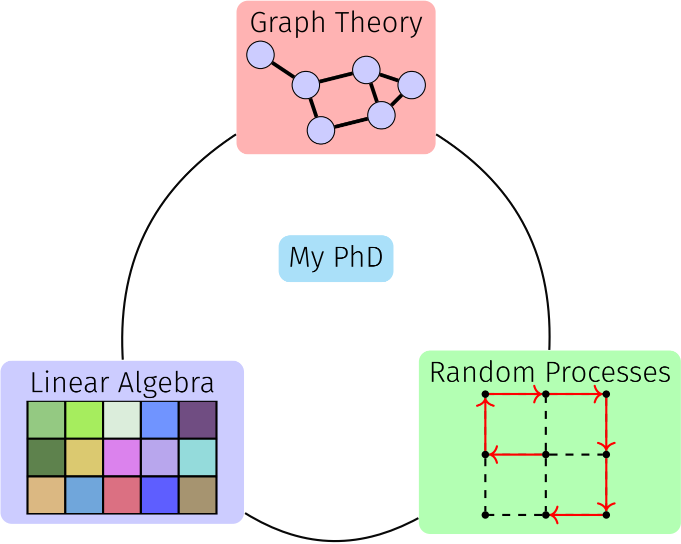
Wilson's Algorithm for Randomized Linear Algebra

Yusuf Yiğit Pilavcı

Advisors:
Pierre-Olivier Amblard
Simon Barthelmé
Nicolas Tremblay

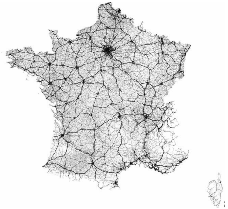
15/11/2022

What's inside?



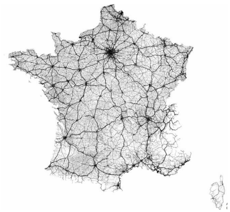
Graphs are *Ubiquitous*...

Graphs are *Ubiquitous*...

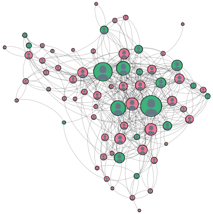


Road Networks

Graphs are *Ubiquitous*...

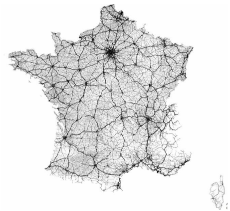


Road Networks

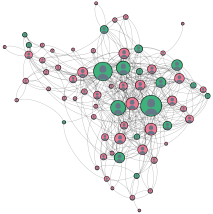


Social Networks

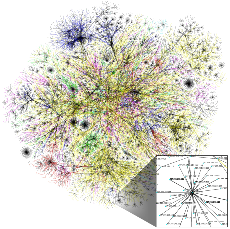
GRAPHS ARE *UBIQUITOUS*...



Road Networks

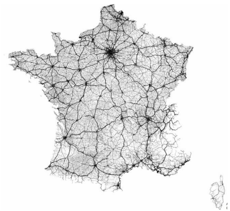


Social Networks

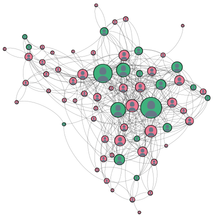


Internet

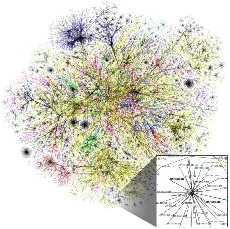
Graphs are *Ubiquitous*...



Road Networks



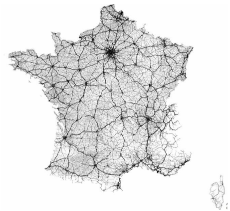
Social Networks



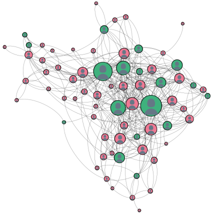
Internet

Brain Networks

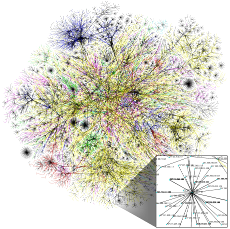
GRAPHS ARE *UBIQUITOUS*...



Road Networks

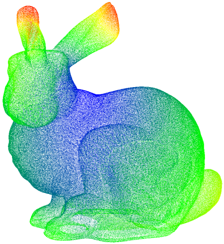


Social Networks



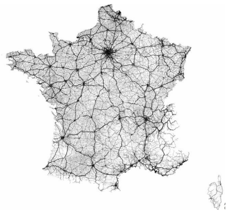
Internet

Brain Networks

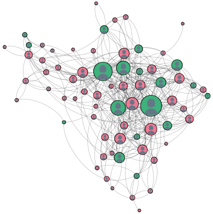


Point Clouds Networks

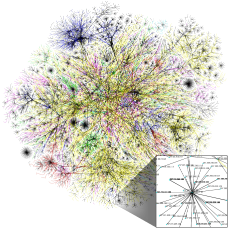
Graphs are *Ubiquitous*...



Road Networks

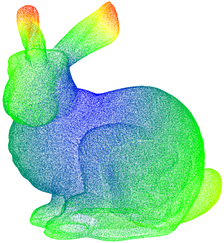


Social Networks

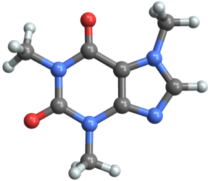


Internet

Brain Networks

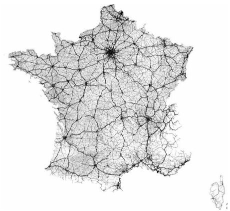


Point Clouds Networks

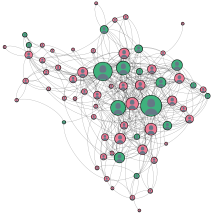


Molecule Networks

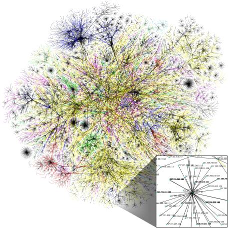
Graphs are *Ubiquitous*...



Road Networks

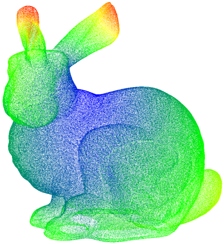


Social Networks

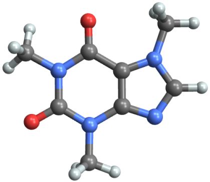


Internet

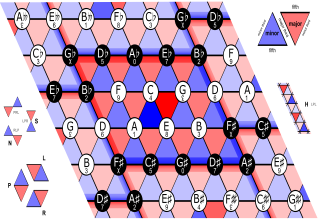
Brain Networks



Point Clouds Networks

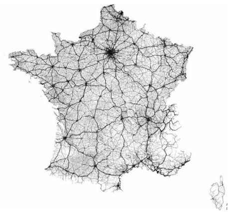


Molecule Networks

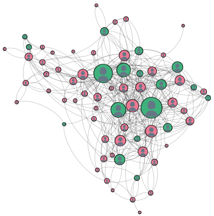


Tonnetz

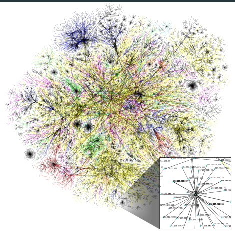
Graphs are *Ubiquitous*...



Road Networks

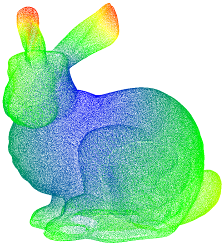


Social Networks

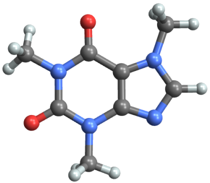


Internet

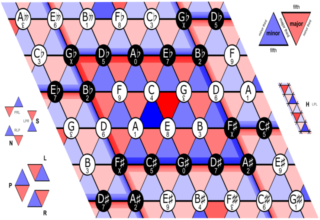
Brain Networks



Point Clouds Networks



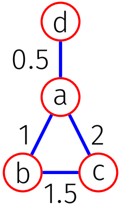
Molecule Networks



Tonnetz

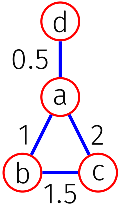
...

