(\boldsymbol{L}) \quad \textit{s.t.} \quad \boldsymbol{L}\boldsymbol{1}=\boldsymbol{0},\ \boldsymbol{L}=\boldsymbol{L}^T$$

where $\boldsymbol{C}$ is estimated covariance matrix $\boldsymbol{C}=\frac{1}{n-1}\sum_{i=1}^n \mathrm{col}_i\left(\boldsymbol{X}-\bar{\boldsymbol{X}}\right)\mathrm{col}_i\left(\boldsymbol{X}-\bar{\boldsymbol{X}}\right)^T$

$$\mathrm{tr}\{\boldsymbol{C}\boldsymbol{L}\}\ \propto\ \mathrm{tr}\left\{\boldsymbol{X}^T\boldsymbol{L}\boldsymbol{X}\right\}$$

Note

- these methods pose no constraints on graph structure except the number of graph nodes

Learning Graphs from Data (cont.)

Learning specific graph structure

- grid, tree, bipartite, multi-component, regular, densely connected etc.
- challenge is how to specify the graph structure as optimization constraint

Strategy

- spectral constraints on eigenvalues or eigenvectors of $\boldsymbol{L}$ or $\boldsymbol{W}$

$$\boldsymbol{\Lambda}(\boldsymbol{L}) \in \mathcal{S}_\lambda \quad \text{or} \quad \boldsymbol{\Lambda}(\boldsymbol{W}) \in \mathcal{S}_\lambda$$

- algorithms for such non-convex problems were developed in literature

Example: k-component graph

- learn k clusters, i.e., $\lambda_1 = \cdots = \lambda_k = 0$, and additional constraint

$$\Rightarrow \mathrm{rank}(\boldsymbol{L}) = N - k \quad \Leftrightarrow \quad \sum_{i=1}^{k} \lambda_i(\boldsymbol{L}) = 0$$

- unforeseen problem: k-component graph learning generates isolated nodes
  $\rightarrow$ add constraint(s) to avoid zero-degree nodes, e.g. $\mathrm{diag}(\boldsymbol{W}\mathbf{1}) = \boldsymbol{I}$

Learning Graphs from Data (cont.)

Learning correlation graph

- for $n = 1, 2, \ldots, N$, sequentially determine the edge weights W_{nm}

$$\min_{\mathrm{row}_n(\boldsymbol{W})} \left\| \boldsymbol{x}_n - \sum_{\substack{m=1 \\ m \neq n}}^{N} W_{nm} \boldsymbol{x}_m \right\|_2^2 + \alpha \, \|\mathrm{row}_n(\boldsymbol{W})\|_1 \quad \text{(LASSO minimization)}$$

LASSO minimization

$$\min_{\boldsymbol{x}} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 + \alpha \, \|\boldsymbol{x}\|_1$$

is solved iteratively as

$$\boldsymbol{x}_{k+1} = \mathrm{soft}\left(2\beta \boldsymbol{A}^T (\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}_k) + \boldsymbol{x}_k, \beta\alpha\right), \; k = 1, 2, \ldots$$

where the learning step β

$$0 < \beta < 1/(2\lambda_{\max}), \; \lambda_{\max} \text{ is max eigenvalue of } \boldsymbol{A}^T \boldsymbol{A}$$

and

$$\mathrm{soft}(y, u) = \begin{cases} y + u, & y < -u \\ 0, & |y| \leq u \\ y - u, & y > u \end{cases}$$

Learning Graphs from Data (cont.)

Non-Gaussian data

- tweak $\mathrm{tr}\{\boldsymbol{C}\boldsymbol{L}\}$ term in objective function, e.g. majorization-minimization iterations [Palomar et al. 2017]

$$\mathrm{tr}\left\{\boldsymbol{C}^k\boldsymbol{L}\right\} \text{ where } \boldsymbol{C}^k = \frac{1}{n}\sum_{i=1}^{n} \frac{\mathrm{col}_i(\boldsymbol{X})\,\mathrm{col}_i(\boldsymbol{X})^T}{\mathrm{col}_i(\boldsymbol{X})^T\,\boldsymbol{L}^k\,\mathrm{col}_i(\boldsymbol{X})},\ k = 1,2,\ldots$$

Time-varying data

- divide data to T chunks, estimate graph $\boldsymbol{L}_t$ for every data chunk $t = 1,2,\ldots,T$
- add regularization term $\sum_{t=2}^{T} d(\boldsymbol{L}_t, \boldsymbol{L}_{t-1})$ to objective function to maintain graph consistency in time, e.g.

$$d(\boldsymbol{L}_t, \boldsymbol{L}_{t-1}) = \|\boldsymbol{L}_t - \boldsymbol{L}_{t-1}\|_2^2$$

- $\boldsymbol{L}_t$ can be estimated sequentially to reduce complexity

Spectral Clustering

Task

- partition the graph in K clusters of approximately equal size
- edges between clusters should have small weights

Solution

- any clustering algorithm e.g. K-means

Define graph cut

- graph weights W_{ij} between nodes $v_i \in V$ and $v_j \in V$

$$V_1 \cup V_1^c = V,\ V_1 \cap V_1^c = \emptyset \quad \Rightarrow \quad \mathrm{cut}(V_1, V_1^c) = \sum_{i \in V_1, j \in V_1^c} W_{ij} \quad \text{(other operators: avg, min)}$$

- for K disjoint node subsets

$$\cup_{k=1}^{K} V_k = V,\ V_k \cap V_l = \emptyset\ \forall k, l \quad \Rightarrow \quad \mathrm{cut}(V_1, \ldots, V_K) = \sum_{k=1}^{K} \mathrm{cut}\left(V_k, V_k^c\right)$$

- normalization to encourage equal-size clusters

$$\Rightarrow \quad \mathrm{cut}(V_1, \ldots, V_K) = \sum_{k=1}^{K} \frac{\mathrm{cut}\left(V_k, V_k^c\right)}{|V_k|}$$

Spectral Clustering (cont.)

Optimum clustering via graph signal (case $K = 2$)

given V_1, let $a = \sqrt{\frac{|V_1^c|}{|V_1|}}$, and define graph signal $f(v_i) = f_i = \begin{cases} a & v_i \in V_1 \\ 1/a & v_i \in V_1^c \end{cases}$

the smoothness, $\boldsymbol{f}^T \boldsymbol{L} \boldsymbol{f} = \frac{1}{2} \sum_{i,j=1}^{N} W_{ij} (f_i - f_j)^2 = \frac{N}{2} \sum_{k=1}^{2} \frac{\mathrm{cut}\left(V_k, V_k^c\right)}{|V_k|}$

$$\Rightarrow \quad \min_{V_1 \subset V} \boldsymbol{f}^T \boldsymbol{L} \boldsymbol{f} \quad \textit{s.t.} \quad \boldsymbol{f}^T \mathbf{1} = 0,\ \|\boldsymbol{f}\|_2^2 = N \quad \text{(NP-hard problem)}$$

Applications of graph clustering

- graph partitioning
  → graph signals often smooth within clusters
  → nodal domain of f is a sub-graph where f in all nodes have the same sign
- hierarchical graph representation
- graph reduction
  → replace small clusters with single node
- graph visualization
- graph coloring

