gipsa-lab

Grenoble | images | parole | signal | automatique | laboratoire

Wilson's Algorithm for Randomized Linear Algebra

Yusuf Yiğit Pilavcı

09/06/2022

UMR 5216

Introduction

- Matrix multiplication, inversion, eigendecomposition and SVD are some of the indispensable operations in many fields.

Introduction

- Matrix multiplication, inversion, eigendecomposition and SVD are some of the indispensable operations in many fields.
- In large scale, faster algorithms might be necessary

Introduction

- Matrix multiplication, inversion, eigendecomposition and SVD are some of the indispensable operations in many fields.
- In large scale, faster algorithms might be necessary
- Randomized numerical linear algebra (RNLA) aims to facilitate them via Monte Carlo methods [Martinsson and Tropp 2020].

Introduction

- Matrix multiplication, inversion, eigendecomposition and SVD are some of the indispensable operations in many fields.
- In large scale, faster algorithms might be necessary
- Randomized numerical linear algebra (RNLA) aims to facilitate them via Monte Carlo methods [Martinsson and Tropp 2020].
- The algorithms in this branch run under a scheme called sample-and-solve.

Introduction

- This thesis takes another direction for developing RLA algorithms for *Laplacian-based numerical algebra*.

Introduction

- This thesis takes another direction for developing RLA algorithms for *Laplacian-based numerical algebra*.
- The graph Laplacians have many applications in:

Introduction

- This thesis takes another direction for developing RLA algorithms for *Laplacian-based numerical algebra*.
- The graph Laplacians have many applications in:
  - Graph signal processing (filtering, sampling, ...) [Shuman et al.]

Introduction

- This thesis takes another direction for developing RLA algorithms for *Laplacian-based numerical algebra*.
- The graph Laplacians have many applications in:
  - Graph signal processing (filtering, sampling, ...) [Shuman et al.]
  - Machine learning (semi-supervised learning, GNNs, ...) [Zhu; Wu et al.]

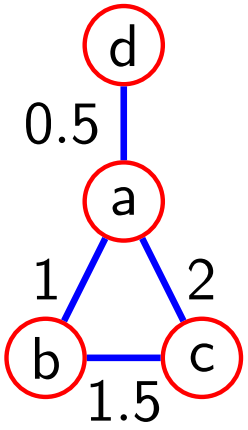
Introduction

- This thesis takes another direction for developing RLA algorithms for *Laplacian-based numerical algebra*.
- The graph Laplacians have many applications in:
  - Graph signal processing (filtering, sampling, ...) [Shuman et al.]
  - Machine learning (semi-supervised learning, GNNs, ...) [Zhu; Wu et al.]
  - Graph visualization, clustering, sparsification ...

Introduction

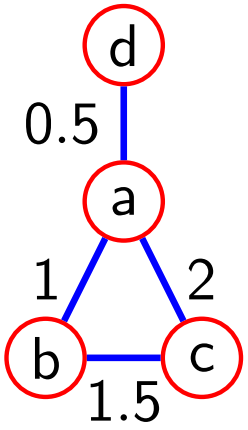
- This thesis takes another direction for developing RLA algorithms for *Laplacian-based numerical algebra*.
- The graph Laplacians have many applications in:
  - Graph signal processing (filtering, sampling, ...) [Shuman et al.]
  - Machine learning (semi-supervised learning, GNNs, ...) [Zhu; Wu et al.]
  - Graph visualization, clustering, sparsification ...
- We leverage the rich theoretical links between the graph Laplacians and a special random process over graphs, called Random Spanning Forests.

Graphs and Matrices



$\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$

Graphs and Matrices

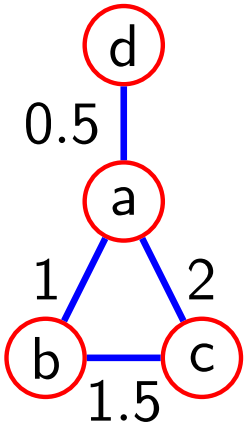


$$\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$$

$$\begin{bmatrix} 0 & 1 & 2 & 0.5 \\ 1 & 0 & 1.5 & 0 \\ 2 & 1.5 & 0 & 0 \\ 0.5 & 0 & 0 & 0 \end{bmatrix}$$

Adjacency matrix W

Graphs and Matrices



$$\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$$

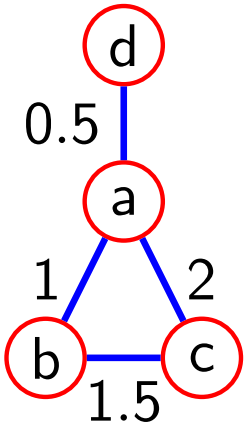
$$\begin{bmatrix} 0 & 1 & 2 & 0.5 \\ 1 & 0 & 1.5 & 0 \\ 2 & 1.5 & 0 & 0 \\ 0.5 & 0 & 0 & 0 \end{bmatrix}$$

Adjacency matrix W

$$\begin{bmatrix} 3.5 & 0 & 0 & 0 \\ 0 & 2.5 & 0 & 0 \\ 0 & 0 & 3.5 & 0 \\ 0 & 0 & 0 & 0.5 \end{bmatrix}$$

Degree matrix D

Graphs and Matrices



$\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$

$$\begin{bmatrix} 0 & 1 & 2 & 0.5 \\ 1 & 0 & 1.5 & 0 \\ 2 & 1.5 & 0 & 0 \\ 0.5 & 0 & 0 & 0 \end{bmatrix}$$

Adjacency matrix W

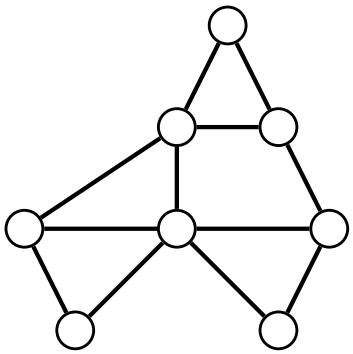
$$\begin{bmatrix} 3.5 & 0 & 0 & 0 \\ 0 & 2.5 & 0 & 0 \\ 0 & 0 & 3.5 & 0 \\ 0 & 0 & 0 & 0.5 \end{bmatrix}$$

Degree matrix D

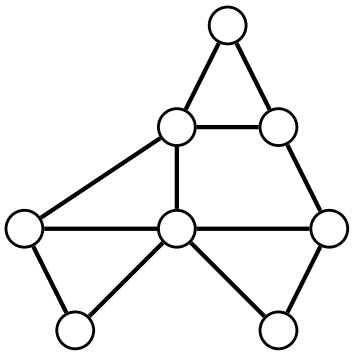
$$\begin{bmatrix} 3.5 & -1 & -2 & -0.5 \\ -1 & 2.5 & -1.5 & 0 \\ -2 & -1.5 & 3.5 & 0 \\ -0.5 & 0 & 0 & 0.5 \end{bmatrix}$$

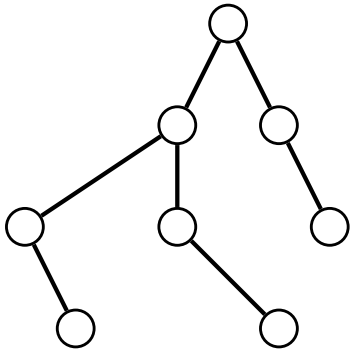
Laplacian matrix $\mathrm{L} = \mathrm{D} - \mathrm{W}$