Graph related Linear Algebra



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

Graph related Linear Algebra

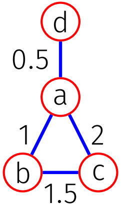


$\mathcal{G} = ({\color{red}\mathcal{V}}, {\color{blue}\mathcal{E}}, \mathrm{w})$

$$\begin{array}{cc} & \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{array}{c} a \\ b \\ c \\ d \end{array} & \begin{bmatrix} 0 & 1 & 2 & 0.5 \\ 1 & 0 & 1.5 & 0 \\ 2 & 1.5 & 0 & 0 \\ 0.5 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Adjacency matrix W

Graph related Linear Algebra



$\mathcal{G} = (\color{red}{\mathcal{V}}, \color{blue}{\mathcal{E}}, \mathrm{w})$

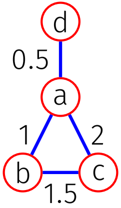
$$\begin{array}{c c} & \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{array}{c} a \\ b \\ c \\ d \end{array} & \begin{bmatrix} 0 & 1 & 2 & 0.5 \\ 1 & 0 & 1.5 & 0 \\ 2 & 1.5 & 0 & 0 \\ 0.5 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

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

Graph related Linear Algebra



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

$$\begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & 1 & 2 & 0.5 \\ b & 1 & 0 & 1.5 & 0 \\ c & 2 & 1.5 & 0 & 0 \\ d & 0.5 & 0 & 0 & 0 \end{array}$$

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

$L = D - W$

The Graph Laplacian is *Ubiquitous*...

The Graph Laplacian is *Ubiquitous*...

Theory

- Connectivity Analysis
- Graph Partitioning
- Spanning Trees
- Random Walks (Loop-Erased)...

The Graph Laplacian is *Ubiquitous*...

Theory

- Connectivity Analysis
- Graph Partitioning
- Spanning Trees
- Random Walks (Loop-Erased)...

Applications

- Graph Signal Processing
- Machine learning
- Visualization
- Sparsification
- Robustness analysis...

The Graph Laplacian is *Ubiquitous*...

Theory

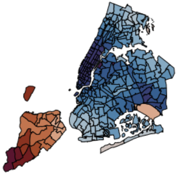
- Connectivity Analysis
- Graph Partitioning
- Spanning Trees
- Random Walks (Loop-Erased)...

Applications

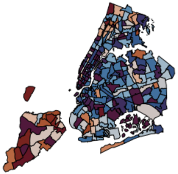
- Graph Signal Processing
- Machine learning
- Visualization
- Sparsification
- Robustness analysis...

⚠ However, some computations do not scale with large graphs.

Original Signal:



$\mathbf{y}$:



Original Signal: $\quad y$: $\quad \hat{x}$:

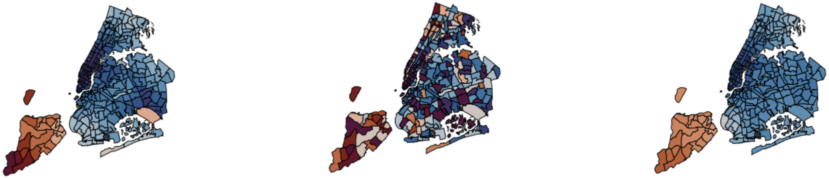


Figure 3: Median taxi fees paid in drop-off locations in NYC

Graph Signal Smoothing

Original Signal: $\mathbf{y}$: $\hat{\mathbf{x}}$:

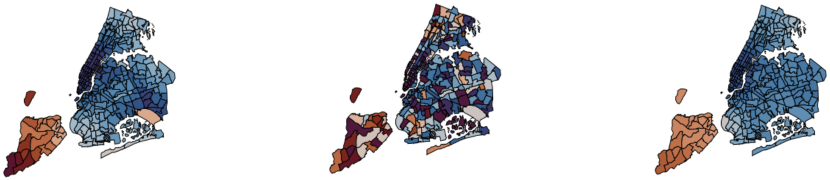


Figure 3: Median taxi fees paid in drop-off locations in NYC

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

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}\in\mathbb{R}^n} q \underbrace{||\mathbf{y}-\mathbf{x}||_2^2}_{\text{Fidelity}} + \underbrace{\mathbf{x}^T L \mathbf{x}}_{\text{Regularization}}, \quad q > 0$$

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

- 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:

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

- Besides smoothing, this solution plays a role as building block in solving many other graph related-problems.

- The explicit solution to this problem is:

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

- Besides smoothing, this solution plays a role as building block in solving many other graph related-problems.
- 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:

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

- Besides smoothing, this solution plays a role as building block in solving many other graph related-problems.
- 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.

Graph Signal Smoothing

- The explicit solution to this problem is:

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

- Besides smoothing, this solution plays a role as building block in solving many other graph related-problems.
- 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.
- For SDD linear systems, there is a growing body of works starting from (Spielman and Teng 2004).

Inverse Trace Estimation

- Trace is an essential operation:

$$\boxed{\mathrm{tr}(\mathsf{A}) = \sum_{i=1}^{n} \delta_i^\top \mathsf{A} \delta_i}$$

Inverse Trace Estimation

- Trace is an essential operation:

$$\boxed{\operatorname{tr}(A) = \sum_{i=1}^{n} \delta_i^\top A \delta_i}$$

$$\boxed{\operatorname{tr}(A^{-1}) = \sum_{i=1}^{n} \delta_i^\top A^{-1} \delta_i}$$

Inverse Trace Estimation

- Trace is an essential operation:

$$\mathrm{tr}(A) = \sum_{i=1}^{n} \delta_i^\top A \delta_i$$

$$\mathrm{tr}(A^{-1}) = \sum_{i=1}^{n} \delta_i^\top A^{-1} \delta_i$$

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

Inverse Trace Estimation

- Trace is an essential operation:

$$\boxed{\operatorname{tr}(A) = \sum_{i=1}^{n} \delta_i^\top A \delta_i}$$

$$\boxed{\operatorname{tr}(A^{-1}) = \sum_{i=1}^{n} \delta_i^\top A^{-1} \delta_i}$$

- 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.

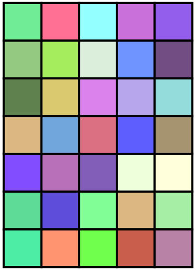
Inverse Trace Estimation

- Trace is an essential operation:

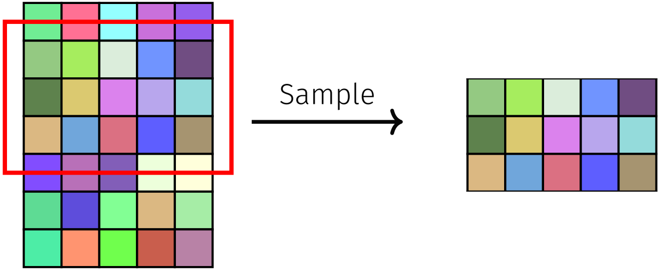
$$\boxed{\mathrm{tr}(A) = \sum_{i=1}^{n} \delta_i^\top A \delta_i} \qquad \boxed{\mathrm{tr}(A^{-1}) = \sum_{i=1}^{n} \delta_i^\top A^{-1} \delta_i}$$

- 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 $\mathrm{tr}(K)$.

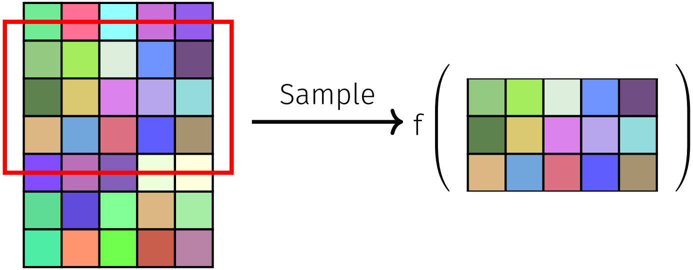
- RLA is a branch of numerical linear algebra developing Monte Carlo methods.



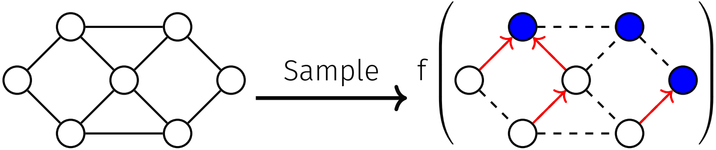
- RLA is a branch of numerical linear algebra developing Monte Carlo methods.



- RLA is a branch of numerical linear algebra developing Monte Carlo methods.



- RLA algorithms for Laplacian-based numerical algebra by using Random Spanning Forests.