Graph Downsampling

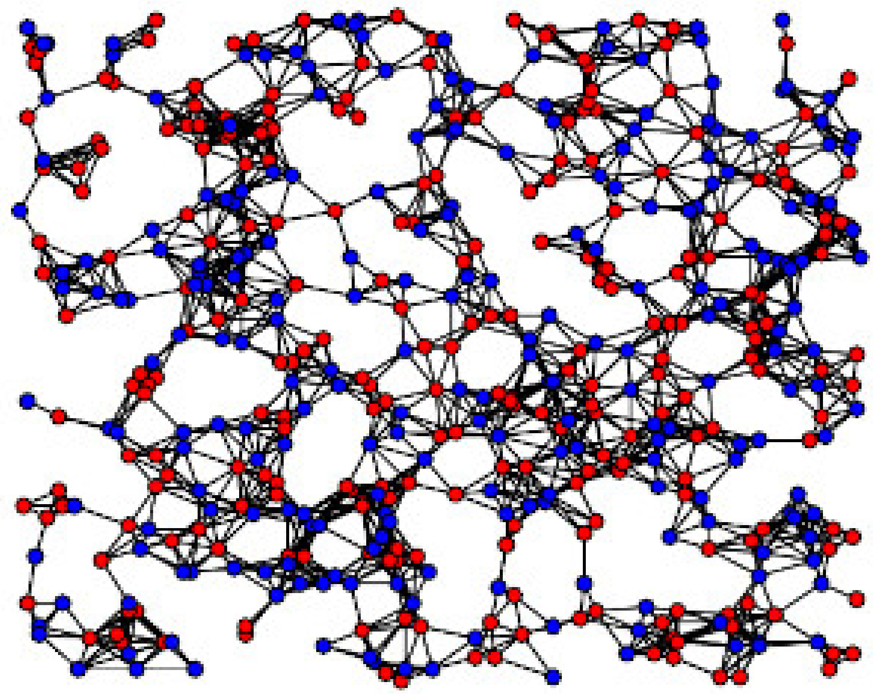
Task

- remove less important nodes and/or edges
  $\rightarrow$ how to select those?
  $\rightarrow$ e.g. account for edge weights and node degrees
- aim: reduce the graph size by half

Solution using the largest eigenvector

- let $\boldsymbol{u}_{\max}$ be the eigenvector of $\boldsymbol{L}$ corresponding to eigenvalue $\lambda_{\max}$

$$V_{\text{keep}} = \left\{ v_i \in V : \begin{cases} \boldsymbol{u}_{\max}(i) \geq 0 \\ \boldsymbol{u}_{\max}(i) \leq 0 \\ N/2\text{-largest } |\boldsymbol{u}_{\max}(i)| \end{cases} \right\}$$



- note that V_{keep} and V_{keep}^c represents the graph cut

Random Signals on Graphs

Stationary graph signal

- can be expressed as

$$\boldsymbol{x} = \underbrace{\boldsymbol{x}_0}_{\text{deterministic}} + \underbrace{\sum_{l=0}^{N-1} h_l \boldsymbol{L}^l}_{\boldsymbol{H}} \boldsymbol{w}$$

  where $\boldsymbol{w}$ are zero-mean uncorrelated random samples
- corresponding covariance $\boldsymbol{C}_{\boldsymbol{x}} = \mathrm{E}\left[\boldsymbol{x}\boldsymbol{x}^T\right] = \boldsymbol{H}\boldsymbol{H}^T$
- if $\boldsymbol{L} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{L}^T$, then $\boldsymbol{s} = \mathrm{diag}\left(\boldsymbol{U}^T\boldsymbol{C}_{\boldsymbol{x}}\boldsymbol{U}\right)$ is power spectral density vector

Non-stationary graph signal

- $\boldsymbol{w}$ does not need to be white i.e. $\boldsymbol{C}_{\boldsymbol{w}} = \mathrm{E}\left[\boldsymbol{w}\boldsymbol{w}^T\right] \neq \boldsymbol{I}$
- consequently, $\boldsymbol{U}^T\boldsymbol{C}_{\boldsymbol{x}}\boldsymbol{U}$ is not a diagonal matrix
- one strategy: if $\boldsymbol{C}_{\boldsymbol{w}}$ is known, find $\boldsymbol{H}$, so that $\hat{\boldsymbol{C}}_{\boldsymbol{x}} = \frac{1}{M}\sum_{m=1}^{M} \boldsymbol{x}_m\boldsymbol{x}_m^T$ and $\boldsymbol{C}_{\boldsymbol{x}} = \mathrm{E}\left[\boldsymbol{x}\boldsymbol{x}^T\right] = \boldsymbol{H}\boldsymbol{C}_{\boldsymbol{w}}\boldsymbol{H}^T$ are close in some sense assuming symmetry $\boldsymbol{H} = \boldsymbol{H}^T$

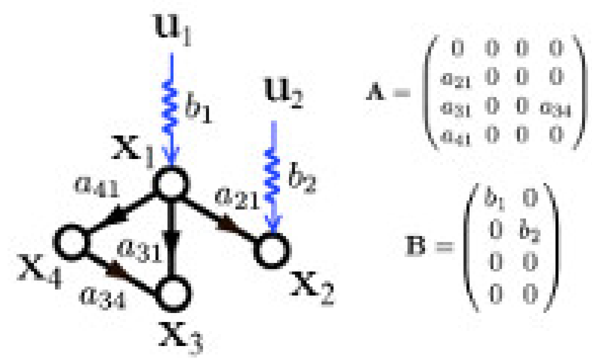
Graph observability

Linear graph

- $a_{ij}(t)$ is edge weight between nodes i and j
- x_i is weight of node i, so the internal state $\boldsymbol{x}(t) = [x_1(t), \ldots, x_n(t)]^T$
- if time-invariant, state-space description

$$
\begin{aligned}
\dot{\boldsymbol{x}}(t) &= \boldsymbol{A}\,\boldsymbol{x}(t) + \boldsymbol{B}\,\boldsymbol{u}(t) \\
\boldsymbol{y}(t) &= \boldsymbol{C}\,\boldsymbol{x}(t)
\end{aligned}
$$

... this is a linear MIMO system!



Observability

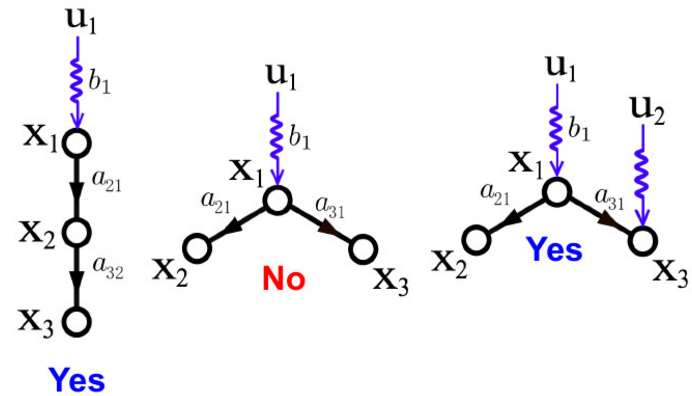
- a state (values of internal variables) can be obtained from finite observations
  $\rightarrow$ find state trajectory $\boldsymbol{x}(t)$ from any initial state $\boldsymbol{x}(0)$ to the current state
- strongly connected components (there is path between any two nodes)
  $\rightarrow$ have to observe at least one node from each SCC
- for linear model, can use maximum matching to find minimum # sensors
  $\rightarrow$ often much larger than # SCC (due to model symmetries)
- surprisingly, for non-linear model, sensors predicted by SCCs are necessary as well as sufficient (since model symmetries are rare)

Graph controlability

Kalman's condition

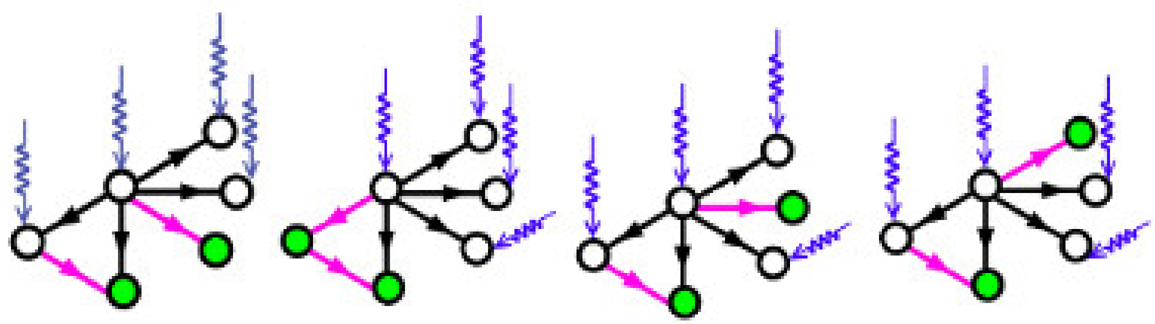
- controlability matrix $\boldsymbol{D}$ must be full rank i.e. $\text{rank}\boldsymbol{D} = n$ where $n = |\boldsymbol{x}|$

$$\boldsymbol{D} = [\,\boldsymbol{B}, \boldsymbol{A}\boldsymbol{B}, \boldsymbol{A}^2\boldsymbol{B}, \ldots, \boldsymbol{A}^{n-1}\boldsymbol{B}\,]$$



Driver nodes

- for linear networks, maximum matching again helps



- surprisingly, driver nodes tend to avoid hubs
  → average degree of driver nodes is smaller than average degree of graph
  → # driver nodes mainly determined by degree distribution

"Sparse and heterogeneous networks are harder to control than dense and homogeneous networks."