Spanning Forests, Roots and Partitions



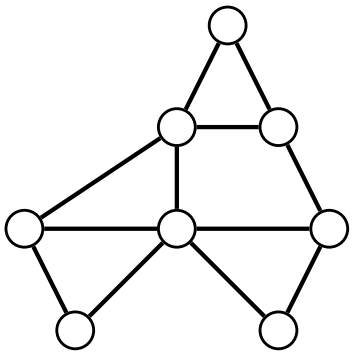
Spanning Forests, Roots and Partitions

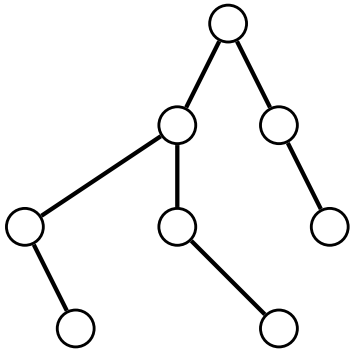




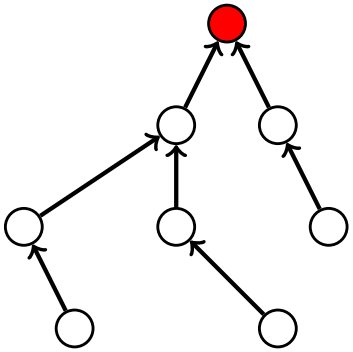
Spanning Tree

Spanning Forests, Roots and Partitions



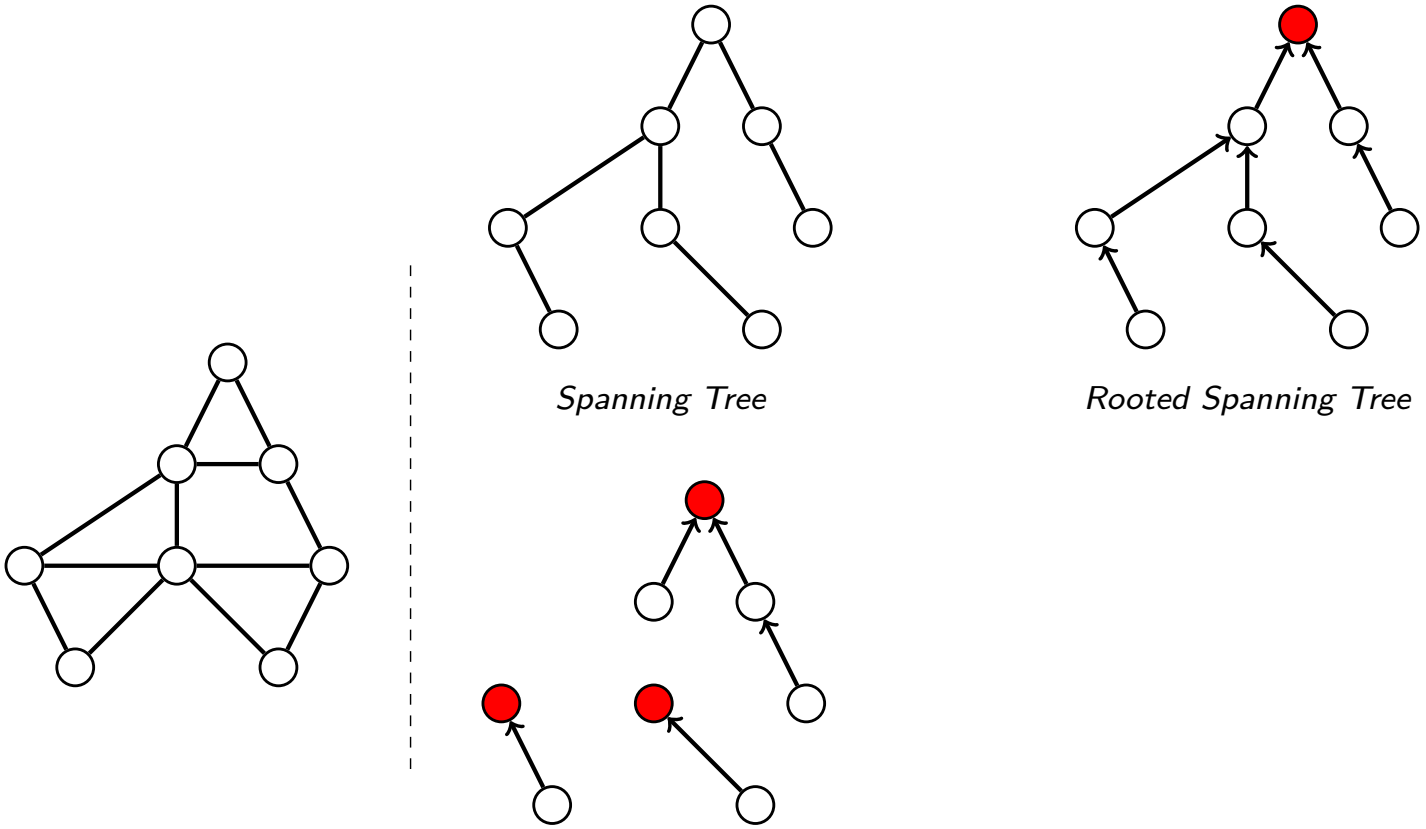


Spanning Tree

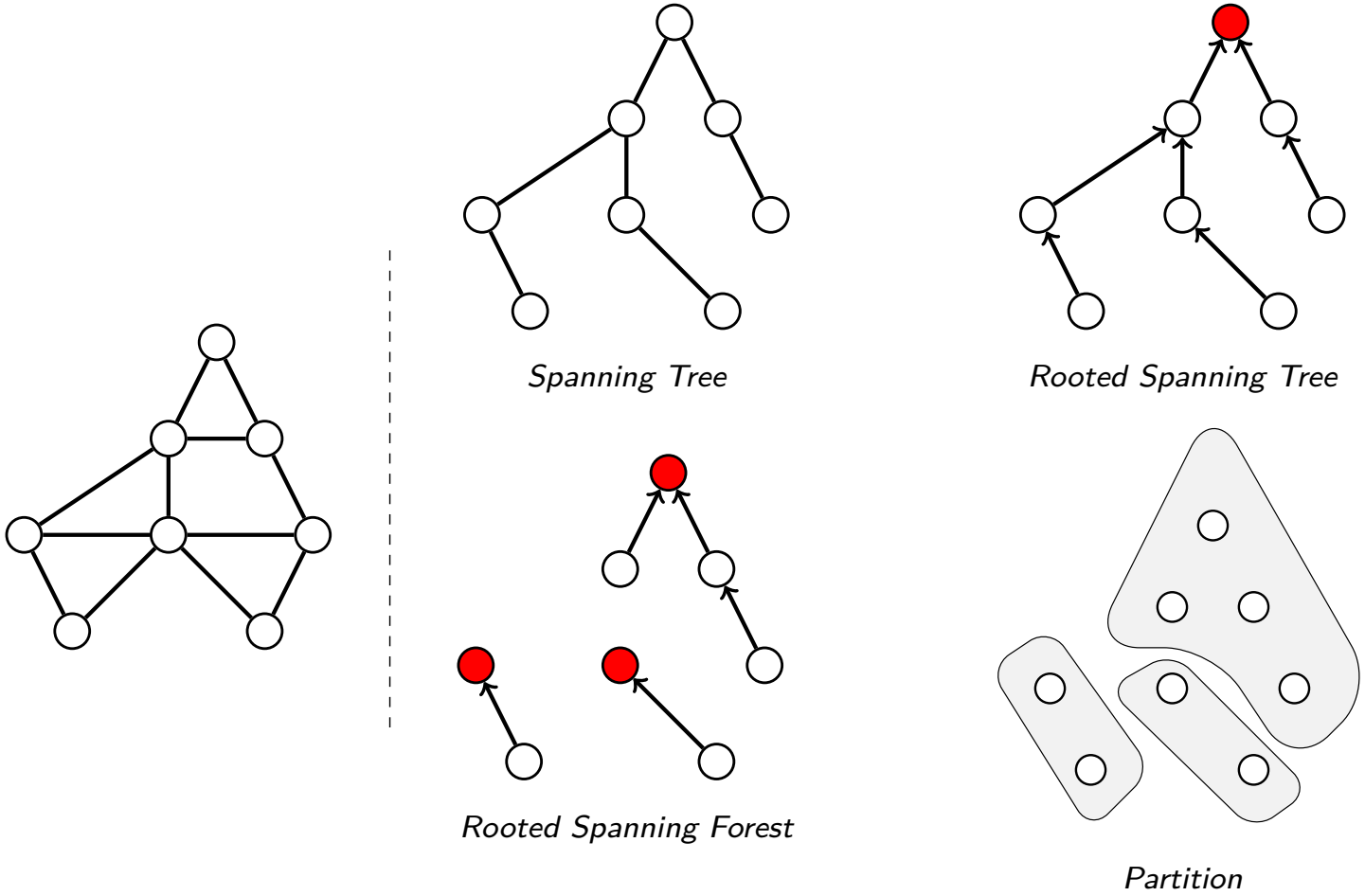


Rooted Spanning Tree

Spanning Forests, Roots and Partitions

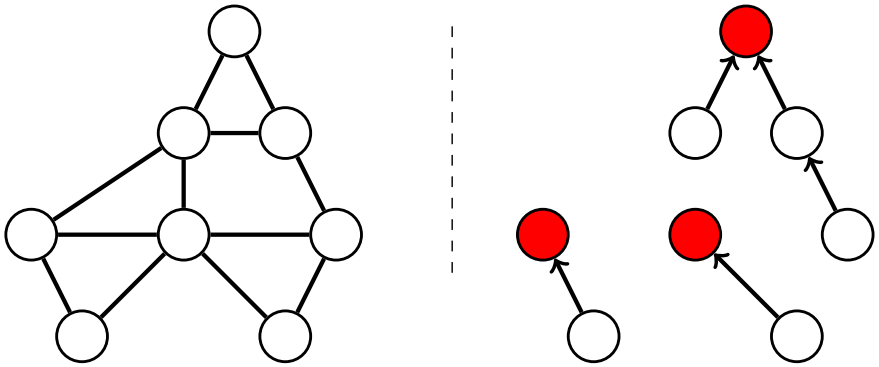


Spanning Forests, Roots and Partitions



A few more notation

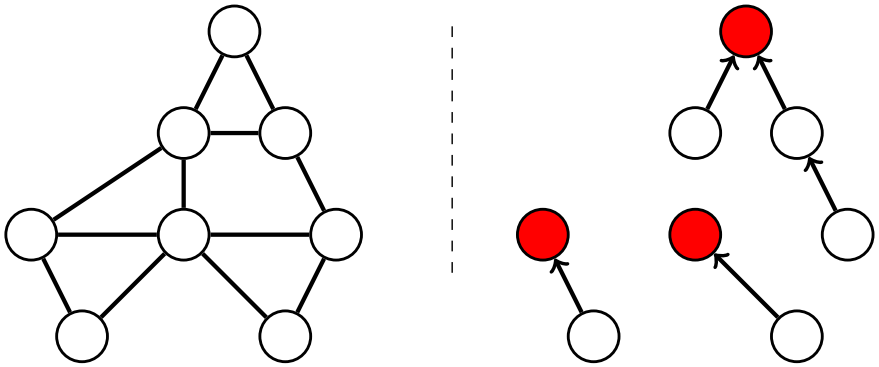
- Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$, we denote:



- a spanning forest by ϕ

A few more notation

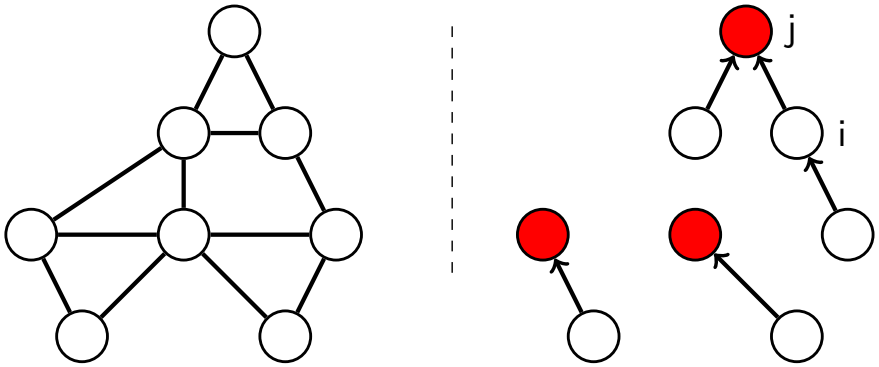
- Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$, we denote:



- a spanning forest by ϕ and its root set by $\rho(\phi)$,

A few more notation

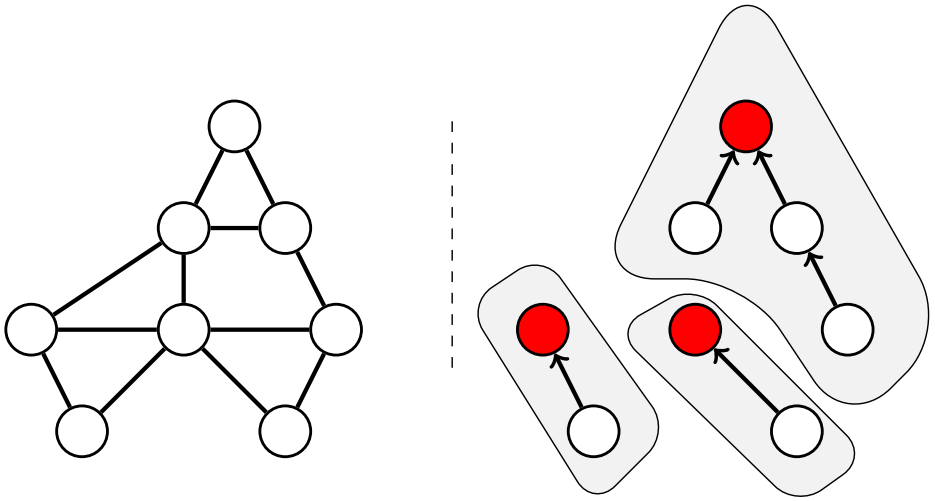
- Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$, we denote:



  - a spanning forest by ϕ and its root set by $\rho(\phi)$,
  - the root of vertex i in ϕ by $r_\phi(i) = j$

A few more notation

- Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$, we denote:



  - a spanning forest by ϕ and its root set by $\rho(\phi)$,
  - the root of vertex i in ϕ by $r_\phi(i) = j$

Random Spanning Forests: **What**, Why and How?

Definition (RSF)

A random spanning forest Φ_q on a graph $\mathcal{G}$ is spanning forest selected over all spanning forests of $\mathcal{G}$ according to the following distribution:

$$P(\Phi_q = \phi) \propto q^{|\rho(\phi)|} \prod_{(i,j)\in\mathcal{E}_\phi} w(i,j)$$

Random Spanning Forests: **What**, Why and How?

Definition (RSF)

A random spanning forest Φ_q on a graph $\mathcal{G}$ is spanning forest selected over all spanning forests of $\mathcal{G}$ according to the following distribution:

$$P(\Phi_q = \phi) \propto q^{|\rho(\phi)|} \prod_{(i,j)\in\mathcal{E}_\phi} w(i,j)$$

- $q > 0$ changes the expected number of roots.

Random Spanning Forests: What, **Why** and How?

- In a particular case, RSFs boil down to uniform spanning trees (USTs):

$$\mathbb{P}(T = \tau) = \frac{1}{|\mathcal{T}|}, \quad \forall \tau \in \mathcal{T}$$

Random Spanning Forests: What, **Why** and How?

- In a particular case, RSFs boil down to uniform spanning trees (USTs):

$$\mathbb{P}(T = \tau) = \frac{1}{|\mathcal{T}|}, \quad \forall \tau \in \mathcal{T}$$

- USTs are well-studied in statistical physics, probability theory and computer science.

Random Spanning Forests: What, **Why** and How?

- In a particular case, RSFs boil down to uniform spanning trees (USTs):

$$\mathbb{P}(T = \tau) = \frac{1}{|\mathcal{T}|}, \quad \forall \tau \in \mathcal{T}$$

- USTs are well-studied in statistical physics, probability theory and computer science.
- There exists an efficient algorithm to sample a spanning tree, called Wilson's algorithm.

Random Spanning Forests: What, **Why** and How?

- In a particular case, RSFs boil down to uniform spanning trees (USTs):

$$\mathbb{P}(T = \tau) = \frac{1}{|\mathcal{T}|}, \quad \forall \tau \in \mathcal{T}$$

- USTs are well-studied in statistical physics, probability theory and computer science.
- There exists an efficient algorithm to sample a spanning tree, called Wilson's algorithm.
- Many other properties of USTs and more also extend to the RSFs.

Random Spanning Forests: What, **Why** and How?

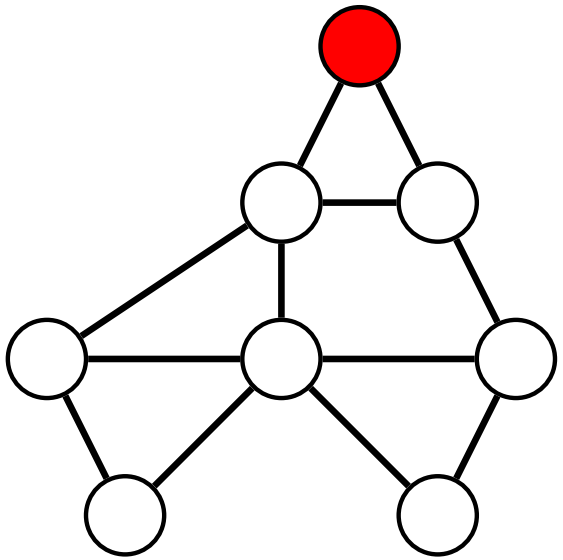
- In a particular case, RSFs boil down to uniform spanning trees (USTs):

$$\mathbb{P}(T = \tau) = \frac{1}{|\mathcal{T}|}, \quad \forall \tau \in \mathcal{T}$$

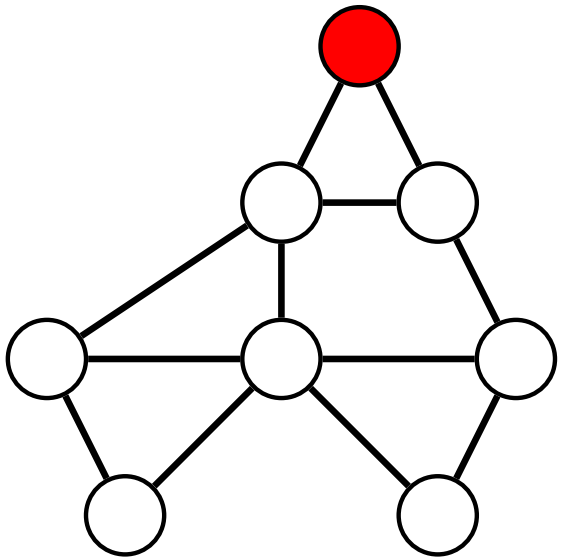
- USTs are well-studied in statistical physics, probability theory and computer science.
- There exists an efficient algorithm to sample a spanning tree, called Wilson's algorithm.
- Many other properties of USTs and more also extend to the RSFs.
- Moreover, we have the following identity:

$$\forall i, j \in \mathcal{V}, \quad \mathbb{P}(r_{\Phi_q}(i) = j) = \mathrm{K}_{i,j}, \text{ with } \mathrm{K} = q(\mathrm{L} + q\mathrm{I})^{-1}.$$

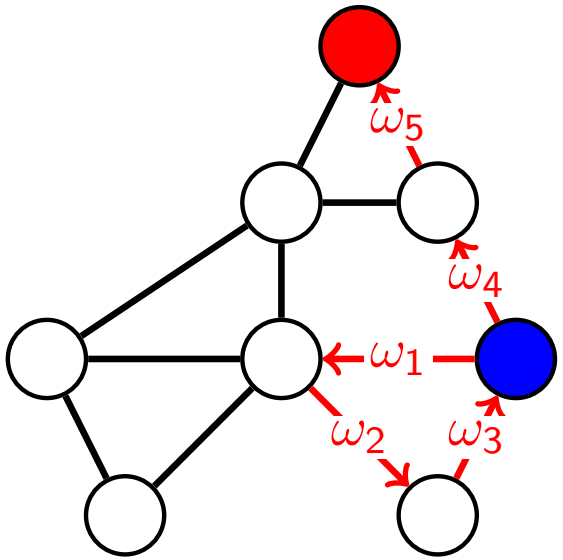
Random Spanning Forests: What, Why and **How**?