Outline

Random Spanning Forests (RSF)

RSF-based Algorithms

Conclusion

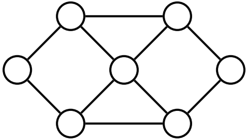
Outline

Random Spanning Forests (RSF)

RSF-based Algorithms

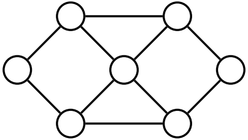
Conclusion

Spanning Forests

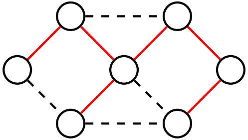


Graph

Spanning Forests

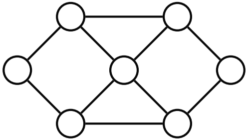


Graph

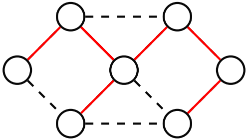


Spanning Tree

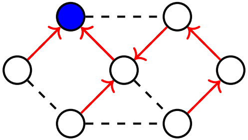
Spanning Forests



Graph

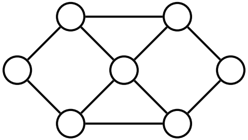


Spanning Tree

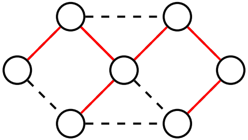


Rooted
Spanning Tree

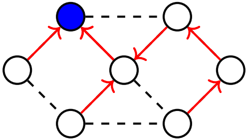
Spanning Forests



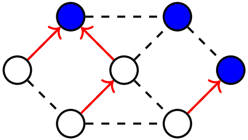
Graph



Spanning Tree



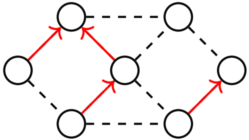
Rooted
Spanning Tree



Rooted
Spanning Forest

Forest Notations

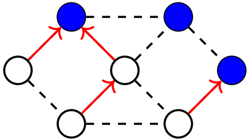
- Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathrm{w})$, we denote:



  - a spanning forest by ϕ and

Forest Notations

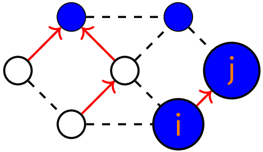
- Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathrm{w})$, we denote:



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

Forest Notations

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



  - a spanning forest by ϕ and its root set by $\rho(\phi)$,
  - the root of vertex i in ϕ by $\mathrm{r}_\phi(\mathrm{i}) = \mathrm{j}$.

- Random spanning forests is the process of selecting a forest at random over all possible ones.

- Random spanning forests is the process of selecting a forest at random over all possible ones.

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 is the process of selecting a forest at random over all possible ones.

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.

- The random roots $\rho(\Phi_q)$ is a determinantal point process with a marginal kernel $K = q(L + qI)^{-1}$ (Avena et al. 2018):

$$\forall S \subseteq \mathcal{V}, \quad \mathbb{P}(S \subseteq \rho(\Phi_q)) = \det K_S.$$

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

- The random roots $\rho(\Phi_q)$ is a determinantal point process with a marginal kernel $K = q(L + qI)^{-1}$ (Avena et al. 2018):

$$\forall S \subseteq \mathcal{V}, \quad \mathbb{P}(S \subseteq \rho(\Phi_q)) = \det K_S.$$

- Moreover, we have the following identity (Avena et al. 2018):

$$\forall i, j \in \mathcal{V}, \quad \mathbb{P}(r_{\Phi_q}(i) = j) = K_{i,j}.$$

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

- The random roots $\rho(\Phi_q)$ is a determinantal point process with a marginal kernel $K = q(L + qI)^{-1}$ (Avena et al. 2018):

$$\forall S \subseteq \mathcal{V}, \quad \mathbb{P}(S \subseteq \rho(\Phi_q)) = \det K_S.$$

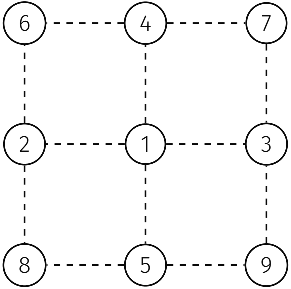
- Moreover, we have the following identity (Avena et al. 2018):

$$\forall i, j \in \mathcal{V}, \quad \mathbb{P}(r_{\Phi_q}(i) = j) = K_{i,j}.$$

- There is an efficient algorithm to sample RSFs, called Wilson's algorithm (Wilson 1996).

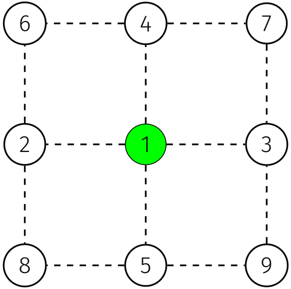
- Consider an simple random walk on $\mathcal{G}$ with the transition rule:
    - take a step from i to j with probability $\frac{w(i,j)}{q+d_i}$,
    - interrupt at any node i with a probability $\frac{q}{q+d_i}$.

- Consider an simple random walk on $\mathcal{G}$ with the transition rule:
  - take a step from i to j with probability $\frac{w(i,j)}{q+d_i}$,
  - interrupt at any node i with a probability $\frac{q}{q+d_i}$.

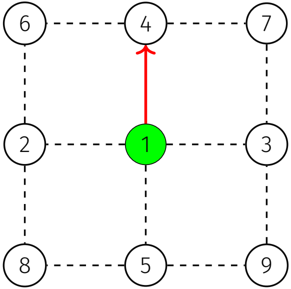


Random Spanning Forests: What, Why and **How**?

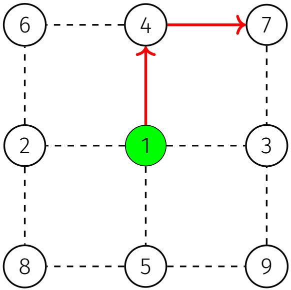
- Consider an simple random walk on $\mathcal{G}$ with the transition rule:
  - take a step from i to j with probability $\frac{w(i,j)}{q+d_i}$,
  - interrupt at any node i with a probability $\frac{q}{q+d_i}$.



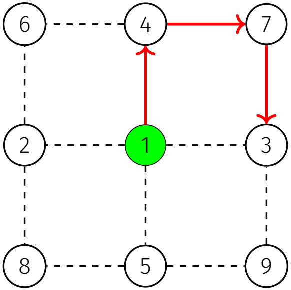
- Consider an simple random walk on $\mathcal{G}$ with the transition rule:
  - take a step from i to j with probability $\frac{w(i,j)}{q+d_i}$,
  - interrupt at any node i with a probability $\frac{q}{q+d_i}$.



- Consider an simple random walk on $\mathcal{G}$ with the transition rule:
  - take a step from i to j with probability $\frac{w(i,j)}{q+d_i}$,
  - interrupt at any node i with a probability $\frac{q}{q+d_i}$.

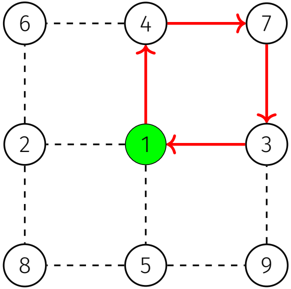


- Consider an simple random walk on $\mathcal{G}$ with the transition rule:
  - take a step from i to j with probability $\frac{w(i,j)}{q+d_i}$,
  - interrupt at any node i with a probability $\frac{q}{q+d_i}$.

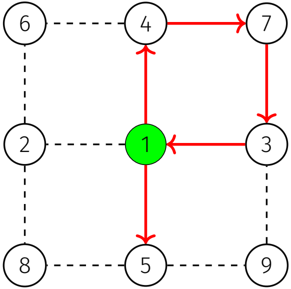


Random Spanning Forests: What, Why and **How**?

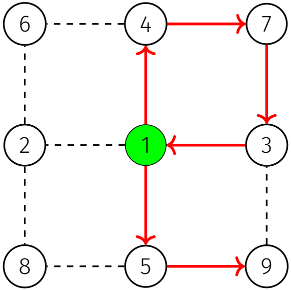
- Consider an simple random walk on $\mathcal{G}$ with the transition rule:
  - take a step from i to j with probability $\frac{w(i,j)}{q+d_i}$,
  - interrupt at any node i with a probability $\frac{q}{q+d_i}$.



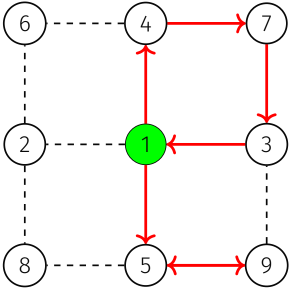
- Consider an simple random walk on $\mathcal{G}$ with the transition rule:
  - take a step from i to j with probability $\frac{w(i,j)}{q+d_i}$,
  - interrupt at any node i with a probability $\frac{q}{q+d_i}$.



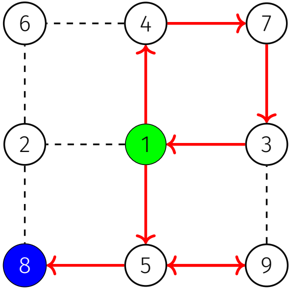
- Consider an simple random walk on $\mathcal{G}$ with the transition rule:
  - take a step from i to j with probability $\frac{w(i,j)}{q+d_i}$,
  - interrupt at any node i with a probability $\frac{q}{q+d_i}$.



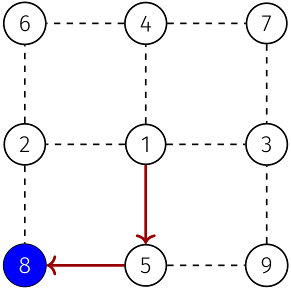
- Consider an simple random walk on $\mathcal{G}$ with the transition rule:
  - take a step from i to j with probability $\frac{w(i,j)}{q+d_i}$,
  - interrupt at any node i with a probability $\frac{q}{q+d_i}$.