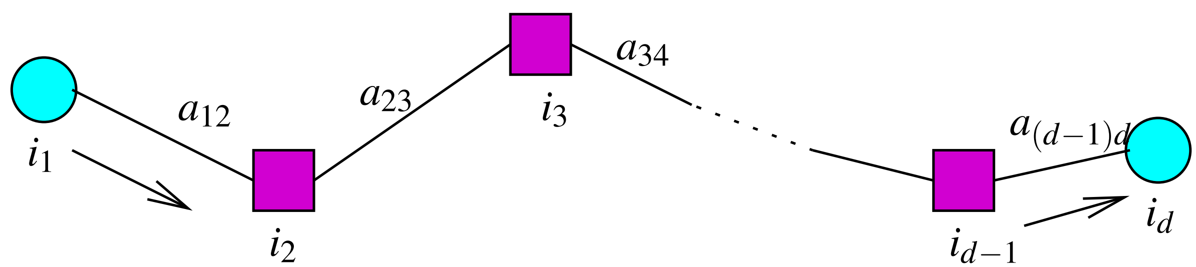
Graph tomography

Linear graph

- assume a graph with the weight $a_{ij} \in \mathcal{R}$ between nodes i and j
- let path $\mathcal{P} = \{(i_1 i_2), (i_2 i_3), \ldots, (i_{d-1} i_d)\}$ from node i_1 to node i_d
  → accumulated weight along the path $\mathcal{P}$ is $a_{i_1 i_d} = \sum_{(ij)\in\mathcal{P}} a_{ij}$
  → in matrix notation

$$\boldsymbol{y} = \boldsymbol{P} \cdot \boldsymbol{a} + \boldsymbol{w}$$

  where rows of binary matrix $\boldsymbol{P} \in \{0,1\}^{m\times n}$ correspond to each measured path, $\boldsymbol{a}$ is graph adjacency vector, and m denotes the number of probes



Multiplicative weights

- if $a_{i_1 i_d} = \prod_{(ij)\in\mathcal{P}} a_{ij}$, we can assume model

$$\log y_{i_1 i_d} = \sum_{(ij)\in\mathcal{P}} \log a_{ij} + w$$

Graph tomography (cont.)

Task 1

- given measurements $\boldsymbol{y}$, find minimal path matrix $\boldsymbol{P}$ to recover k weights in $\boldsymbol{a}$
  $\rightarrow$ this assumes that the graph structure is known
- the path matrix $\boldsymbol{P}$ is minimal if either the sum-length of all probing paths considered is minimum, or if the longest among these paths is minimized
- trivial (non-minimal) design: measure all weights a_{ij} one by one
  $\rightarrow$ determining all paths of a general graph is NP-hard
  $\rightarrow$ in practice, only a subset of nodes may be used for I/O or as gateways

Task 2

- knowing the graph structure, and given subset of paths $\boldsymbol{P}$ and the corresponding measurements $\boldsymbol{y}$, recover k graph weights in $\boldsymbol{a}$
  $\rightarrow$ $\boldsymbol{P}$ may not be large enough to enable compressive sensing of $\boldsymbol{a}$
  $\rightarrow$ some weights in $\boldsymbol{a}$ may be calculated if overdetermined system ($m \geq k$)
- different strategy:
  $\rightarrow$ determine weights change $\Delta\boldsymbol{a}$ if nominal weights $\boldsymbol{a}$ are known

$$\boldsymbol{y} = \boldsymbol{P} \cdot (\boldsymbol{a} + \Delta\boldsymbol{a}) + \boldsymbol{w} = \underbrace{\boldsymbol{P} \cdot \boldsymbol{a}}_{\text{known}} + \boldsymbol{P} \cdot \Delta\boldsymbol{a} + \boldsymbol{w}$$

  assuming k'-sparse change vector $\Delta\boldsymbol{a}$ with $k' \ll k$

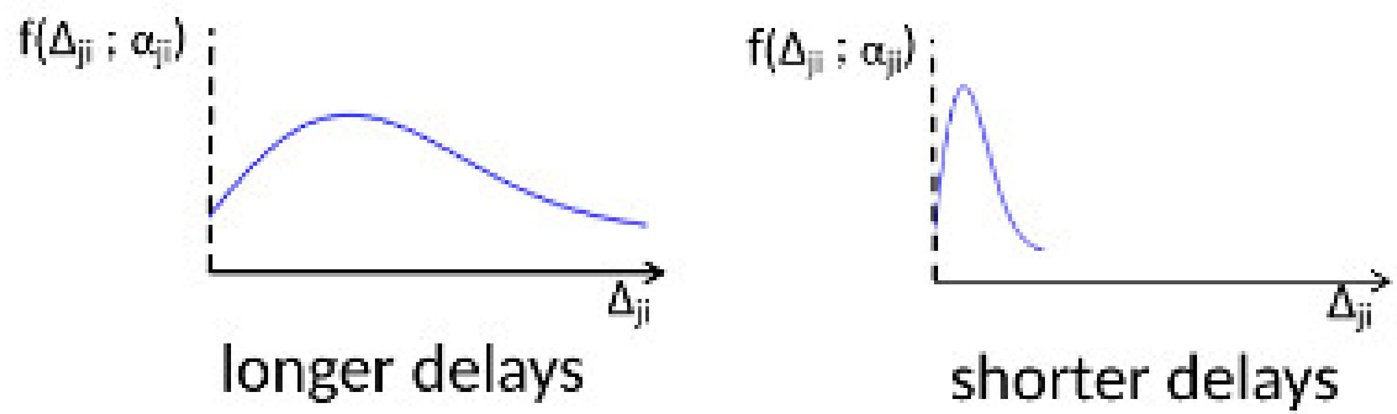
Information diffusion[1]

Diffusion in networks

- information, ideas, behaviors, epidemics, computer viruses, economic transactions, petitions etc.
- transfer occurs in a particular random time

Mathematical model

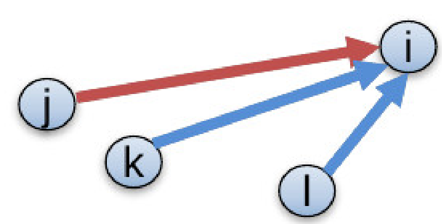
- from node j to i with a rate parameter $\alpha_{ji} \sim 1/ < \Delta_{ij} >$



$f(\Delta_{ij}, \alpha_{ij})$ usually exponential or power-law but also multi-modal distributions

- information cascade: a set $\{t_i\}_i$ where

$$\Delta_{ji} = (t_j - t_i) \sim f(\Delta_{ij}, \alpha_{ij})$$



- node is activated once by the first parent

[1]MG Rodriguez, Le Song, KDD tutorial, 2015.

Information diffusion (cont.)