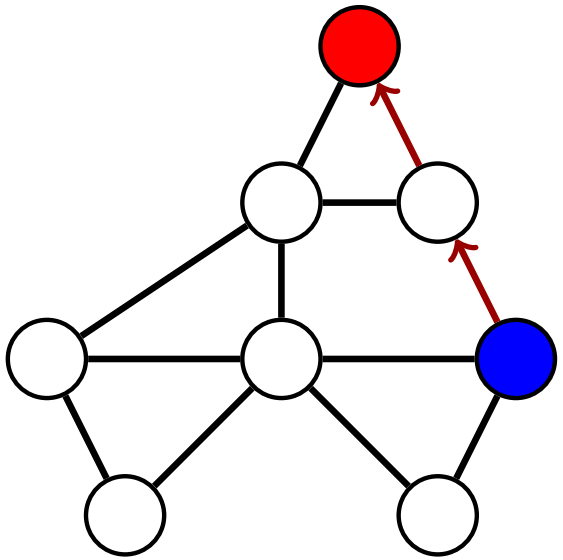
Random Spanning Forests: What, Why and **How**?



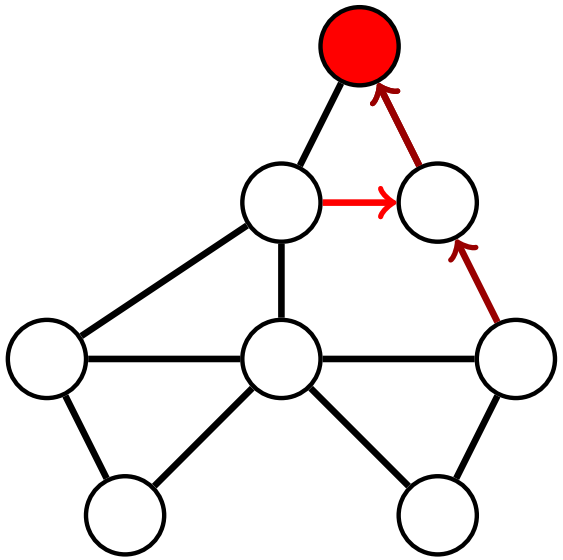
Random Spanning Forests: What, Why and **How**?



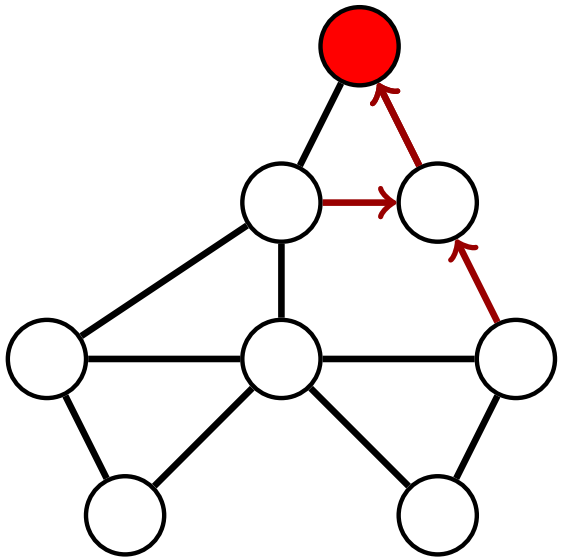
Random Spanning Forests: What, Why and **How**?



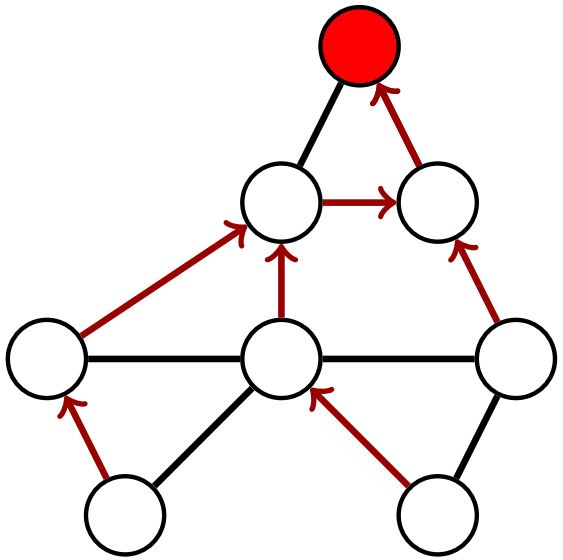
Random Spanning Forests: What, Why and **How**?



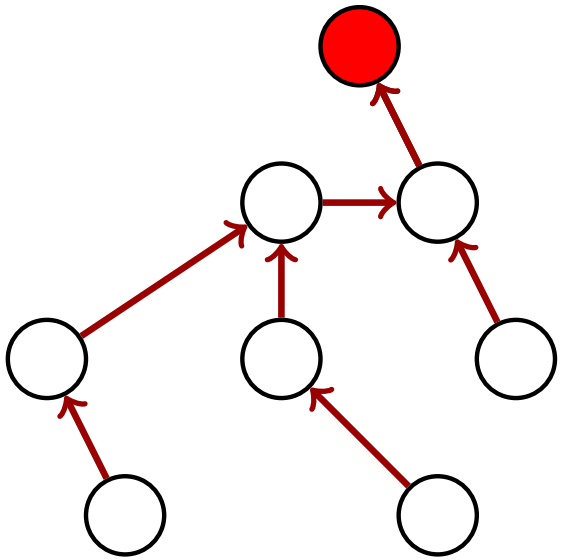
Random Spanning Forests: What, Why and **How**?



Random Spanning Forests: What, Why and **How**?



Random Spanning Forests: What, Why and **How**?



Random Spanning Forests: What, Why and **How**?

- In case of weighted graphs, Wilson's algorithm generate spanning trees distributed according to:

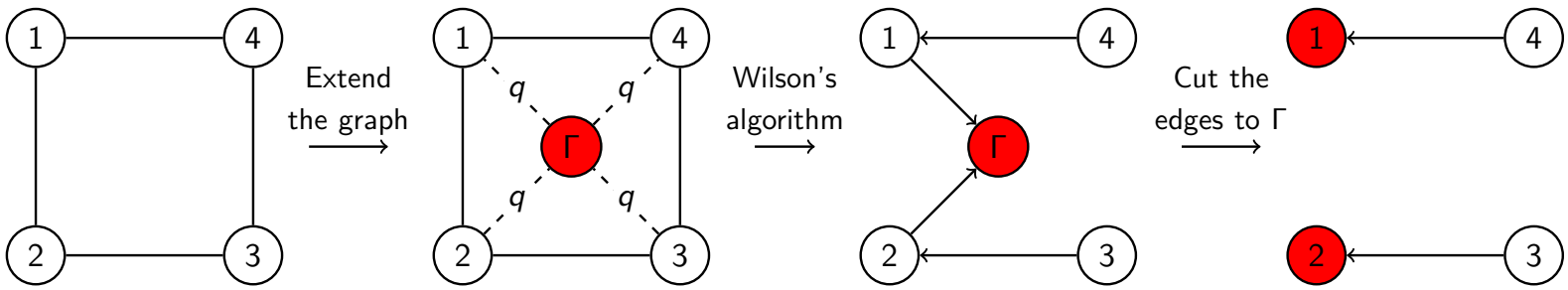
$$\mathbb{P}(T = \tau) \propto \prod_{(i,j) \in \mathcal{E}_\tau} w(i,j)$$

Random Spanning Forests: What, Why and **How**?

- In case of weighted graphs, Wilson's algorithm generate spanning trees distributed according to:

$$\mathbb{P}(T = \tau) \propto \prod_{(i,j)\in\mathcal{E}_\tau} w(i,j)$$

- Adapting Wilson's algorithm to sample RSFs is easy:

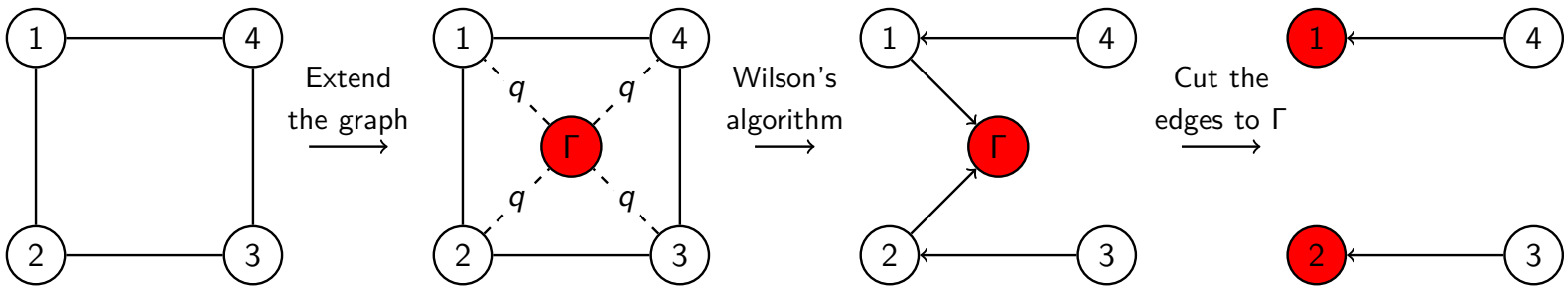


Random Spanning Forests: What, Why and **How**?