- Consider an simple random walk on $\mathcal{G}$ with the transition rule:
  - take a step from i to j with probability $\frac{w(i,j)}{q+d_i}$,
  - interrupt at any node i with a probability $\frac{q}{q+d_i}$.

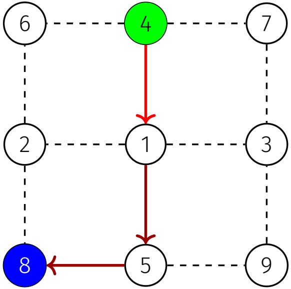


- Consider an simple random walk on $\mathcal{G}$ with the transition rule:
  - take a step from i to j with probability $\frac{w(i,j)}{q+d_i}$,
  - interrupt at any node i with a probability $\frac{q}{q+d_i}$.

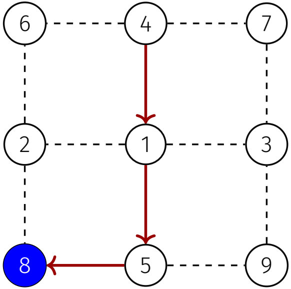


Random Spanning Forests: What, Why and **How?**

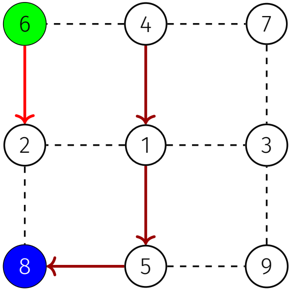
- Consider an simple random walk on $\mathcal{G}$ with the transition rule:
    - take a step from i to j with probability $\frac{w(i,j)}{q+d_i}$,
    - interrupt at any node i with a probability $\frac{q}{q+d_i}$.



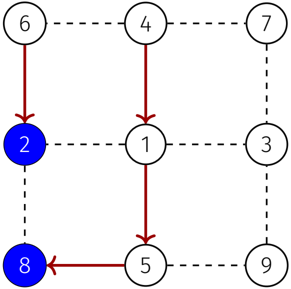
- Consider an simple random walk on $\mathcal{G}$ with the transition rule:
  - take a step from i to j with probability $\frac{w(i,j)}{q+d_i}$,
  - interrupt at any node i with a probability $\frac{q}{q+d_i}$.



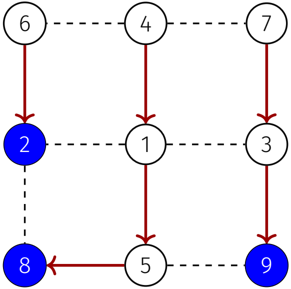
- Consider an simple random walk on $\mathcal{G}$ with the transition rule:
  - take a step from i to j with probability $\frac{w(i,j)}{q+d_i}$,
  - interrupt at any node i with a probability $\frac{q}{q+d_i}$.



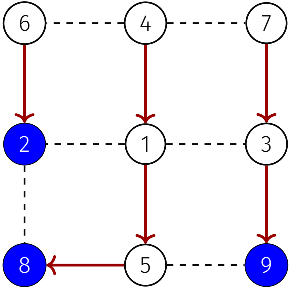
- Consider an simple random walk on $\mathcal{G}$ with the transition rule:
  - take a step from i to j with probability $\frac{w(i,j)}{q+d_i}$,
  - interrupt at any node i with a probability $\frac{q}{q+d_i}$.



- Consider an simple random walk on $\mathcal{G}$ with the transition rule:
  - take a step from i to j with probability $\frac{w(i,j)}{q+d_i}$,
  - interrupt at any node i with a probability $\frac{q}{q+d_i}$.



- Consider an simple random walk on $\mathcal{G}$ with the transition rule:
  - take a step from i to j with probability $\frac{w(i,j)}{q+d_i}$,
  - interrupt at any node i with a probability $\frac{q}{q+d_i}$.



- The expected number of steps is known:

$$\mathrm{tr}\left[(\mathsf{L}+q\mathsf{I})^{-1}(\mathsf{D}+q\mathsf{I})\right] \leq \frac{2|\mathcal{E}|}{q} + |\mathcal{V}|.$$

Outline

Random Spanning Forests (RSF)

RSF-based Algorithms

Conclusion

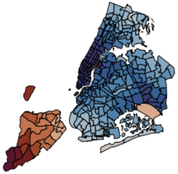
Main Contributions

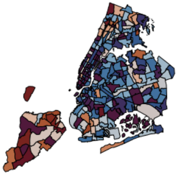
Challenges

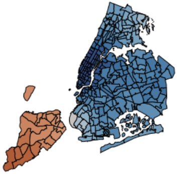
- Graph Signal Smoothing
- Trace Estimation
- Estimating Effective Resistances

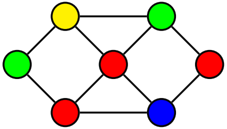
Challenges

- **Graph Signal Smoothing**

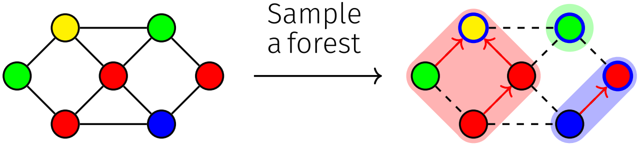
  Original Signal: 

  $\mathbf{y}$: 

  $\hat{\mathbf{x}} = \mathbf{K}\mathbf{y}$: 

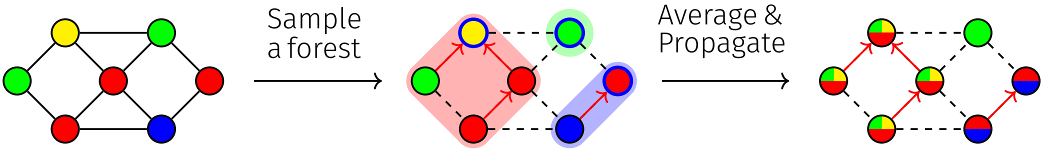


Smoothing Via Forests



- Random partitions are sampled via random spanning forests.

Smoothing Via Forests



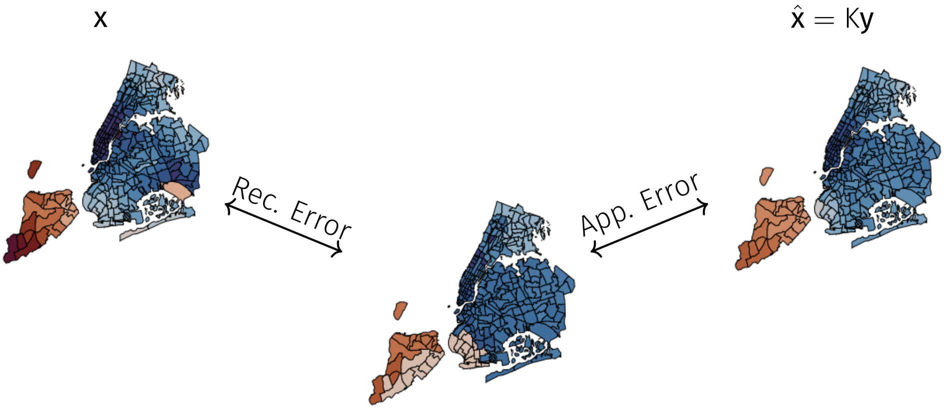
- Random partitions are sampled via random spanning forests.
- This yields an unbiased estimator $\bar{x}$.

Comparison with State of the art

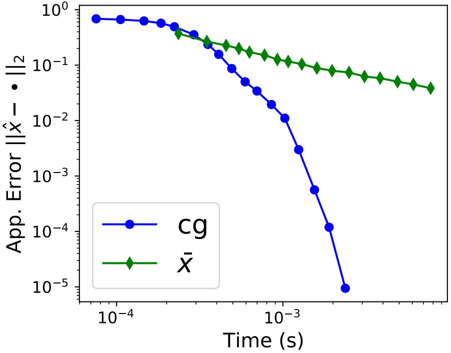
- We compare the algorithms in approximation error (error respect to $\hat{x}$) and reconstruction error (error respect to x)

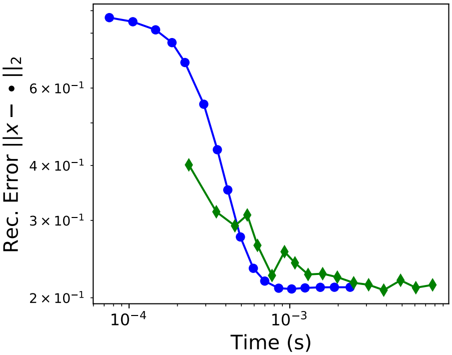
Comparison with State of the art

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



Comparison with State of the art





Gradient Descent Update as Control Variate

- The approximation error can be improved by variance reduction.