Network inference problem

- given information cascades $\{t_i^1\}_i$, $\{t_i^2\}_i$, ... infer connection (adjacency) matrix and rate constants $\alpha_{ij} > 0$
  $\rightarrow \alpha_{ij} = 0$ if no connection from i to j
- assuming independence of individual diffusion events, we can obtain and then maximize the likelihood of observed cascades $c \in C$

$$\hat{\boldsymbol{A}} = \mathrm{argmax}_{\boldsymbol{A}} \sum_{c \in C} \log f(\{t_i^c\}_i, \boldsymbol{A})$$

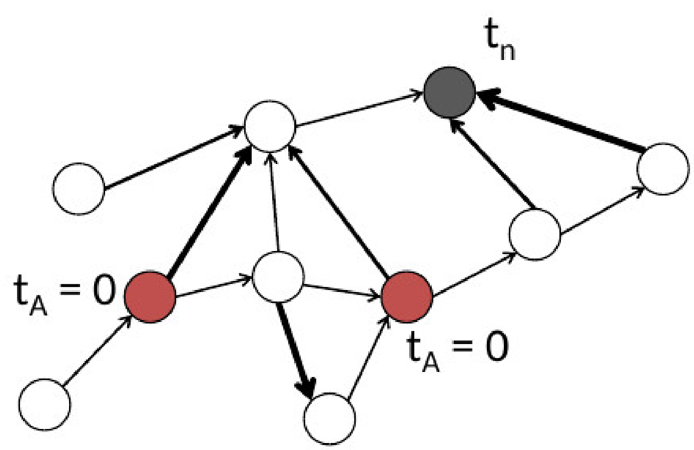
- can we recover completely recover $\boldsymbol{A}$?
  $\rightarrow$ number of cascades $|C|$ needs to be large (cf. compressive sensing)

$$|C| \approx O(d_{\max}^3 \log N)$$

  where $d_{\max}$ is the maximum in-degree in a network of N nodes

Other inferences

- find probability $\Pr(t_n \leq T|\boldsymbol{A})$



Information diffusion (cont.)

Content-sensitive diffusion

$$\Delta_{ji} \sim f(\Delta_{ji}, \alpha_{jim}), \quad m = 1, 2, 3, \ldots$$

- average rate $\bar{\alpha}_{ji} = \sum_m \alpha_{jim} \Pr(m)$

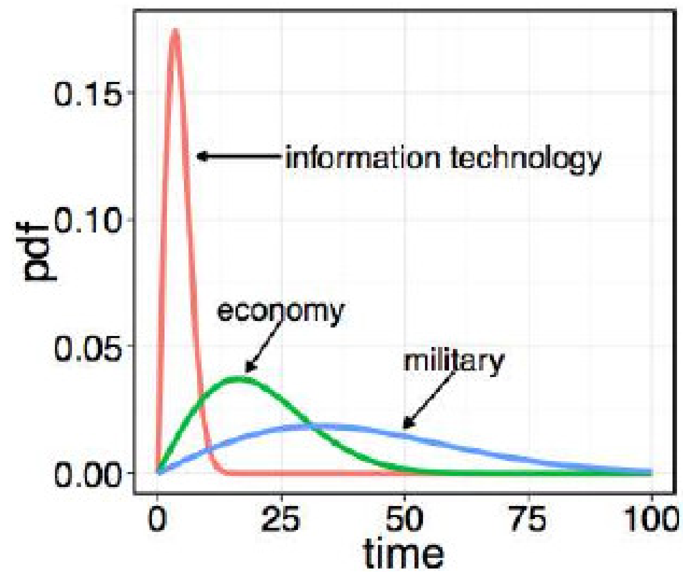
Time-varying rates and connections

$$\Delta_{ji}(\tau) \sim f(\Delta_{ji}, \alpha_{ji}(\tau)),\ \alpha_{ji}(\tau) \geq 0$$

- tracking variations

$$\hat{\boldsymbol{A}}(\tau) = \operatorname{argmax}_{\boldsymbol{A}(\tau)} \sum_{c \in C} w_c(\tau) \log f(\{t_i^c\}_i, \boldsymbol{A}(\tau))$$

where weights $w_c(\tau) = \exp(\tau/T)$
(i.e. forget history more in the past)



Practical challenges

- difficult to identify individual cascades in observed data
  $\rightarrow$ most cascades are single values (never propagates)
- solution: assume counting process (of a specific activity or event)

$$N_i(t) = \sum_{-\infty}^{t} \underbrace{\lambda_i(\tau)}\ \mathrm{d}\tau$$

where the process intensity $\lambda_i \propto \alpha_i$ has the unit [events/hour]

Information diffusion (cont.)

Exogenous activity

- drivers external to the network

Endogenous activity

- response to activities of other users within the network

Total intensity

$$\lambda(t) = \underbrace{\lambda^0(t)}_{\text{exogenous}} + \underbrace{\lambda^*(t)}_{\text{endogenous}}$$

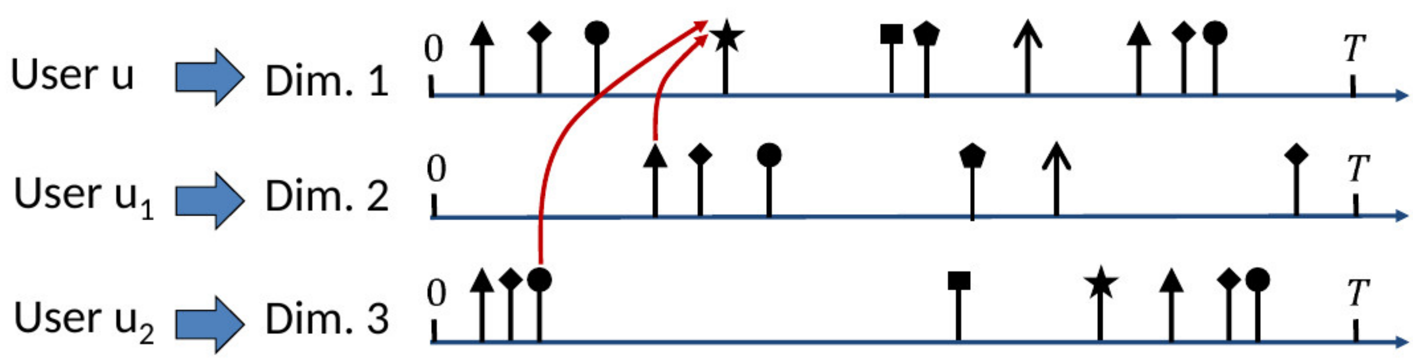
Hawkes process:

$$\lambda^*(t) = \sum_{i:t_i<t} a_{uu_i} g(t - t_i)$$

a_{uu_i}: influence of neighbor u_i on user u

$g(t)$: memory (lifetime of the event or activity)

Correlated events/activities



Information diffusion (cont.)

Maximizing the influence

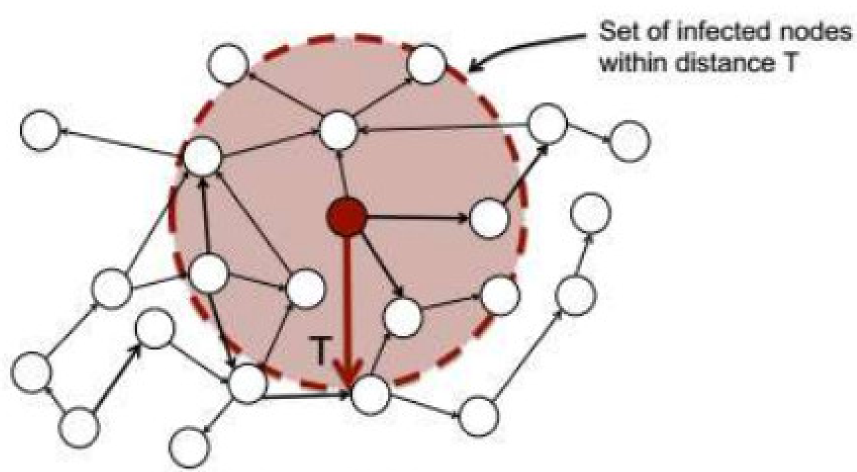
- influence up to time T from nodes in A

$$\sigma(A;T) = \mathrm{E}[N(A;T)] = \sum_{n=1}^{N} \Pr(t_n \leq T|A)$$

- maximization (NP-hard problem)

$$A^* = \mathrm{argmax}_{|A|\leq K}\, \sigma(A;T)$$

$\rightarrow$ we can use a greedy algorithm to solve it or other approximate solutions, e.g.



influence estimation

Information diffusion (cont.)