- In case of weighted graphs, Wilson's algorithm generate spanning trees distributed according to:

$$\mathbb{P}(T = \tau) \propto \prod_{(i,j)\in\mathcal{E}_\tau} w(i,j)$$

- Adapting Wilson's algorithm to sample RSFs is easy:



- The time complexity is $\mathcal{O}(\frac{|\mathcal{E}|}{q})$.

Problems

- Graph signal smoothing.
- Trace estimation.

Graph Signal Smoothing

Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$,

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{z}\in\mathbb{R}^n} q \underbrace{||\mathbf{y} - \mathbf{z}||^2}_{\text{Fidelity}} + \underbrace{\mathbf{z}^T \mathrm{L} \mathbf{z}}_{\text{Regularization}} \quad , \quad q > 0$$

where L is the graph Laplacian and $\mathbf{z}^T \mathrm{L} \mathbf{z} = \sum_{(i,j)\in\mathcal{E}} w(i,j)(z_i - z_j)^2$.

Graph Signal Smoothing

- The explicit solution to this problem is:

$$\hat{\mathbf{x}} = \mathrm{K}\mathbf{y} \text{ with } \mathrm{K} = q(\mathrm{L} + q\mathrm{I})^{-1}$$

Graph Signal Smoothing

- The explicit solution to this problem is:

$$\hat{\mathbf{x}} = \mathrm{K}\mathbf{y} \text{ with } \mathrm{K} = q(\mathrm{L} + q\mathrm{I})^{-1}$$

- Direct computation of K requires $\mathcal{O}(n^3)$ elementary operations due to the inverse.

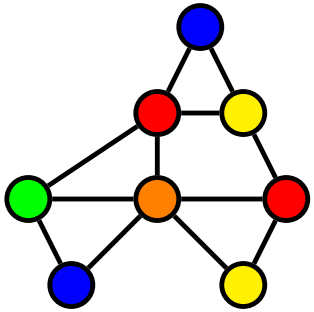
Graph Signal Smoothing

- The explicit solution to this problem is:

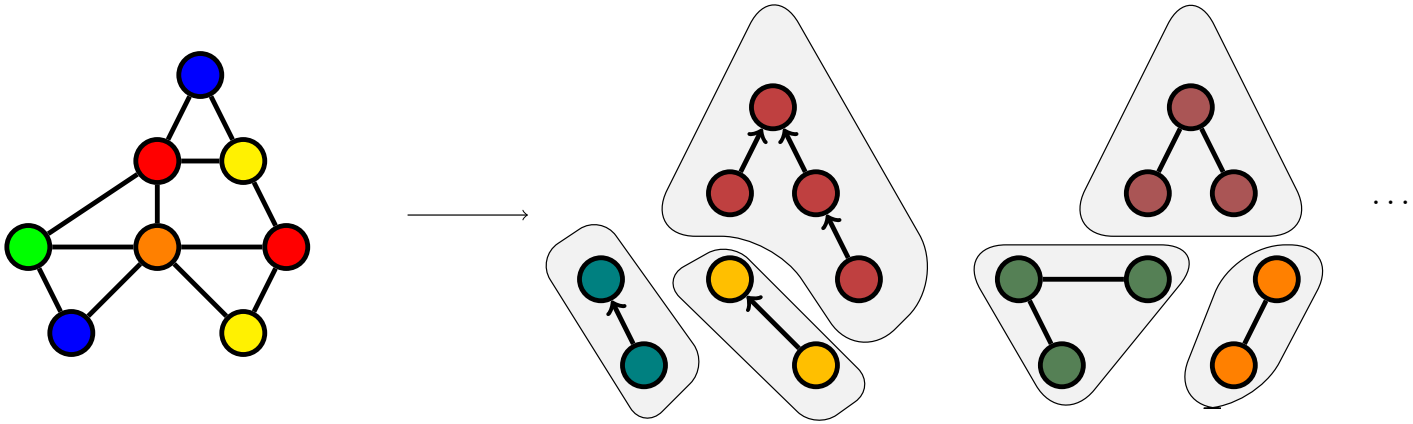
$$\hat{\mathbf{x}} = \mathrm{K}\mathbf{y} \text{ with } \mathrm{K} = q(\mathrm{L} + q\mathrm{I})^{-1}$$

- Direct computation of K requires $\mathcal{O}(n^3)$ elementary operations due to the inverse.
- For large n, iterative methods and polynomial approximations are state-of-the-art. Both compute $\hat{\mathbf{x}}$ in linear time in the number of edges $|\mathcal{E}|$.

Forest Estimator

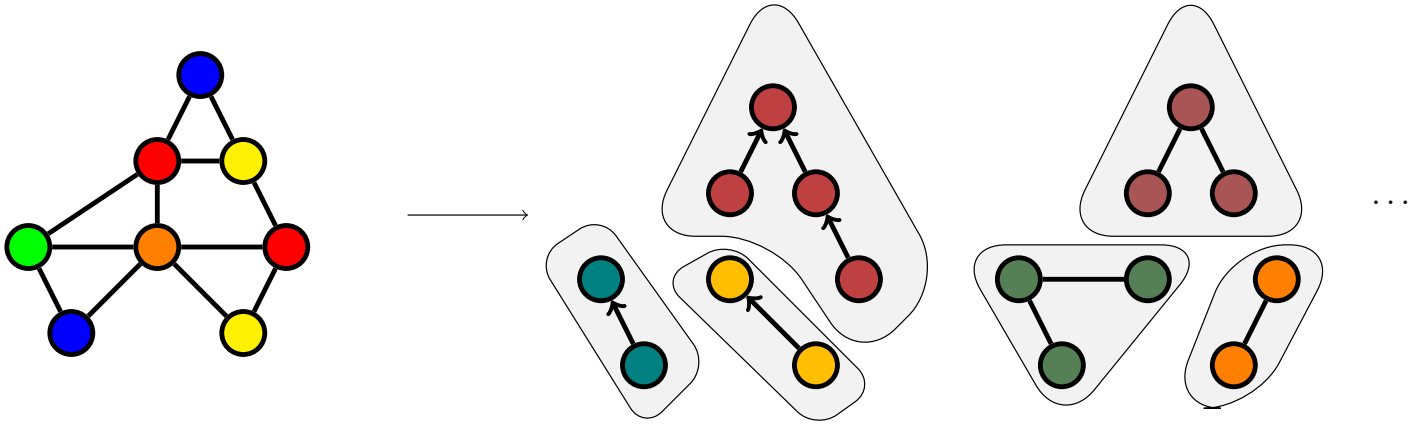


Forest Estimator



- Random partitions are sampled via random spanning forests.

Forest Estimator



- Random partitions are sampled via random spanning forests.
- This yields an unbiased estimator $\bar{\mathbf{x}}$.

Comparison with State of the art (SOTA)