Gradient Descent Update as Control Variate

- The approximation error can be improved by variance reduction.
- The solution $\hat{\mathbf{x}}$ also minimizes:

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

Gradient Descent Update as Control Variate

- The approximation error can be improved by variance reduction.
- The solution $\hat{\mathbf{x}}$ also minimizes:

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

- 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 $\nabla F(\mathbf{x}_k) = K^{-1}\mathbf{x}_k - \mathbf{y}$.

Gradient Descent Update as Control Variate

- The approximation error can be improved by variance reduction.
- The solution $\hat{\mathbf{x}}$ also minimizes:

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

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

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

Properties of this estimator

- $\bar{z}$ 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

- $\bar{\mathsf{z}}$ 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:

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

Properties of this estimator

- $\bar{z}$ 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:

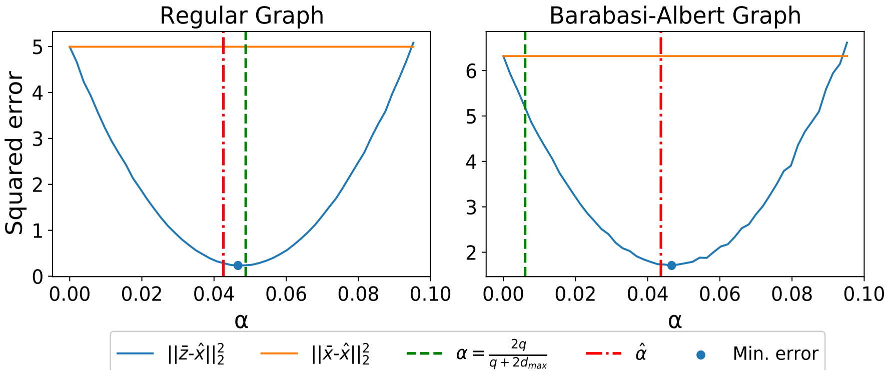
$$\alpha^\star = \frac{\mathrm{tr}(\mathrm{Cov}(K^{-1}\bar{x}, \bar{x}))}{\mathrm{tr}(\mathrm{Var}(K^{-1}\bar{x}))}.$$

- 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:

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

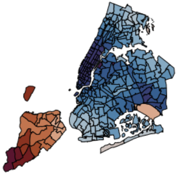
Range of α

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

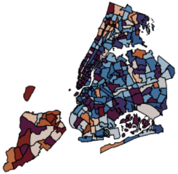


An Illustration

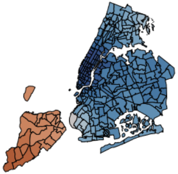
x:



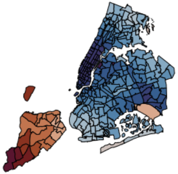
y:



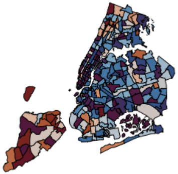
$\hat{x}$:



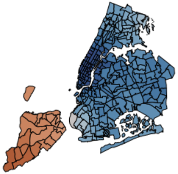
x:



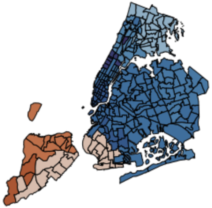
y:



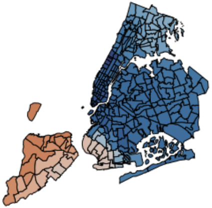
$\hat{x}$:



$\bar{x}$, N=1:



$\bar{z}$, N=1:



An Illustration

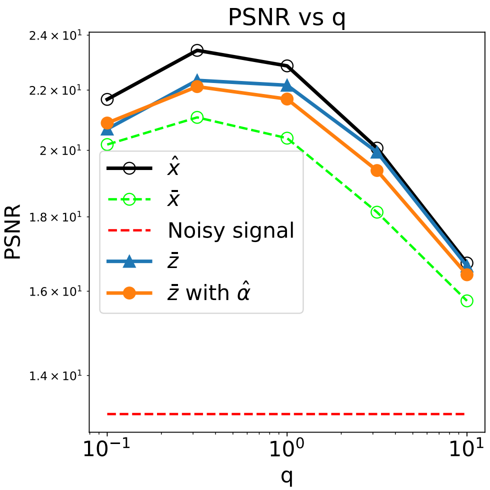


Figure 7: PSNR vs q, N=2

Challenges

- Graph Signal Smoothing
- **Trace Estimation**
- Estimating Effective Resistances

Inverse Trace Estimation: Hutchinson's Estimator

- A famous algorithm for estimating $\mathrm{tr}(K)$ is Hutchinson's estimator:

$$h := \frac{1}{N}\sum_{i=1}^{N} \mathbf{a}^{(i)\top} K \mathbf{a}^{(i)}$$

where $\mathbf{a}^{(i)} \in \{-1, 1\}^n$ is a random vector with $\mathbb{P}(\mathbf{a}_j^{(i)} = \pm 1) = 1/2$.

Inverse Trace Estimation: Hutchinson's Estimator

- A famous algorithm for estimating $\mathrm{tr}(K)$ is Hutchinson's estimator:

$$h := \frac{1}{N}\sum_{i=1}^{N} \mathbf{a}^{(i)\top} K \mathbf{a}^{(i)}$$

  where $\mathbf{a}^{(i)} \in \{-1, 1\}^n$ is a random vector with $\mathbb{P}(\mathbf{a}_j^{(i)} = \pm 1) = 1/2$.
- It is an unbiased estimator of $\mathrm{tr}(K)$.

Inverse Trace Estimation: Hutchinson's Estimator

- A famous algorithm for estimating $\mathrm{tr}(K)$ is Hutchinson's estimator:

$$h := \frac{1}{N} \sum_{i=1}^{N} \mathbf{a}^{(i)\top} K \mathbf{a}^{(i)}$$

  where $\mathbf{a}^{(i)} \in \{-1, 1\}^n$ is a random vector with $\mathbb{P}(\mathbf{a}_j^{(i)} = \pm 1) = 1/2$.
- It is an unbiased estimator of $\mathrm{tr}(K)$.
- The cumbersome computation here is $K\mathbf{a}^{(i)}$ for N vectors.

Inverse Trace Estimation: Hutchinson's Estimator

- A famous algorithm for estimating $\mathrm{tr}(K)$ is Hutchinson's estimator:

$$h := \frac{1}{N} \sum_{i=1}^{N} \mathbf{a}^{(i)\top} K \mathbf{a}^{(i)}$$

  where $\mathbf{a}^{(i)} \in \{-1, 1\}^n$ is a random vector with $\mathbb{P}(\mathbf{a}_j^{(i)} = \pm 1) = 1/2$.
- It is an unbiased estimator of $\mathrm{tr}(K)$.
- The cumbersome computation here is $K\mathbf{a}^{(i)}$ for N vectors.
- It 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 (Barthelme et al. 2019):

$$\mathsf{s} \coloneqq |\rho(\Phi_{\mathsf{q}})| \text{ with } \mathbb{E}[\mathsf{s}] = \mathrm{tr}(\mathsf{K})$$

Forest based Trace Estimator

- Another unbiased estimator is by RSFs (Barthelme et al. 2019):

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

- This estimator gives an comparable performance with the existing algorithms.

Forest based Trace Estimator

- Another unbiased estimator is by RSFs (Barthelme et al. 2019):

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

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

- One can rewrite $\bar{\mathbf{x}} = \bar{\mathrm{S}}\mathbf{y}$.

Variance Reduction via Control Variates

- One can rewrite $\bar{\mathbf{x}} = \bar{\mathsf{S}}\mathbf{y}$.
- The control variate estimator for K:

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

Variance Reduction via Control Variates

- One can rewrite $\bar{\mathsf{x}} = \bar{\mathsf{S}}\mathbf{y}$.
- The control variate estimator for K:

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

- We define the new trace estimator as

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

- One can rewrite $\bar{\mathsf{x}} = \bar{\mathsf{S}}\mathbf{y}$.
- The control variate estimator for K:

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

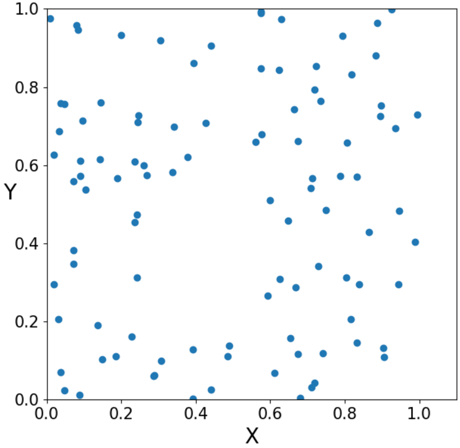
- We define the new trace estimator as

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

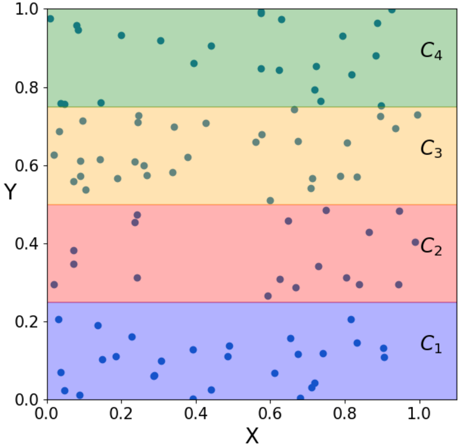
- A safe value of α is $\frac{2q}{q+2d_{max}}$. We also observe that $\frac{2q}{q+2d_{avg}}$ is usually a good estimate of $\alpha^{\star}$.