Source localization

- observations often incomplete, can we still find the first node in a cascade?
- need to eliminate all unobserved events from the likelihood before maximizing it → computationally very difficult problem
  → importance sampling-like strategies seem to work well

Activity shaping

Influence maximization	Fixed incentive	One time same information	Aim is maximum adoption
Activity shaping	Variable incentive	Multiple time, multiple info, recurrent	Many different shaping tasks

- incentivize few users to produce a given level of overall activity
  → exogenous activity → endogenous activity
- example objectives:
  - reach average activity $\mathrm{E}[\lambda(t)]$ at time t where $\lambda(t) = \lambda^{*}(t) + \lambda^{0}(t)$
  - maximize utility $U(\mathrm{E}[\lambda(t)])$
  - maximize activity of the least active user
  - maximize the total number of events in the network

Part 3:

Survey of problems and tasks in graph signal processing

Problems in Network Science

Subgraphs

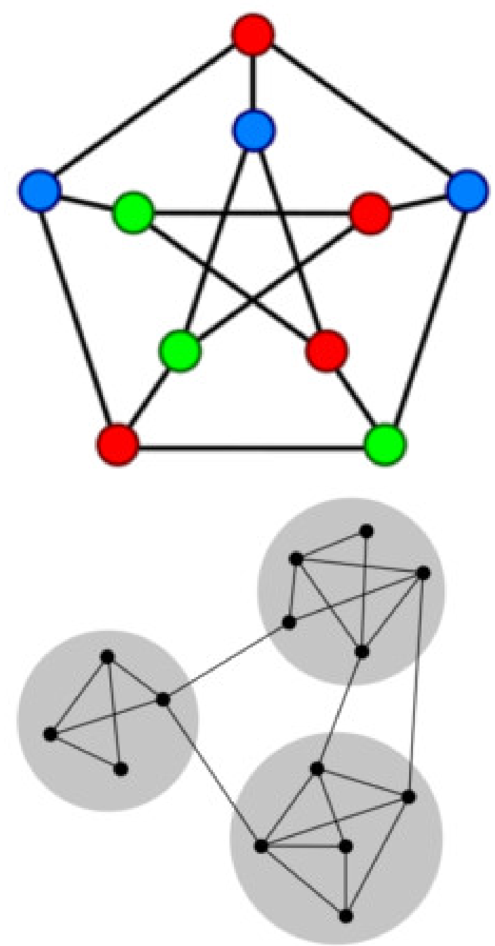
- community detection
- maximum clique identification
- strongly connected components

Path

- minimum spanning tree
- maximum/perfect matching
- shortest path between a pair of nodes
- longest path in a graph (diameter)
- traveling salesman problem

Many other

- drawing and visualization
  → graph coloring
- graph search
- linear ordering of nodes (for acyclic graphs)
  → more generally, ranking and sorting of nodes
- maximum flow/minimum cut



(mostly static graphs)

Problems in Graph Signal Processing

Main concerns

- graph model given *a priory* or learned from data
- working with directed graphs
- how to assign variables to graph
- time-varying and/or random graph structure and variables
- linear and non-linear processing of graph signals
- scalability to very large graph signals

Graph processing tasks (graph structure only)

- graph algebra
  → graph fields
  → addition, multiplication, modulo operations
- graph conversions and transforms
  → different types of graph
  → to/from other discrete structures
- graph structure measures and description
  → well defined by Network Science (motifs, clustering etc.)
- graph embedding in high-dimensional vector space
  → often used in machine learning tasks

Problems in Graph Signal Processing (cont.)

Graph signal processing tasks (graph structure and variables)

- defining graph signals
  → mapping variables to graph (nodes, edges, simplexes)
  → learning graph from data (given type or general graph)
  → graph signal ⟷ manifold conversions
- orthonormal decomposition
  → *a priory* basis selection or basis learned from data
- spectral (frequency domain) representation
  → Fourier analysis (discrete Fourier transforms)
  → wavelet transforms
- filtering
  → in time and in frequency domains
  → convolution, de-convolution, windowing
  → designing low-pass and high-pass filters
  → observability and controlability
- sampling, downsampling and compression
  → lossless or near-perfect reconstruction
  → graph signal tomography

Problems in Graph Signal Processing (cont.)

Graph signal processing tasks (cont.)

- invariant measures
  → graph signal shift in time and frequency domains
- clustering, classification, sorting, ranking samples of graph signals
- time-varying signals
  → flows and diffusion on graphs (information)
  → events spreading over graphs (epidemics)
  → event detection from small number of observations
- random graph signals
  → stationarity and ergodicity
  → de-noising and smoothing
  → parameter estimation from small number of observations

Driving applications of graph signal processing

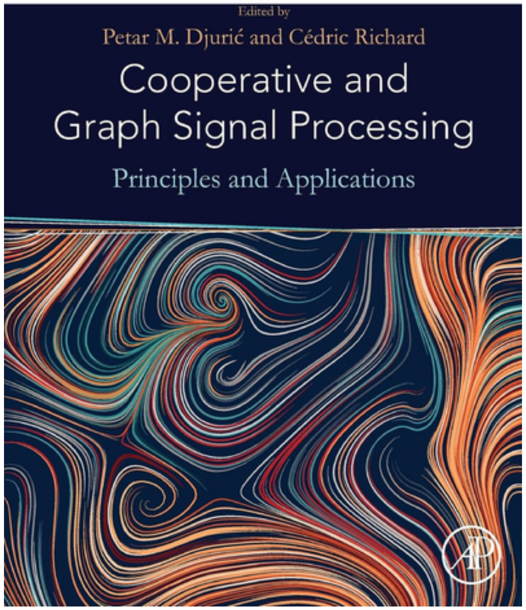
- image processing
- brain functional networks
- semi-supervised and deep graph learning

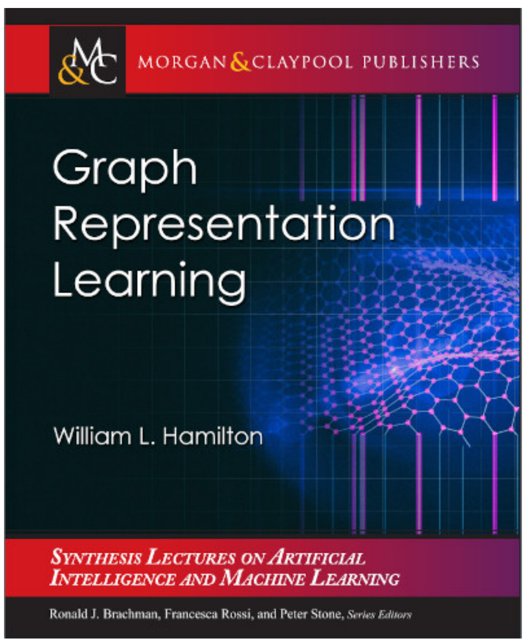
  ... bottom line: taking into account signal (graph) structure can greatly improve the performance

Part 4:

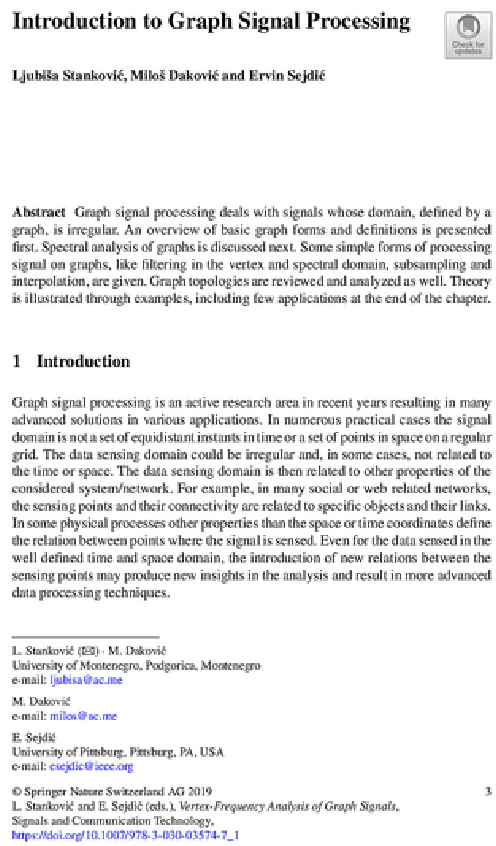
Recommended readings on graph signal processing

Books





Book Chapters



Introduction to Graph Signal Processing

Ljubiša Stanković, Miloš Daković and Ervin Sejdić

Abstract Graph signal processing deals with signals whose domain, defined by a graph, is irregular. An overview of basic graph forms and definitions is presented first. Spectral analysis of graphs is discussed next. Some simple forms of processing signal on graphs, like filtering in the vertex and spectral domain, subsampling and interpolation, are given. Graph topologies are reviewed and analyzed as well. Theory is illustrated through examples, including few applications at the end of the chapter.

1 Introduction

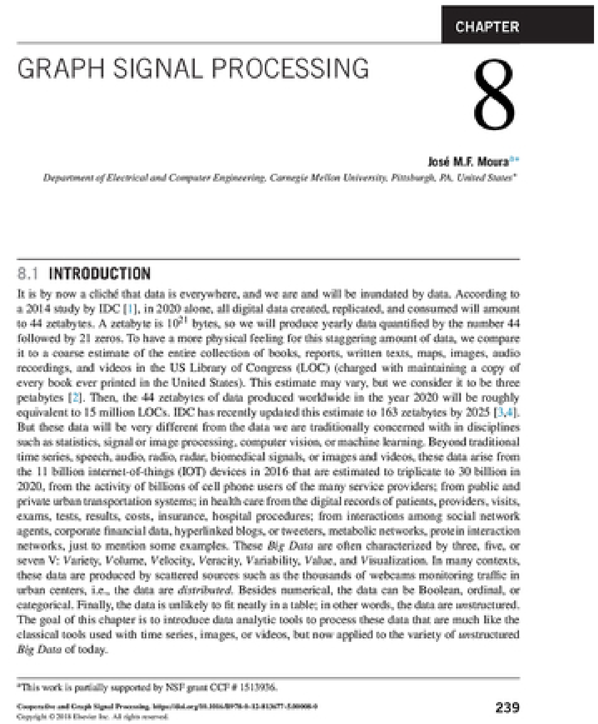
Graph signal processing is an active research area in recent years resulting in many advanced solutions in various applications. In numerous practical cases the signal domain is not a set of equidistant instants in time or a set of points in space on a regular grid. The data sensing domain could be irregular and, in some cases, not related to the time or space. The data sensing domain is then related to other properties of the considered system/network. For example, in many social or web related networks, the sensing points and their connectivity are related to specific objects and their links. In some physical processes other properties than the space or time coordinates define the relation between points where the signal is sensed. Even for the data sensed in the well defined time and space domain, the introduction of new relations between the sensing points may produce new insights in the analysis and result in more advanced data processing techniques.

L. Stanković (✉) · M. Daković
University of Montenegro, Podgorica, Montenegro
e-mail: ljubisa@ac.me

M. Daković
e-mail: milos@ac.me

E. Sejdić
University of Pittsburg, Pittsburg, PA, USA
e-mail: esejdic@ieee.org

© Springer Nature Switzerland AG 2019 3
L. Stanković and E. Sejdić (eds.), *Vertex-Frequency Analysis of Graph Signals*, Signals and Communication Technology,
https://doi.org/10.1007/978-3-030-03574-7_1



CHAPTER 8

GRAPH SIGNAL PROCESSING

José M.F. Moura[a,*]
Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, United States[a]

8.1 INTRODUCTION

It is by now a cliché that data is everywhere, and we are and will be inundated by data. According to a 2014 study by IDC [1], in 2020 alone, all digital data created, replicated, and consumed will amount to 44 zetabytes. A zetabyte is 10^{21} bytes, so we will produce yearly data quantified by the number 44 followed by 21 zeros. To have a more physical feeling for this staggering amount of data, we compare it to a coarse estimate of the entire collection of books, reports, written texts, maps, images, audio recordings, and videos in the US Library of Congress (LOC) (charged with maintaining a copy of every book ever printed in the United States). This estimate may vary, but we consider it to be three petabytes [2]. Then, the 44 zetabytes of data produced worldwide in the year 2020 will be roughly equivalent to 15 million LOCs. IDC has recently updated this estimate to 163 zetabytes by 2025 [3,4]. But these data will be very different from the data we are traditionally concerned with in disciplines such as statistics, signal or image processing, computer vision, or machine learning. Beyond traditional time series, speech, audio, radio, radar, biomedical signals, or images and videos, these data arise from the 11 billion internet-of-things (IOT) devices in 2016 that are estimated to triplicate to 30 billion in 2020, from the activity of billions of cell phone users of the many service providers; from public and private urban transportation systems; in health care from the digital records of patients, providers, visits, exams, tests, results, costs, insurance, hospital procedures; from interactions among social network agents, corporate financial data, hyperlinked blogs, or tweeters, metabolic networks, protein interaction networks, just to mention some examples. These *Big Data* are often characterized by three, five, or seven V: *Variety*, *Volume*, *Velocity*, *Veracity*, *Variability*, *Value*, and *Visualization*. In many contexts, these data are produced by scattered sources such as the thousands of webcams monitoring traffic in urban centers, i.e., the data are *distributed*. Besides numerical, the data can be Boolean, ordinal, or categorical. Finally, the data is unlikely to fit neatly in a table; in other words, the data are *unstructured*. The goal of this chapter is to introduce data analytic tools to process these data that are much like the classical tools used with time series, images, or videos, but now applied to the variety of *unstructured Big Data* of today.

*This work is partially supported by NSF grant CCF # 1513936.

Cooperative and Graph Signal Processing. 239
Copyright © 2018 Elsevier Inc. All rights reserved.

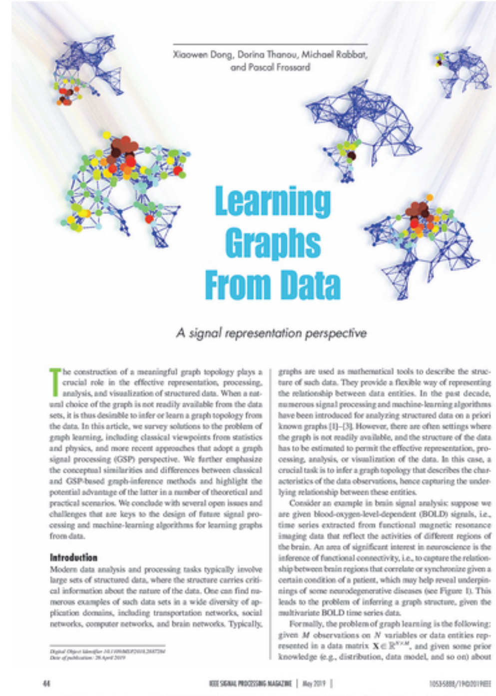
Journal Papers


Topological Signal Processing over Simplicial Complexes

Sergio Barbarossa, *Fellow, IEEE*, Stefania Sardellitti, *Member, IEEE*

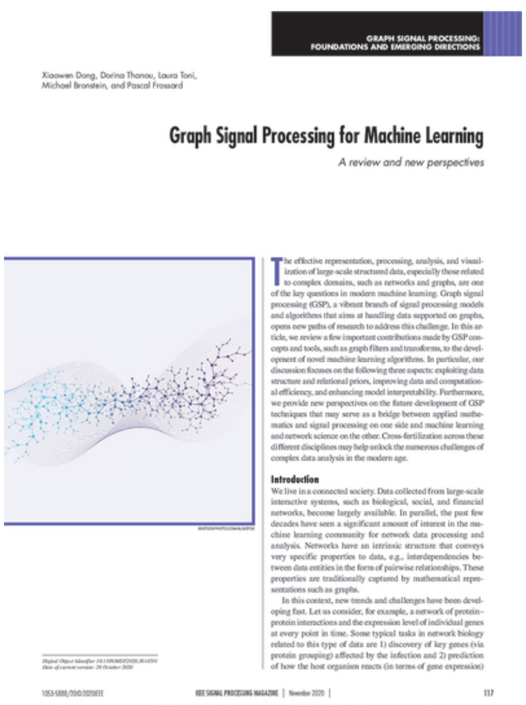

Adaptive Graph Signal Processing: Algorithms and Optimal Sampling Strategies

Paolo Di Lorenzo, *Member, IEEE*, Paolo Banelli, *Member, IEEE*, Elvin Isufi, *Student Member, IEEE*, Sergio Barbarossa, *Fellow, IEEE*, and Geert Leus, *Fellow, IEEE*