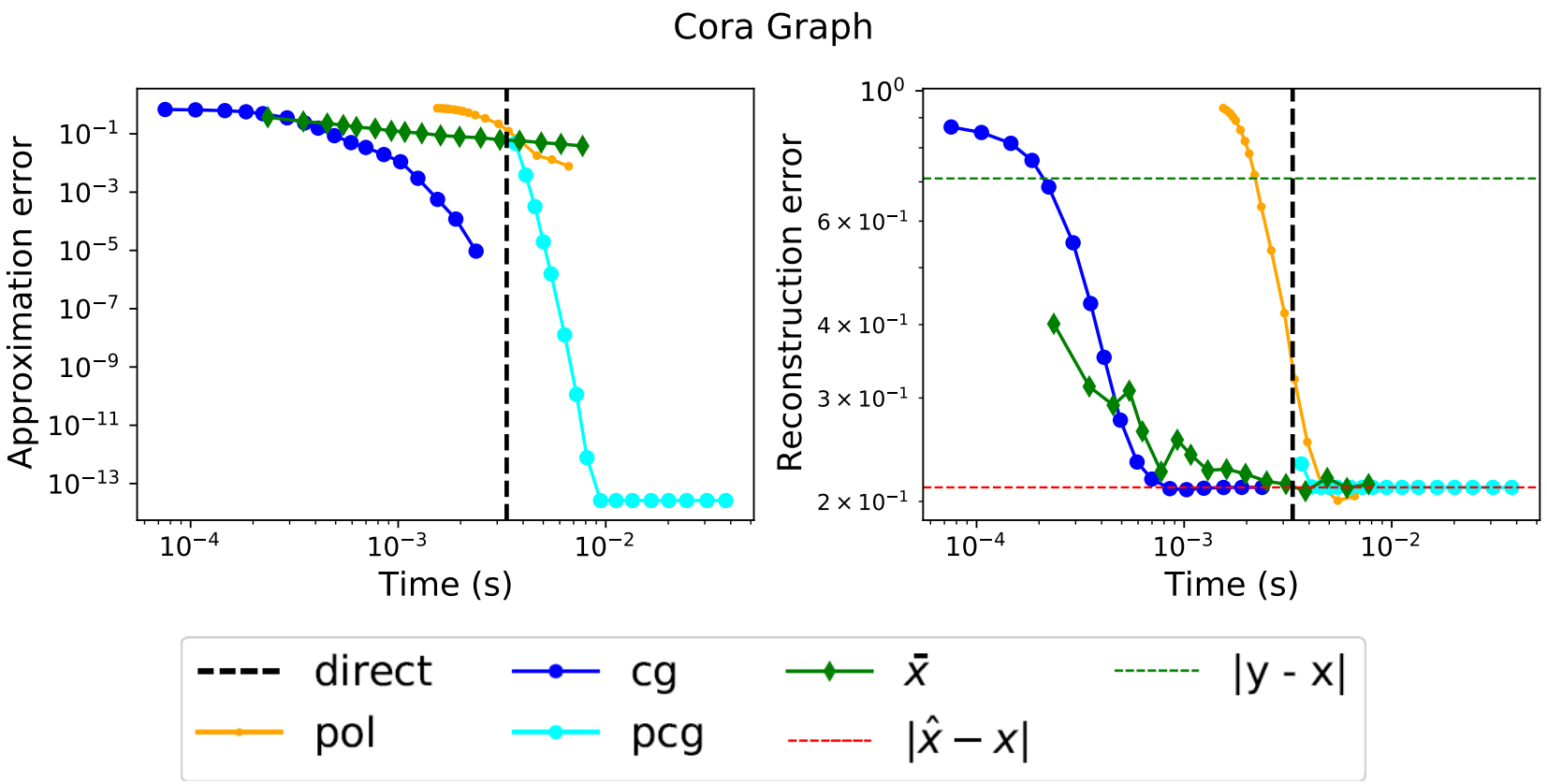
- We compare $\bar{\mathbf{x}}$ with:
  - Direct computation via Cholesky decomposition,
  - Polynomial approximation,
  - (Preconditioned) Conjugate gradient descent,

Comparison with State of the art (SOTA)

- We compare $\bar{\mathbf{x}}$ with:
  - Direct computation via Cholesky decomposition,
  - Polynomial approximation,
  - (Preconditioned) Conjugate gradient descent,
- We assume a smooth original signal $\mathbf{x}$ and noisy measurements $\mathbf{y} = \mathbf{x} + \epsilon$ with $\epsilon \in \mathcal{N}(0, \sigma)$.

Comparison with State of the art (SOTA)

- We compare all algorithms in approximation error (error respect to $\hat{\mathbf{x}}$) and reconstruction error (error respect to $\mathbf{x}$)



Variance Reduction

- Monte Carlo convergence rate $\mathcal{O}(N^{-1/2})$ might be slow.

Variance Reduction

- Monte Carlo convergence rate $\mathcal{O}(N^{-1/2})$ might be slow.
- The performance can be improved via variance reduction techniques.

Variance Reduction

- Monte Carlo convergence rate $\mathcal{O}(N^{-1/2})$ might be slow.
- The performance can be improved via variance reduction techniques.
- One technique is applicable to our case is called *control variate*.

Variance Reduction

- Monte Carlo convergence rate $\mathcal{O}(N^{-1/2})$ might be slow.
- The performance can be improved via variance reduction techniques.
- One technique is applicable to our case is called *control variate*.
- Control variate method deploys an additional random quantity which

Variance Reduction

- Monte Carlo convergence rate $\mathcal{O}(N^{-1/2})$ might be slow.
- The performance can be improved via variance reduction techniques.
- One technique is applicable to our case is called *control variate*.
- Control variate method deploys an additional random quantity which
  - is correlated with the estimated one,

Variance Reduction

- Monte Carlo convergence rate $\mathcal{O}(N^{-1/2})$ might be slow.
- The performance can be improved via variance reduction techniques.
- One technique is applicable to our case is called *control variate*.
- Control variate method deploys an additional random quantity which
  - is correlated with the estimated one,
  - has a known expectation

  to reduce the variance.

Gradient Descent Update as Control Variate

- The solution $\hat{\mathbf{x}}$ also minimizes:

$$F(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathrm{K}^{-1}\mathbf{x} - \mathbf{x}^\top \mathbf{y}.$$

Gradient Descent Update as Control Variate

- The solution $\hat{\mathbf{x}}$ also minimizes:

$$F(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathrm{K}^{-1}\mathbf{x} - \mathbf{x}^\top \mathbf{y}.$$

- The gradient descent algorithm draws the following iteration scheme:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \nabla F(\mathbf{x}_k)$$

where $\alpha \in \mathbb{R}$ and $\nabla F(\mathbf{x}_k) = \mathrm{K}^{-1}\mathbf{x}_k - \mathbf{y}$.

Gradient Descent Update as Control Variate

- The solution $\hat{\mathbf{x}}$ also minimizes:

$$F(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathrm{K}^{-1}\mathbf{x} - \mathbf{x}^\top \mathbf{y}.$$

- The gradient descent algorithm draws the following iteration scheme:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \nabla F(\mathbf{x}_k)$$

  where $\alpha \in \mathbb{R}$ and $\nabla F(\mathbf{x}_k) = \mathrm{K}^{-1}\mathbf{x}_k - \mathbf{y}$.
- We propose to apply the gradient descent update on the previous estimator $\bar{\mathbf{x}}$:

$$\bar{\mathbf{z}} := \bar{\mathbf{x}} - \alpha(\mathrm{K}^{-1}\bar{\mathbf{x}} - \mathbf{y})$$

Properties of this estimator

- $\bar{\mathbf{z}}$ is unbiased.

Properties of this estimator

- $\bar{\mathbf{z}}$ is unbiased.
- A matrix-vector product with L is needed only once.

Properties of this estimator

- $\bar{\mathbf{z}}$ is unbiased.
- A matrix-vector product with L is needed only once.
- For certain values of α, we have improved performance.

Properties of this estimator

- $\bar{\mathbf{z}}$ is unbiased.
- A matrix-vector product with L is needed only once.
- For certain values of α, we have improved performance.
- The optimal value is:

$$\alpha^\star = \frac{\mathrm{tr}(\mathrm{Cov}(\mathrm{K}^{-1}\bar{\mathbf{x}}, \bar{\mathbf{x}}))}{\mathrm{tr}(\mathrm{Var}(\mathrm{K}^{-1}\bar{\mathbf{x}}))}.$$

Properties of this estimator

- $\bar{\mathbf{z}}$ is unbiased.
- A matrix-vector product with L is needed only once.
- For certain values of α, we have improved performance.
- The optimal value is:

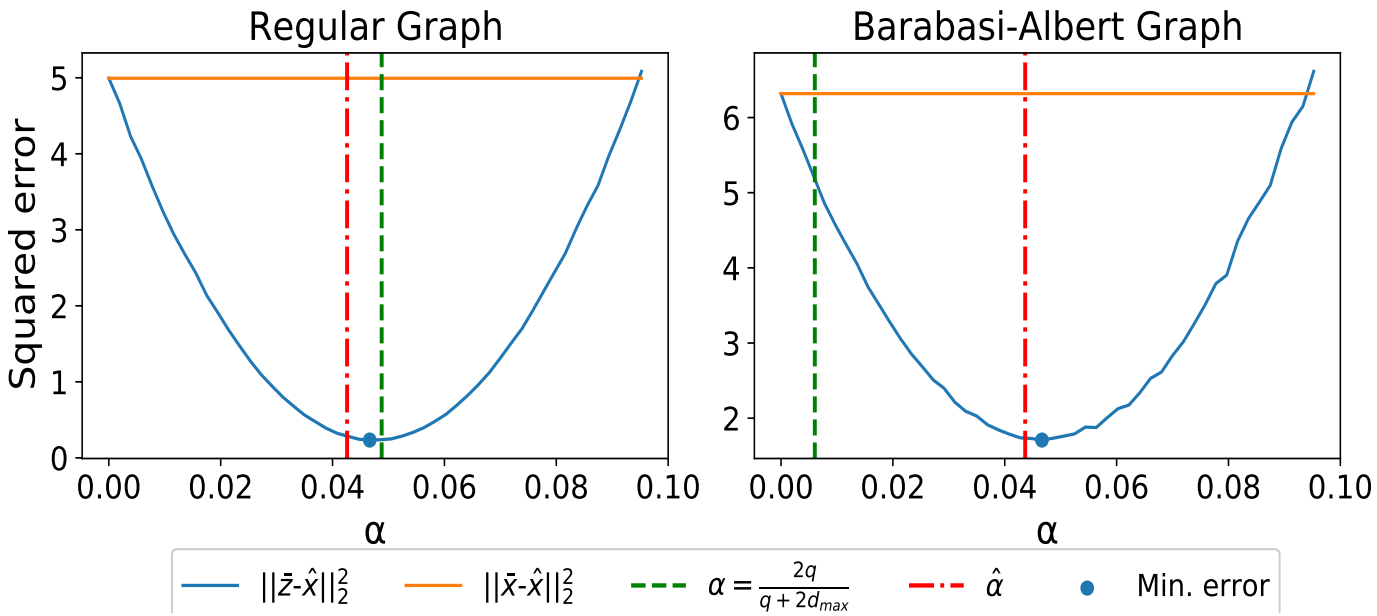
$$\alpha^{\star} = \frac{\mathrm{tr}(\mathrm{Cov}(\mathrm{K}^{-1}\bar{\mathbf{x}}, \bar{\mathbf{x}}))}{\mathrm{tr}(\mathrm{Var}(\mathrm{K}^{-1}\bar{\mathbf{x}}))}.$$