Variance Reduction via Stratification

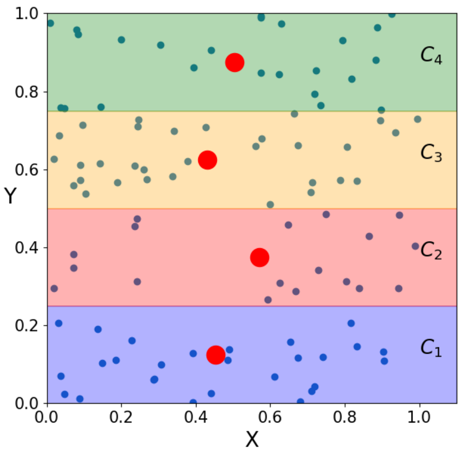
Variance Reduction via Stratification



Variance Reduction via Stratification



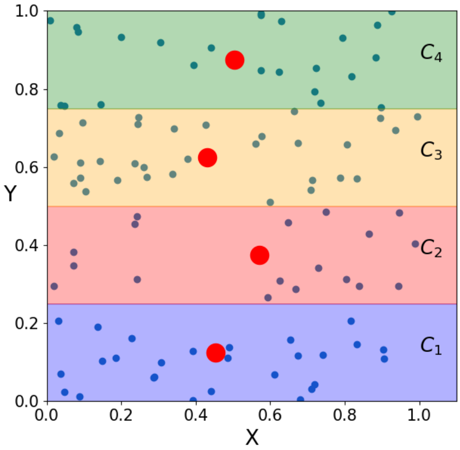
Variance Reduction via Stratification



- The stratified estimator is:

$$X_{st} := \underbrace{\sum_{k=1}^{K} \underbrace{\frac{1}{N_k}\left(\sum_{\substack{j=1\\ Y\in C_k}}^{N_k} X^{(j)}\right)}_{\text{Conditional Expectation}} \mathbb{P}(Y \in C_i)}_{\text{Marginalization over Y}} .$$

Variance Reduction via Stratification

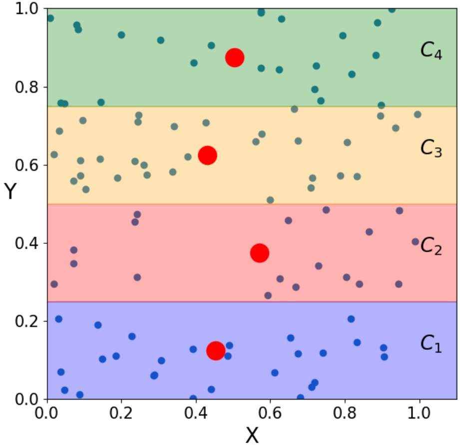


- The stratified estimator is:

$$X_{st} := \sum_{k=1}^{K} \underbrace{\underbrace{\frac{1}{N_k}\left(\sum_{\substack{j=1\\ Y\in C_k}}^{N_k} X^{(j)}\right)}_{\text{Conditional Expectation}} \mathbb{P}(Y \in C_i)}_{\text{Marginalization over } Y}.$$

- For certain allocations of N_k's, one has reduced variance

Variance Reduction via Stratification



- The stratified estimator is:

$$X_{st} := \sum_{k=1}^{K} \underbrace{\underbrace{\frac{1}{N_k}\left(\sum_{\substack{j=1\\ Y\in C_k}}^{N_k} X^{(j)}\right)}_{\text{Conditional Expectation}} \quad \mathbb{P}(Y \in C_i)}_{\text{Marginalization over } Y}.$$

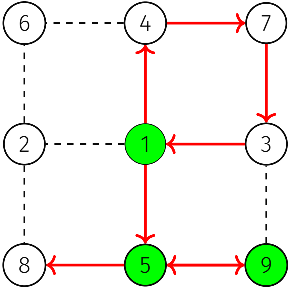
- For certain allocations of N_k's, one has reduced variance
- We need to have a random variable Y such that:
    - $X|Y$ is easy to sample,
    - $\mathbb{P}(Y \in C_i)$ is accessible.

Variance Reduction via Stratification

- $Y = |\rho_1(\Phi_q)|$ as the number of the roots that are sampled at the first visit.

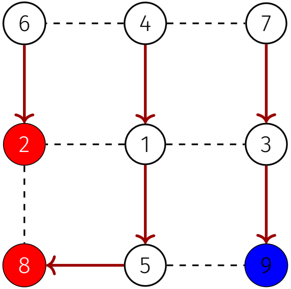
Variance Reduction via Stratification

- $Y = |\rho_1(\Phi_q)|$ as the number of the roots that are sampled at the first visit.



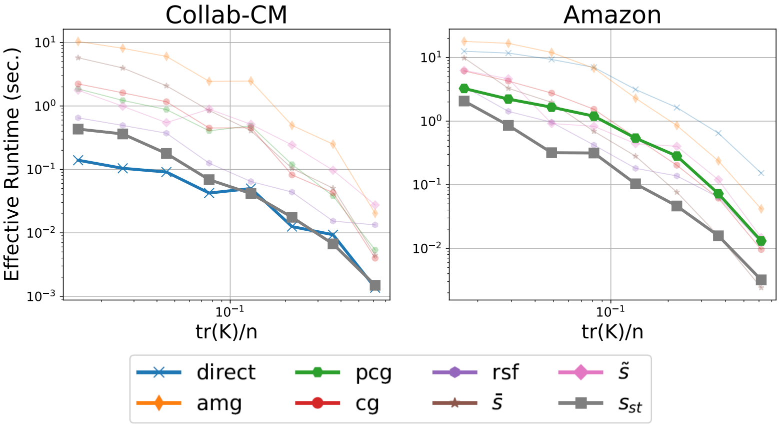
Variance Reduction via Stratification

- $Y = |\rho_1(\Phi_q)|$ as the number of the roots that are sampled at the first visit.



Comparison with Hutchinson's Estimator

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



Challenges

- Graph Signal Smoothing
- Trace Estimation
- Estimating Effective Resistances

Definition (Electrical Representation)
In the electrical representation of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathrm{w})$, each edge is a resistor with a resistance $\frac{1}{\mathrm{w(i,j)}}$.

Definition (Electrical Representation)
In the electrical representation of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$, each edge is a resistor with a resistance $\frac{1}{w(i,j)}$.

- The effective resistance between node i and j:

$$R_{i,j} := L^{\dagger}_{i,i} + L^{\dagger}_{j,j} - L^{\dagger}_{i,j} - L^{\dagger}_{j,i}$$

Definition (Electrical Representation)
In the electrical representation of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathrm{w})$, each edge is a resistor with a resistance $\frac{1}{\mathrm{w}(i,j)}$.

- The effective resistance between node i and j:

$$R_{i,j} := L^{\dagger}_{i,i} + L^{\dagger}_{j,j} - L^{\dagger}_{i,j} - L^{\dagger}_{j,i}$$

- The effective conductance:

$$I_{i,j} := \frac{1}{R_{i,j}}$$

- $R_{i,j}$ is a distance metric between i and j.

Effective Resistances: What, **Why** and How ?

- $R_{i,j}$ is a distance metric between i and j.
- They are of central importance in many graph related applications:
    - Clustering (Alev et al. 2017),
    - Sparsification (Spielman and Srivastava 2011),
    - Learning (Ghosh et al. 2008),
    - Network robustness (Wang et al. 2014).

- $R_{i,j}$ is a distance metric between i and j.
- They are of central importance in many graph related applications:
  - Clustering (Alev et al. 2017),
  - Sparsification (Spielman and Srivastava 2011),
  - Learning (Ghosh et al. 2008),
  - Network robustness (Wang et al. 2014).
- In large scale, they are expensive to compute.

- The well-known algorithms are Monte Carlo estimators.

- The well-known algorithms are Monte Carlo estimators.
- They can be divided into two groups:

- The well-known algorithms are Monte Carlo estimators.
- They can be divided into two groups:
    - **Global methods** estimate $R_{i,j}$'s for all pairs (i, j) (or all edges $(i, j) \in \mathcal{E}$), e.g. estimating by Random Projections (RP) (Spielman and Srivastava 2011) or Spanning Trees (ST) (Hayashi et al. 2016),

Effective Resistances: What, Why and **How**?