Xiaowen Dong, Dorina Thanou, Michael Rabbat, and Pascal Frossard

Learning Graphs From Data

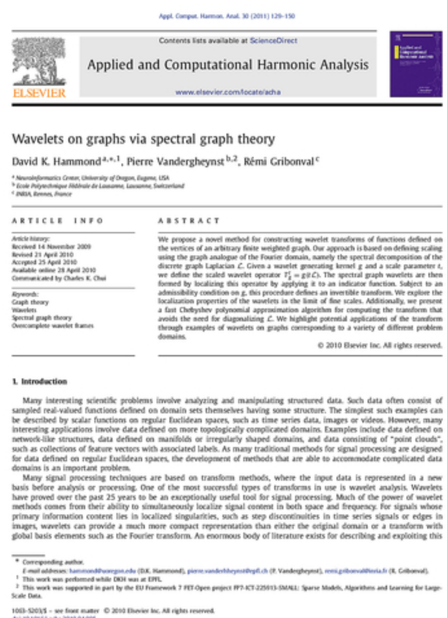
A signal representation perspective


GRAPH SIGNAL PROCESSING: FOUNDATIONS AND EMERGING DIRECTIONS

Xiaowen Dong, Dorina Thanou, Laura Toni, Michael Bronstein, and Pascal Frossard

Graph Signal Processing for Machine Learning

A review and new perspectives


Applied and Computational Harmonic Analysis

Wavelets on graphs via spectral graph theory

David K. Hammond, Pierre Vandergheynst, Rémi Gribonval


Gonzalo Mateos, Santiago Segarra, Antonio G. Marques, and Alejandro Ribeiro

Connecting the Dots

Identifying network structure via graph signal processing

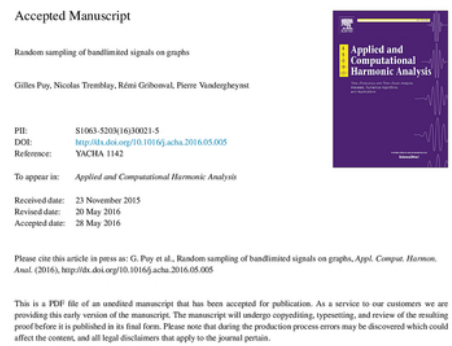
Journal Papers (cont.)


INVITED PAPER

Graph Signal Processing: Overview, Challenges, and Applications

This article presents methods to process data associated to graphs (graph signals) extending techniques (transforms, sampling, and others) that are used for conventional signal data.

By Antonio Ortega, Fellow IEEE, Pascal Frossard, Fellow IEEE, Jelena Kovačević, Fellow IEEE, José M. F. Moura, Fellow IEEE, and Pierre Vandergheynst


Accepted Manuscript

Random sampling of bandlimited signals on graphs

Gilles Puy, Nicolas Tremblay, Rémi Gribonval, Pierre Vandergheynst

PII: S1063-5203(16)30021-5

DOI: http://dx.doi.org/10.1016/j.acha.2016.05.005

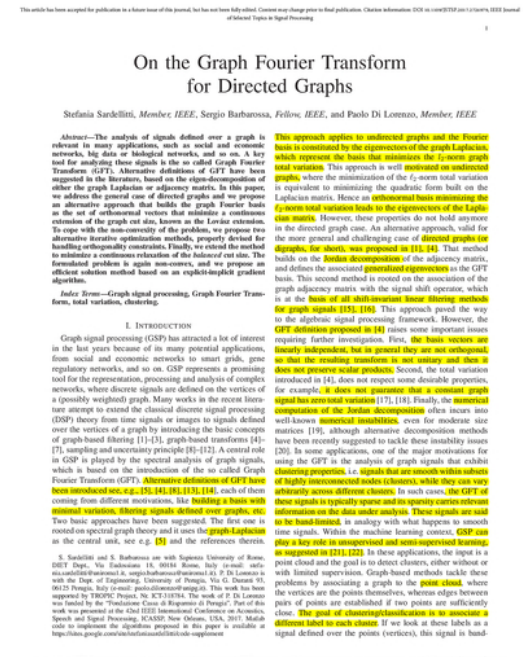
Reference: YACHA 1142

To appear in: Applied and Computational Harmonic Analysis

Received date: 23 November 2015

Revised date: 20 May 2016

Accepted date: 28 May 2016


On the Graph Fourier Transform for Directed Graphs

Stefania Sardellitti, Member, IEEE, Sergio Barbarossa, Fellow, IEEE, and Paolo Di Lorenzo, Member, IEEE


Graph topology inference based on sparsifying transform learning

Stefania Sardellitti, Member, IEEE, Sergio Barbarossa, Fellow, IEEE, and Paolo Di Lorenzo, Member, IEEE

arXiv:1806.01701v1 [eess.SP] 5 Jun 2018


FLOW SMOOTHING AND DENOISING: GRAPH SIGNAL PROCESSING IN THE EDGE-SPACE

Michael T. Schaub and Santiago Segarra


Vertex-Frequency Analysis on Graphs

David I Shuman, Benjamin Ricaud, Pierre Vandergheynst

arXiv:1307.5708v1 [math.FA] 22 Jul 2013


Noname manuscript No. (will be inserted by the editor)

Introduction to Graph Signal Processing

Ljubiša Stanković, Miloš Daković, and Ervin Sejdić

PRESENTATIONS

Graph Signal Processing: An Introductory Overview

Antonio Ortega

Signal and Image Processing Institute
Department of Electrical Engineering
University of Southern California
Los Angeles, California

May 25, 2016

Dictionary Design for Graph Signal Processing

David Shuman

May 25, 2016
Graph Signal Processing Workshop
Philadelphia, PA

Special thanks and acknowledgement to my collaborators:
Andre Archer, Andrew Beveridge, Xiaowen Dong, Mohammad Javad Faraji, Stefan Faridani, Pascal Frossard, Nicki Holighaus, Jason McEwen, Daniel Kressner, Yan Jin, Sunil Narang, Antonio Ortega, Javier Pérez-Trufero, Nathanaël Perraudin, Benjamin Ricaud, Dorina Thanou, Pierre Vandergheynst, Elle Weeks, and Christoph Wiesmeyr

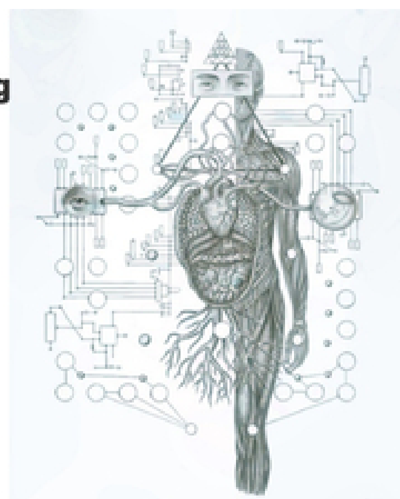
MACALESTER COLLEGE

Geometric Deep Learning

For non-grid 3D images like point clouds and meshes, and inherently graph-based data