- One can either choose a value for α from the safe range (*e.g.* $\alpha = \frac{2q}{q+2d_{max}}$) or estimate from the samples:

$$\hat{\alpha} = \frac{\mathrm{tr}(\widehat{\mathrm{Cov}}(\mathrm{K}^{-1}\bar{\mathbf{x}}, \bar{\mathbf{x}}))}{\mathrm{tr}(\widehat{\mathrm{Var}}(\mathrm{K}^{-1}\bar{\mathbf{x}}))}.$$

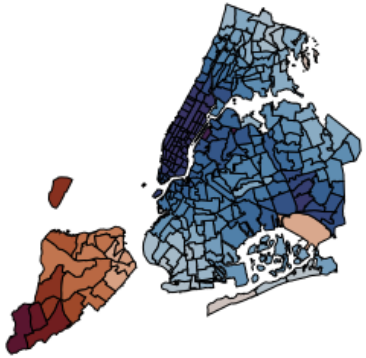
Range of α

- We empirically compare these options of α over a regular and irregular graph:

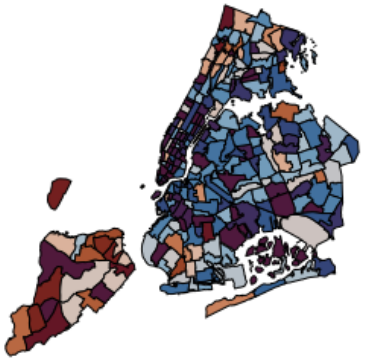


An Illustration

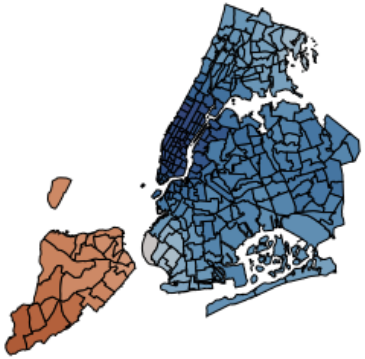
$\mathbf{x}$:



$\mathbf{y}$:

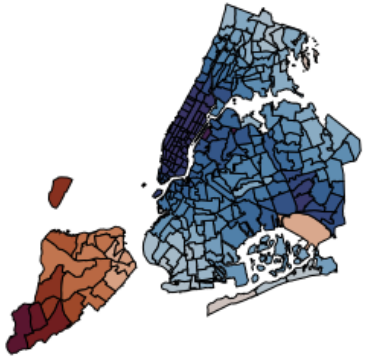


$\hat{\mathbf{x}}$:

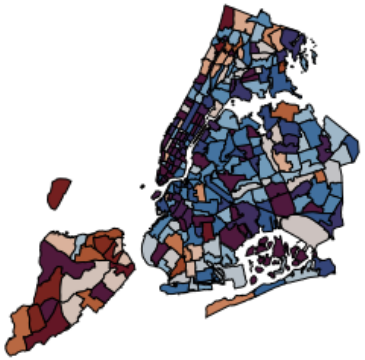


An Illustration

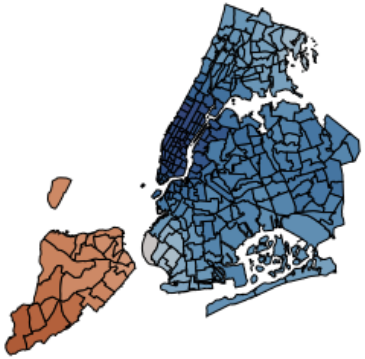
$\mathbf{x}$:



$\mathbf{y}$:



$\hat{\mathbf{x}}$:



An Illustration

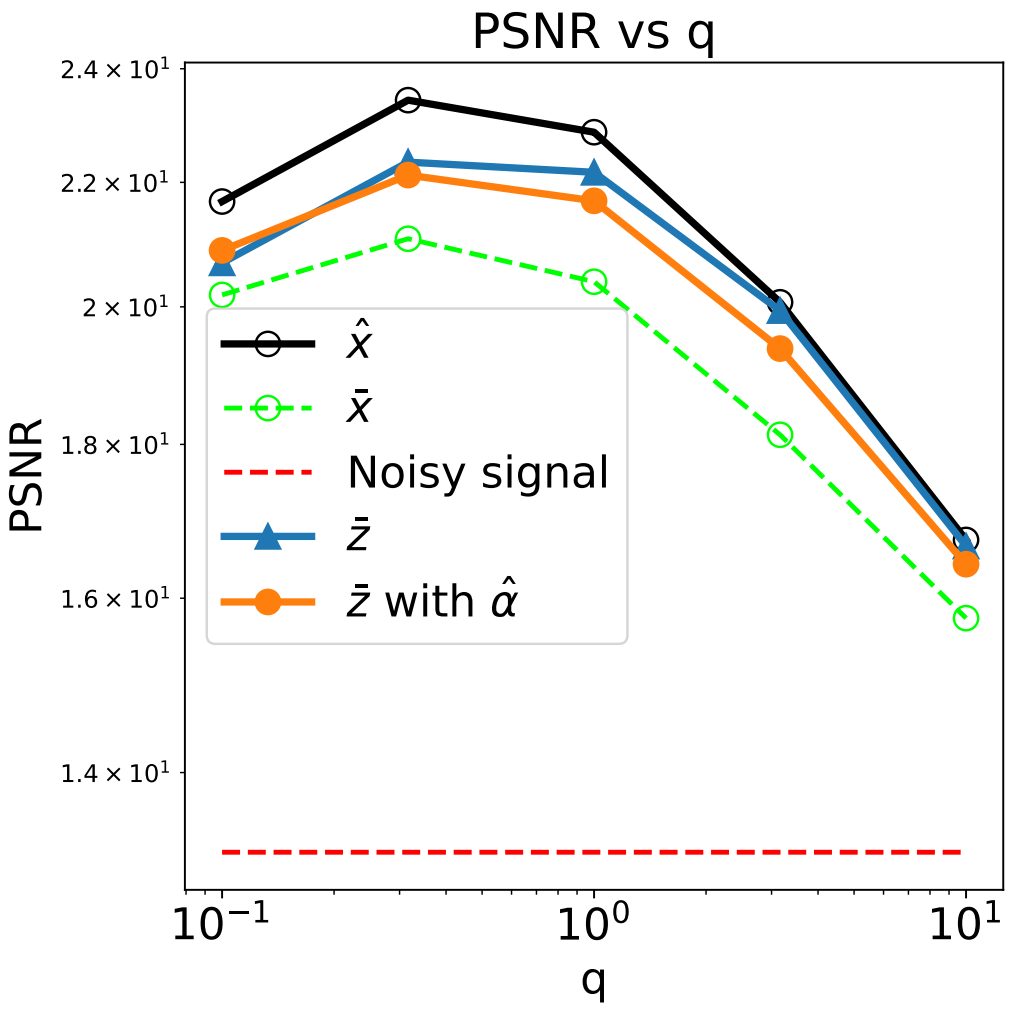


Figure: PSNR vs q, N=2

Inverse Trace Estimation: A Motivation

- How to choose a good value for the hyperparameter q?

Inverse Trace Estimation: A Motivation

- How to choose a good value for the hyperparameter q?
- There are several methods such as Akaike's or Bayesian information criterion, generalized cross validation or Stein's unbiased risk estimator.

Inverse Trace Estimation: A Motivation

- How to choose a good value for the hyperparameter q?
- There are several methods such as Akaike's or Bayesian information criterion, generalized cross validation or Stein's unbiased risk estimator.
- Each uses a quantity called the effective degree of freedom which is equal to tr(K).

Inverse Trace Estimation: SOTA

- SOTA for estimating $\mathrm{tr}(K)$ is Hutchinson's estimator.

Inverse Trace Estimation: SOTA

- SOTA for estimating tr(K) is Hutchinson's estimator.
- It is an unbiased estimator of $\mathrm{tr}(K)$.

Inverse Trace Estimation: SOTA

- SOTA for estimating tr(K) is Hutchinson's estimator.
- It is an unbiased estimator of $\mathrm{tr}(K)$.
- For some samples of a random vector $\mathbf{a}$, it needs to calculate $\mathbf{a}^T \mathrm{K} \mathbf{a}$.

Inverse Trace Estimation: SOTA

- SOTA for estimating tr(K) is Hutchinson's estimator.
- It is an unbiased estimator of $\mathrm{tr}(K)$.
- For some samples of a random vector $\mathbf{a}$, it needs to calculate $\mathbf{a}^T \mathrm{K}\mathbf{a}$.
- Solving for $\mathrm{K}\mathbf{a}$ can be done via:
  - Direct computation via Cholesky decomposition
  - (Preconditioned) Iterative solvers
  - Algebraic Multigrid solvers
  - ...