- The well-known algorithms are Monte Carlo estimators.
- They can be divided into two groups:
  - **Global methods** estimate $R_{i,j}$'s for all pairs (i, j) (or all edges $(i, j) \in \mathcal{E}$), e.g. estimating by Random Projections (RP) (Spielman and Srivastava 2011) or Spanning Trees (ST) (Hayashi et al. 2016),
  - **Local methods** estimate small number of pairs without discovering the whole graph (Peng et al. 2021).

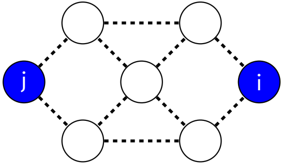
Effective Resistances: What, Why and **How**?

- The well-known algorithms are Monte Carlo estimators.
- They can be divided into two groups:
  - **Global methods** estimate $R_{i,j}$'s for all pairs (i, j) (or all edges $(i, j) \in \mathcal{E}$), e.g. estimating by Random Projections (RP) (Spielman and Srivastava 2011) or Spanning Trees (ST) (Hayashi et al. 2016),
  - **Local methods** estimate small number of pairs without discovering the whole graph (Peng et al. 2021).
- The RSF-based global and local estimators are proposed.

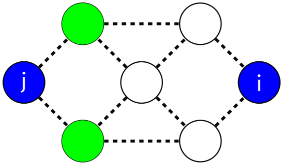
Effective Resistances: What, Why and **How**?

- The well-known algorithms are Monte Carlo estimators.
- They can be divided into two groups:
  - **Global methods** estimate $R_{i,j}$'s for all pairs (i, j) (or all edges $(i, j) \in \mathcal{E}$), e.g. estimating by Random Projections (RP) (Spielman and Srivastava 2011) or Spanning Trees (ST) (Hayashi et al. 2016),
  - **Local methods** estimate small number of pairs without discovering the whole graph (Peng et al. 2021).
- The RSF-based global and local estimators are proposed.

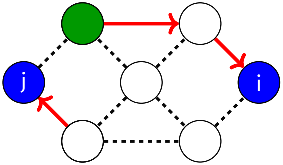
ESTIMATING $R_{i,j}$ VIA LOCAL FORESTS (LF)



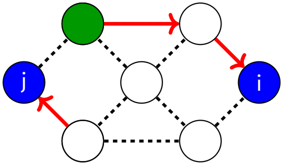
Estimating $R_{i,j}$ via Local Forests (LF)



Estimating $R_{i,j}$ via Local Forests (LF)

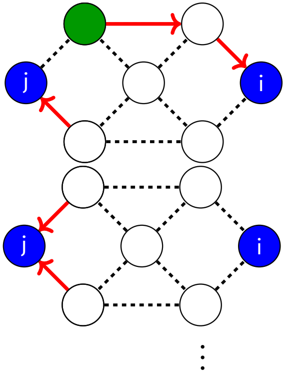


Estimating $R_{i,j}$ via Local Forests (LF)



$\hat{l}_{i,j}^{(1)} = 1$

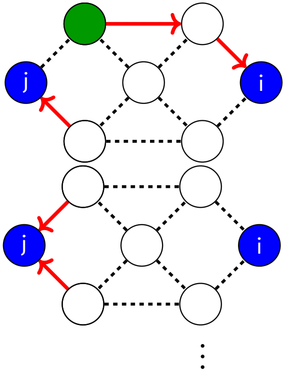
Estimating $R_{i,j}$ via Local Forests (LF)



$\hat{I}^{(1)}_{i,j} = 1$

$\hat{I}^{(2)}_{i,j} = 0$

Estimating $R_{i,j}$ via Local Forests (LF)

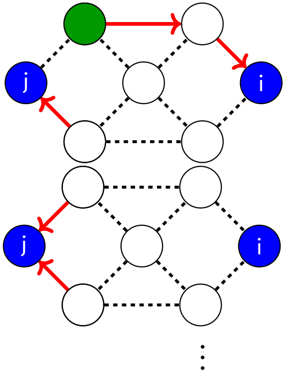


$\hat{\mathbb{I}}_{i,j}^{(1)} = 1$

$\hat{\mathbb{I}}_{i,j}^{(2)} = 0$

$$R_{i,j}^{LF} := \frac{N}{\sum_{k=1}^{N} \hat{\mathbb{I}}_{i,j}^{(k)}}$$

Estimating $R_{i,j}$ via Local Forests (LF)



$\hat{I}^{(1)}_{i,j} = 1$

$\hat{I}^{(2)}_{i,j} = 0$

$$R^{LF}_{i,j} := \frac{N}{\sum_{k=1}^{N} \hat{I}^{(k)}_{i,j}}$$

- R^{LF} is biased but the bias diminishes faster than the variance.

Experiments

- We report the run-time of the local algorithms for approximately the same relative error.

Dataset \ Algorithm	TP	MC2	LF(ours)
Cora	116	11	**2**
Citeseer	362	6	**1**
Pubmed	333	91	**12**
Collab-CM	82	156	**20**

Table 1: Runtime (ms) of the local algorithms over benchmark datasets

Outline

Random Spanning Forests (RSF)

RSF-based Algorithms

Conclusion

- This thesis gives a new perspective for Laplacian-based numerical algebra.

Conclusion

- This thesis gives a new perspective for Laplacian-based numerical algebra.
- This perspective leverages fascinating links between the graph Laplacians and RSFs, yielding efficient solutions to the applications:

Conclusion

- This thesis gives a new perspective for Laplacian-based numerical algebra.
- This perspective leverages fascinating links between the graph Laplacians and RSFs, yielding efficient solutions to the applications:

Not only in,

- Graph signal smoothing,
- Trace estimation,
- Estimating ERs.

Conclusion

- This thesis gives a new perspective for Laplacian-based numerical algebra.
- This perspective leverages fascinating links between the graph Laplacians and RSFs, yielding efficient solutions to the applications:

Not only in,

- Graph signal smoothing,
- Trace estimation,
- Estimating ERs.

But also,

- Graph signal filtering,
- Semi-supervised learning,
- Graph-based optimization.

- This thesis gives a new perspective for Laplacian-based numerical algebra.
- This perspective leverages fascinating links between the graph Laplacians and RSFs, yielding efficient solutions to the applications:

Not only in,

- Graph signal smoothing,
- Trace estimation,
- Estimating ERs.

But also,

- Graph signal filtering,
- Semi-supervised learning,
- Graph-based optimization.

Limitations and Open Questions

Limitations

Possible Directions

Limitations and Open Questions

Limitations

- Restricted to SDDs,

Possible Directions

Limitations and Open Questions

Limitations

- Restricted to SDDs,

Possible Directions

- Extending other matrices via Importance Sampling,

Limitations and Open Questions

Limitations

- Restricted to SDDs,
- Not competitive in high precision regime,

Possible Directions

- Extending other matrices via Importance Sampling,

Limitations and Open Questions

Limitations

- Restricted to SDDs,
- Not competitive in high precision regime,

Possible Directions

- Extending other matrices via Importance Sampling,
- RSFs as Preconditioning,

Limitations

- Restricted to SDDs,
- Not competitive in high precision regime,
- Sampling at small q,

Possible Directions

- Extending other matrices via Importance Sampling,
- RSFs as Preconditioning,

Limitations and Open Questions

Limitations

- Restricted to SDDs,
- Not competitive in high precision regime,
- Sampling at small q,

Possible Directions

- Extending other matrices via Importance Sampling,
- RSFs as Preconditioning,
- Faster sampling algorithms, Early-stop strategies...

Limitations

- Restricted to SDDs,
- Not competitive in high precision regime,
- Sampling at small q,

Possible Directions

- Extending other matrices via Importance Sampling,
- RSFs as Preconditioning,
- Faster sampling algorithms, Early-stop strategies...

Open Questions

- $\rho_1(\Phi_q)$ is a DPP as well:

$$\mathbb{P}(S \subseteq \rho_1(\Phi_q)) = \det K_1 \text{ with } K_1 = q(qI + D)^{-1}.$$

Limitations and Open Questions

Limitations

- Restricted to SDDs,
- Not competitive in high precision regime,
- Sampling at small q,

Possible Directions

- Extending other matrices via Importance Sampling,
- RSFs as Preconditioning,
- Faster sampling algorithms, Early-stop strategies...

Open Questions

- $\rho_1(\Phi_q)$ is a DPP as well:

$$\mathbb{P}(S \subseteq \rho_1(\Phi_q)) = \det K_1 \text{ with } K_1 = q(qI + D)^{-1}.$$

- What happens between $\rho_1(\Phi_q)$ and $\rho(\Phi_q)$?