Version "Wed 13 September 2017"

Petteri Teikari, PhD
http://petteri-teikari.com/
https://www.linkedin.com/in/petteriteikari/



Wavelets on Graphs, an Introduction

Pierre Vandergheynst and David Shuman

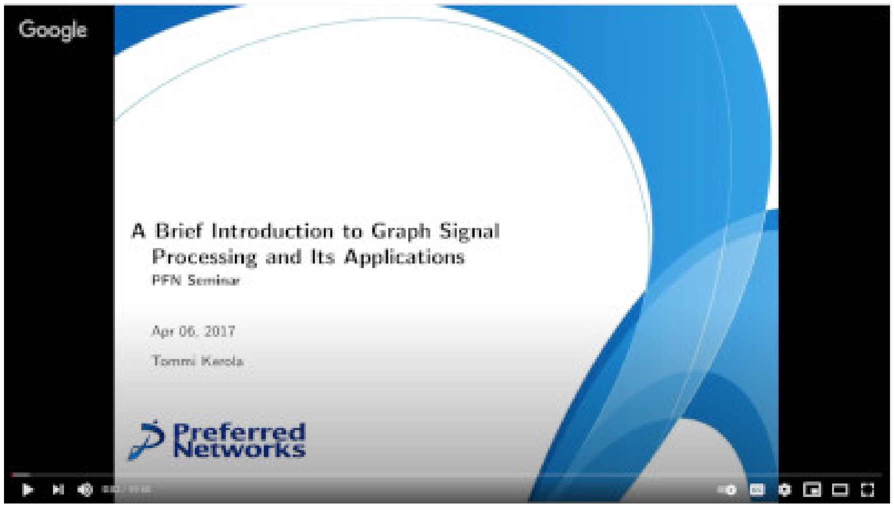
Ecole Polytechnique Fédérale de Lausanne (EPFL)
Signal Processing Laboratory
{pierre.vandergheynst,david.shuman}@epfl.ch

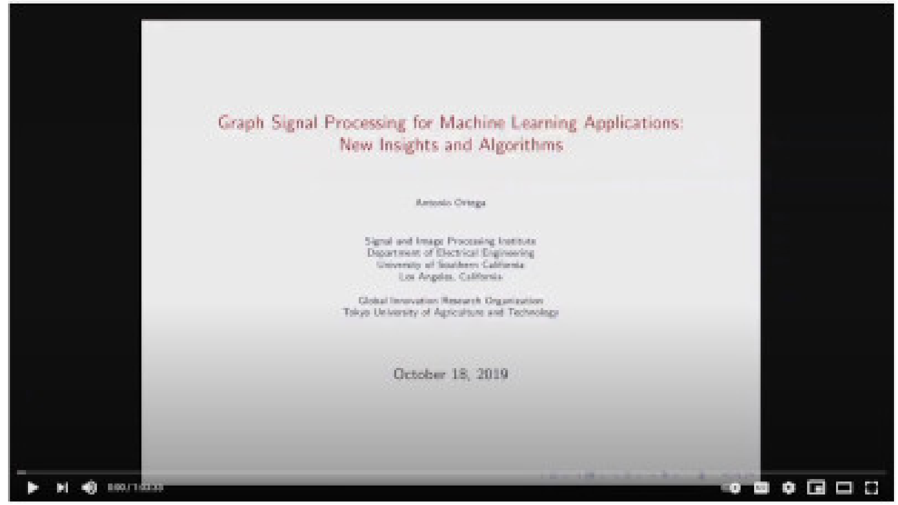
Université de Provence
Marseille, France

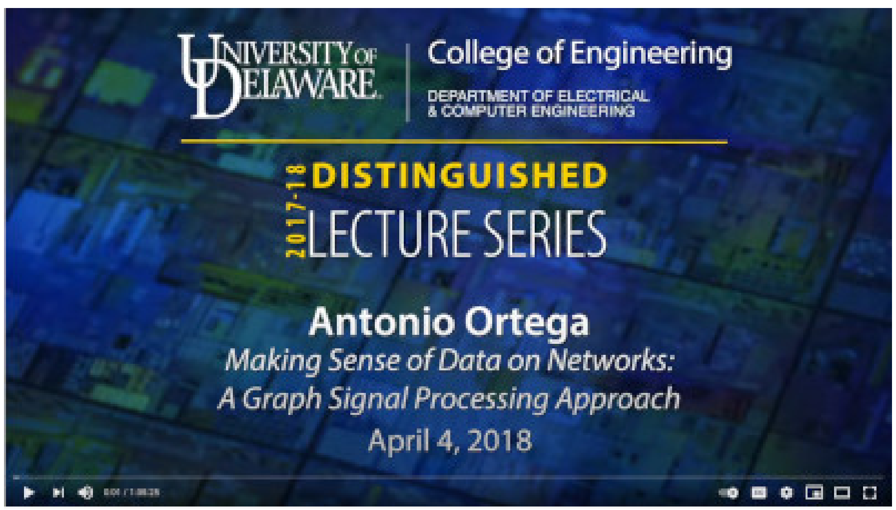
November 17, 2011

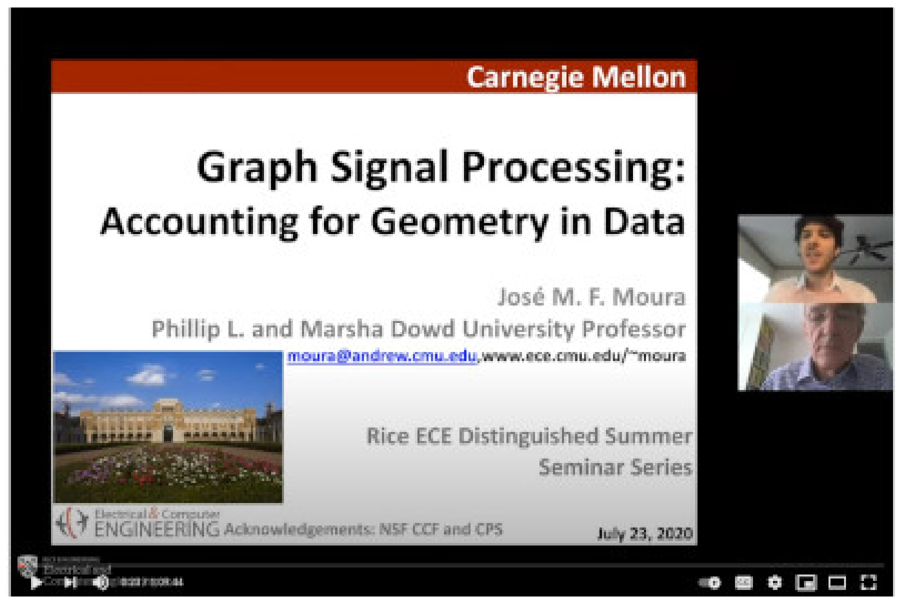
LTS LTS4 EPFL

YOUTUBE PRESENTATIONS









YOUTUBE PRESENTATIONS (CONT.)

Conclusions

In Summary

Main observations

- there is clearly a need to work with graph signals
  $\rightarrow$ network like systems and applications
  $\rightarrow$ accounting for signal structure can significantly improve the performance
- (not surprisingly) the aim is to extend signal processing to graph signals
- most techniques for graph signal processing evolved around eigen-decomposition of $\boldsymbol{W}$ or $\boldsymbol{L}$

... however

- there are often several/many different approaches to accomplishing the same thing or task in graph signal processing
  $\rightarrow$ the field still appears far less mature than e.g. Network Science
  $\rightarrow$ a viable strategy can be transforms of graphs $\longleftrightarrow$ regular signals
- distributed signal processing seems to be appealing practical alternative to (centralized) graph signal processing

Thank you!

pavelloskot@intl.zju.edu.cn