Forest based Trace Estimator

- Another unbiased estimator is by RSFs[Barthelmé et al.]:

$$s := |\rho(\Phi_q)| \text{ with } \mathbb{E}[s] = \mathrm{tr}(\mathrm{K})$$

Forest based Trace Estimator

- Another unbiased estimator is by RSFs[Barthelmé et al.]:

$$s := |\rho(\Phi_q)| \text{ with } \mathbb{E}[s] = \mathrm{tr}(K)$$

- This estimator gives an comparable perforance with SOTA algorithms.

Forest based Trace Estimator

- Another unbiased estimator is by RSFs[Barthelmé et al.]:

$$s := |\rho(\Phi_q)| \text{ with } \mathbb{E}[s] = \mathrm{tr}(K)$$

- This estimator gives an comparable perforance with SOTA algorithms.
- One can use this estimator in case of symmetric diagonally dominant matrices instead of the graph Laplacians.

Variance Reduction via Control Variates

- We give two ways of reducing the expected error of s.

Variance Reduction via Control Variates

- We give two ways of reducing the expected error of s.
- The first one is by adapting the control variate that is previously presented.

Variance Reduction via Control Variates

- We give two ways of reducing the expected error of s.
- The first one is by adapting the control variate that is previously presented.
- Define the random matrix $\bar{\mathsf{S}}$ such that $\bar{\mathbf{x}} = \bar{\mathsf{S}}\mathbf{y}$.

Variance Reduction via Control Variates

- We give two ways of reducing the expected error of s.
- The first one is by adapting the control variate that is previously presented.
- Define the random matrix $\bar{\mathsf{S}}$ such that $\bar{\mathbf{x}} = \bar{\mathsf{S}}\mathbf{y}$.
- $\bar{\mathsf{S}}$ is unbiased for estimating K. Coupling with the control variate, one has another estimator:

$$\bar{\mathsf{Z}} = \bar{\mathsf{S}} - \alpha(\mathsf{K}^{-1}\bar{\mathsf{S}} - \mathsf{I}).$$

Variance Reduction via Control Variates

- We give two ways of reducing the expected error of s.
- The first one is by adapting the control variate that is previously presented.
- Define the random matrix $\bar{\mathsf{S}}$ such that $\bar{\mathbf{x}} = \bar{\mathsf{S}}\mathbf{y}$.
- $\bar{\mathsf{S}}$ is unbiased for estimating K. Coupling with the control variate, one has another estimator:

$$\bar{\mathsf{Z}} = \bar{\mathsf{S}} - \alpha(\mathsf{K}^{-1}\bar{\mathsf{S}} - \mathsf{I}).$$

- We define the new trace estimator as

$$\bar{s} := \mathrm{tr}(\bar{\mathsf{Z}}).$$

Variance Reduction via Control Variates

- We give two ways of reducing the expected error of s.
- The first one is by adapting the control variate that is previously presented.
- Define the random matrix $\bar{\mathsf{S}}$ such that $\bar{\mathbf{x}} = \bar{\mathsf{S}}\mathbf{y}$.
- $\bar{\mathsf{S}}$ is unbiased for estimating K. Coupling with the control variate, one has another estimator:

$$\bar{\mathsf{Z}} = \bar{\mathsf{S}} - \alpha(\mathsf{K}^{-1}\bar{\mathsf{S}} - \mathsf{I}).$$

- We define the new trace estimator as

$$\bar{s} := \mathrm{tr}(\bar{\mathsf{Z}}).$$

- A safe value of α is $\frac{2q}{q+d_{max}}$. We also observe that $\frac{q}{q+d_{avg}}$ is usually a good estimate of $\alpha^\star$.

Variance Reduction via Stratification

- Stratification reduces the Monte Carlo error by dividing the sample space into sub-parts, each called a stratum, based on another random variable.

Variance Reduction via Stratification

- Stratification reduces the Monte Carlo error by dividing the sample space into sub-parts, each called a stratum, based on another random variable.
- Stratified sampling can substantially decrease approximation error when applicable.

Variance Reduction via Stratification

- Stratification reduces the Monte Carlo error by dividing the sample space into sub-parts, each called a stratum, based on another random variable.
- Stratified sampling can substantially decrease approximation error when applicable.
- Thanks to rich theory of RSFs, we can apply stratified sampling on s.

Comparison with SOTA

- We compare the time needed by the estimators for reaching a certain accuracy.

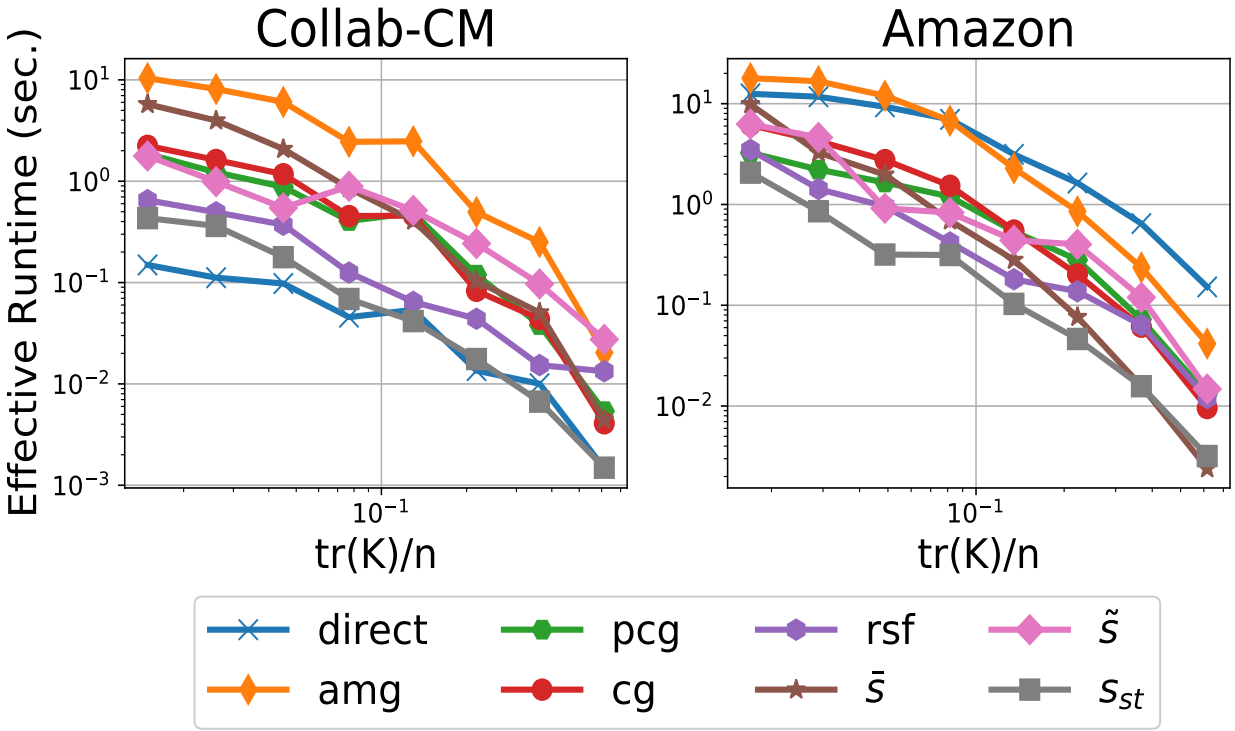


Figure: Effective Runtime vs $\mathrm{tr}(K)/n$.

Conclusion

- We propose an alternative RLA perspective for Laplacian-based numerical algebra.

Conclusion

- We propose an alternative RLA perspective for Laplacian-based numerical algebra.
- This perspective leverages fascinating links between the graph Laplacians and RSFs, yielding applications in graph signal smoothing and inverse trace estimation.

Conclusion

- We propose an alternative RLA perspective for Laplacian-based numerical algebra.
- This perspective leverages fascinating links between the graph Laplacians and RSFs, yielding applications in graph signal smoothing and inverse trace estimation.
- RSFs are like the gifts that keep on giving...

Conclusion

- We propose an alternative RLA perspective for Laplacian-based numerical algebra.
- This perspective leverages fascinating links between the graph Laplacians and RSFs, yielding applications in graph signal smoothing and inverse trace estimation.
- RSFs are like the gifts that keep on giving...
- Estimating effective resistances.

$$\mathrm{R}_{i,j} = \mathrm{L}^{\dagger}_{i,i} + \mathrm{L}^{\dagger}_{j,j} - \mathrm{L}^{\dagger}_{i,j} - \mathrm{L}^{\dagger}_{j,i}.$$

Questions

Thanks! Questions?

Barthelmé, Simon et al. (2019). "Estimating the inverse trace using random forests on graphs". In: *arXiv preprint arXiv:1905.02086*.

Martinsson, Per-Gunnar and Joel A Tropp (2020). "Randomized numerical linear algebra: Foundations and algorithms". In: *Acta Numerica* 29, pp. 403–572.

Shuman, David I et al. (2013). "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains". In: *IEEE signal processing magazine* 30.3, pp. 83–98.

Wu, Zonghan et al. (2020). "A comprehensive survey on graph neural networks". In: *IEEE transactions on neural networks and learning systems* 32.1, pp. 4–24.

Zhu, Xiaojin (2005). *Semi-supervised learning with graphs*. Carnegie Mellon University.