Publications

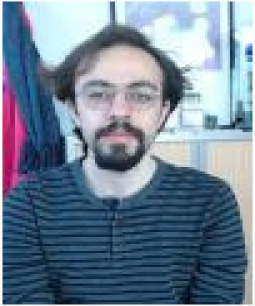
Journal

- Yusuf Yiğit Pilavcı, Pierre-Olivier Amblard, Simon Barthelme, and Nicolas Tremblay (2021). “Graph tikhonov regularization and interpolation via random spanning forests”. In: IEEE transactions on Signal and Information Processing over Networks 7, pp. 359–374

Conference

- Yusuf Yigit Pilavci, Pierre-Olivier Amblard, Simon Barthelme, and Nicolas Tremblay (Sept. 2022). “Variance Reduction for Inverse Trace Estimation via Random Spanning Forests”. In: GRETSI 2022 - XXVIIIème Colloque Francophone de Traitement du Signal et des Images. Nancy, France
- Yusuf Yigit Pilavcı, Pierre-Olivier Amblard, Simon Barthelmé, and Nicolas Tremblay (2022). “Variance Reduction in Stochastic Methods for Large-Scale Regularized Least-Squares Problems”. In: 2022 30th European Signal Processing Conference (EUSIPCO). IEEE, pp. 1771–1775
- Yusuf Y Pilavci, Pierre-Olivier Amblard, Simon Barthelme, and Nicolas Tremblay (2020). “Smoothing graph signals via random spanning forests”. In: ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, pp. 5630–5634

Thanks!



Just a PhD..

Random Spanning Forests

- For fixed subsets $V \subseteq \mathcal{V}$ and $S \subseteq \mathcal{E}$ with $|V| = |S|$, one has:

$$\det B_{S|V} = \begin{cases} \left(\prod_{(i,j)\in S} w(i,j)\right)^{1/2}, & \text{if S forms a spanning forest rooted in V} \\ 0, & \text{otherwise.} \end{cases}$$

- We can count the spanning forests rooted in $R \subseteq \mathcal{V}$:

$$\forall R \subseteq \mathcal{V}, \quad \det L_{-R} = \sum_{\phi \in \mathcal{F}_R} \prod_{(i,j)\in \mathcal{E}_\phi} w(i,j).$$

- The root probability can be seen as a ratio of counts:

$$\mathbb{P}(r_{\Phi_q}(i) = j) = K_{i,j} = q \frac{(-1)^{i+j} \det(L + qI)_{-i|-j}}{\det(L + qI)} = \frac{|\mathcal{F}^{i \to j}|}{|\mathcal{F}|}$$

Loop-Erased Random Walks

Theorem (Law of LERWs (Marchal 2000))
A loop-erased random walk LE(W) on $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathrm{w})$ that is stopped at the boundary $\Delta \subset \mathcal{V}$ has the following probability distribution:

$$\mathbb{P}(\mathrm{LE(W)} = \gamma) = \frac{\det \mathrm{L}_{-\Delta \cup \mathrm{s}(\gamma)}}{\det \mathrm{L}_{-\Delta}} \prod_{(\mathrm{i},\mathrm{j}) \in \gamma} \mathrm{w}(\mathrm{i}, \mathrm{j})$$

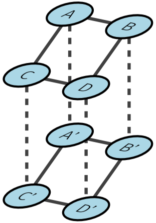
where γ is a fixed path and $\mathrm{s}(\gamma)$ denotes the nodes visited in γ.

Graph Filtering via Duplicated Graph

- The product K$\mathbf{y}$ corresponds to a graph filtering with the transfer function:

$$g_q(\lambda) = \frac{q}{q + \lambda}$$

- We duplicate the graph and the input $\mathbf{y}_d = \begin{bmatrix} \alpha \mathbf{y} \\ \beta \mathbf{y} \end{bmatrix}$



- The transfer function is paramerized by $\theta = (q_1, q_2, \alpha, \beta)$:

$$f_\theta(\lambda) = \frac{\alpha q_1(\lambda + h(0) + q_2) + \beta q_2(h(\lambda))}{(\lambda + h(0) + q_1)(\lambda + h(0) + q_2) - h(\lambda)^2},$$

L_1 Graph Regularization

- Another type of regularization is L_1 regularization:

$$x^\star = \operatorname*{argmin}_{x \in \mathbb{R}^n} \frac{q}{2}||x - y||_2^2 + ||Bx||_1$$

- Alternating direction of multipliers (ADMM) approximates $x^\star$ by:

$$x_{k+1} = \operatorname*{argmin}_{x \in \mathbb{R}^n} \left(\frac{q}{2}||x - y||_2^2 + \frac{\rho}{2}||Bx - z_k + u_k||_2^2 \right)$$
$$z_{k+1} = \operatorname*{argmin}_{z \in \mathbb{R}^m} \left(||z||_1 + \frac{\rho}{2}||Bx_{k+1} - z + u_k||_2^2 \right)$$
$$u_{k+1} = u_k + (Bx^{k+1} - z^{k+1}).$$

Extension to SDDs

- Let $G = U^\top \Lambda U = A^{(p)} + A^{(n)} + D^{(1)} + D^{(2)}$ be an symmetric diagonally dominant matrix where $D^{(1)}_{i,i} = \sum_{i \neq j} G_{i,j}$ and $D^{(2)}_{i,i} = G_{i,i} - D^{(1)}_{i,i}$.
- Construct the graph Laplacians:

$$L_1 = D^{(1)} + A^{(n)} - A^{(p)}/2 = U_1^\top \Lambda_1 U_1$$

$$L_2 = \begin{bmatrix} D^{(1)} + A^{(n)} + D^{(2)}/2 & -D^{(2)}/2 - A^{(p)} \\ -D^{(2)}/2 - A^{(p)} & D^{(1)} + A^{(n)} + D^{(2)}/2 \end{bmatrix}$$

- The eigenvectors of L_2 are $U_2 = \begin{bmatrix} U & U_1 \\ -U & U_1 \end{bmatrix}$
- The eigenvalues of L_2 are $\lambda_2 = \lambda_1 \cup \lambda$

Cross-Validation for GTR

- The leave-one-out cross-validation for graph Tikhonov regularization boils down to:

$$\mathrm{LOOCV}(q) = \frac{1}{n}\left(\sum_{i=1}^{n} \frac{y_i - \hat{x}_i}{1 - K_{i,i}}\right)^2 .$$

- The generalized CV approximation is:

$$\mathrm{GCV}(q) = \frac{1}{N}\left(\sum_{i=1}^{n} \frac{y_i - \hat{x}_i}{1 - \left(\mathrm{tr}(K)/n\right)}\right)^2 .$$

References I

- Alev, Vedat Levi et al. (2017). “Graph clustering using effective resistance”. In: arXiv preprint arXiv:1711.06530.
- Avena, Luca et al. (2018). “Random forests and networks analysis”. In: Journal of Statistical Physics 173.3, pp. 985–1027.
- Barthelme, Simon et al. (Aug. 2019). “Estimating the inverse trace using random forests on graphs”. In: GRETSI 2019 - XXVIIème Colloque francophone de traitement du signal et des images. Lille, France. URL: https://hal.archives-ouvertes.fr/hal-02319194.
- Ghosh, Arpita et al. (2008). “Minimizing effective resistance of a graph”. In: SIAM review 50.1, pp. 37–66.

References II

- Hayashi, Takanori et al. (2016). “Efficient Algorithms for Spanning Tree Centrality.”. In: IJCAI. Vol. 16, pp. 3733–3739.
- Marchal, Philippe (2000). “Loop-erased random walks, spanning trees and Hamiltonian cycles”. In: Electronic Communications in Probability 5, pp. 39–50.
- Peng, Pan et al. (2021). “Local Algorithms for Estimating Effective Resistance”. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, pp. 1329–1338.
- Pilavci, Yusuf Y et al. (2020). “Smoothing graph signals via random spanning forests”. In: ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, pp. 5630–5634.

References III

Pilavci, Yusuf Yigit et al. (Sept. 2022). "Variance Reduction for Inverse Trace Estimation via Random Spanning Forests". In: GRETSI 2022 - XXVIIIème Colloque Francophone de Traitement du Signal et des Images. Nancy, France.

Pilavcı, Yusuf Yigit et al. (2022). "Variance Reduction in Stochastic Methods for Large-Scale Regularized Least-Squares Problems". In: 2022 30th European Signal Processing Conference (EUSIPCO). IEEE, pp. 1771–1775.

Pilavcı, Yusuf Yiğit et al. (2021). "Graph tikhonov regularization and interpolation via random spanning forests". In: IEEE transactions on Signal and Information Processing over Networks 7, pp. 359–374.

Spielman, Daniel A and Nikhil Srivastava (2011). "Graph sparsification by effective resistances". In: SIAM Journal on Computing 40.6, pp. 1913–1926.

- Spielman, Daniel A and Shang-Hua Teng (2004). "Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems". In: Proceedings of the thirty-sixth annual ACM symposium on Theory of computing, pp. 81–90.
- Wang, Xiangrong et al. (2014). "Improving robustness of complex networks via the effective graph resistance". In: The European Physical Journal B 87.9, pp. 1–12.
- Wilson, David Bruce (1996). "Generating random spanning trees more quickly than the cover time". In: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing, pp. 296–303.