Laplacian Paradigm 2.0

8:40-9:10: Merging Continuous and Discrete (Richard Peng)
9:10-9:50: Beyond Laplacian Solvers (Aaron Sidford)
9:50-10:30: Approximate Gaussian Elimination (Sushant Sachdeva)
10:30-11:00: coffee break
11:00-12:00: Analysis using matrix Martingales (Rasmus Kyng)
12:00-14:00 lunch
14:00-15:00 Graph Structure via Eliminations (Aaron Schild)

Website: bit.ly/laplacian2

Merging the Continuous and Discrete

Richard Peng

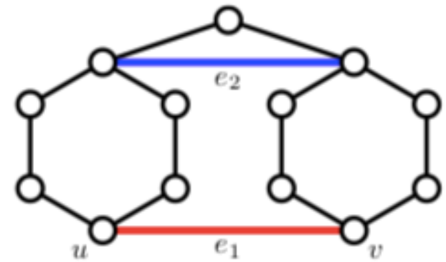
Oct 6, 2018

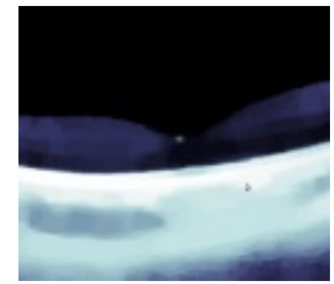
Outline

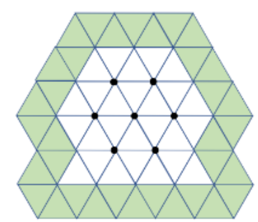
- **Graphs and Laplacians**
- Building Blocks
- Laplacian Paradigm 2.0

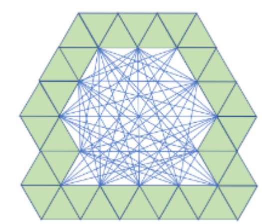
Large Networks

- Data mining:
  centrality, clustering…
- Image/video processing:
  segmentation, denoising …
- Scientific computing:
  stress, fluids, waves…







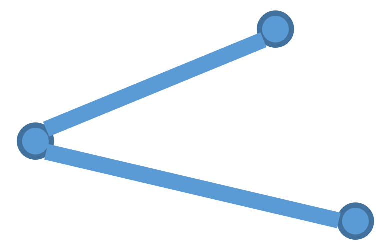


- \ (linear system solve)
- CVX (convex optimization)
- Eigenvector solvers

Graphs and Matrices

High performance computing: non-zeros ⇔ edges, design / analyze matrix algorithms using graph theory

$$\begin{matrix} 2 & -1 & -1 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{matrix}$$



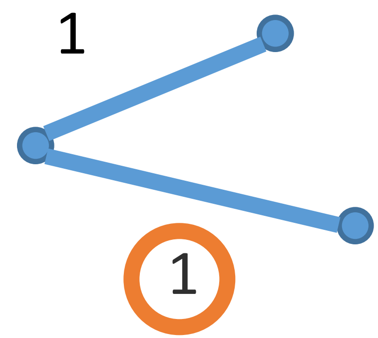
n vertices
m edges

n rows / columns
O(m) non-zeros

Graphs and Matrices

High performance computing: non-zeros ⇔ edges,
design / analyze matrix algorithms using graph theory

$$\begin{matrix} 2 & -1 & -1 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{matrix}$$



n vertices
m edges

n rows / columns
O(m) non-zeros

graph Laplacian matrix **L**

- Diagonal: degrees
- Off-diagonal: -edge weights

Source of Laplacians

2	-1	-1
-1	1	0
-1	0	1

graph Laplacian matrix **L**

- Diagonal: degrees
- Off-diagonal: -edge weights

d-Regular graphs: **L** = d**I** – **A**, **A**: adjacency matrix

Source of Laplacians

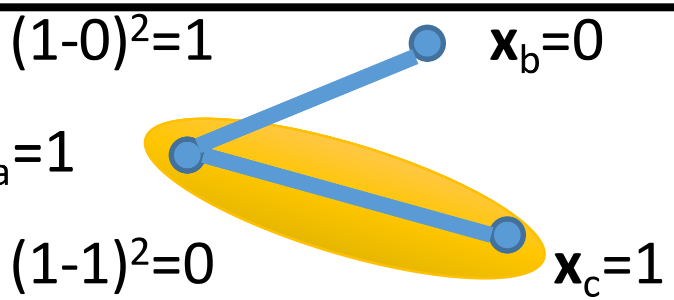
$$\begin{matrix} 2 & -1 & -1 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{matrix}$$

graph Laplacian matrix **L**

- Diagonal: degrees
- Off-diagonal: -edge weights

d-Regular graphs: $\mathbf{L} = d\mathbf{I} - \mathbf{A}$, $\mathbf{A}$: adjacency matrix

Graph cuts: $\mathbf{x}^T\mathbf{L}\mathbf{x} = \sum_{u\sim v} \mathbf{w}_{uv}(\mathbf{x}_u - \mathbf{x}_v)^2$



x indicator vector of cut ➔ weight of cut

Source of Laplacians

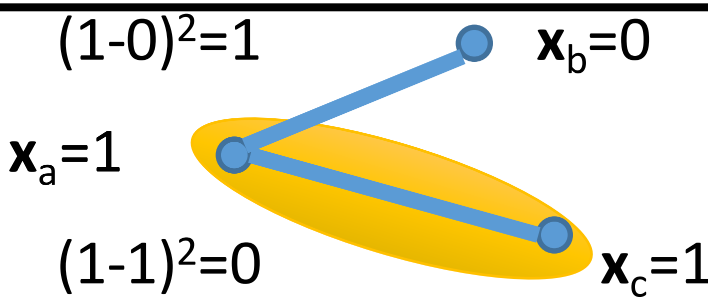
$$\begin{matrix} 2 & -1 & -1 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{matrix}$$

graph Laplacian matrix **L**

- Diagonal: degrees
- Off-diagonal: -edge weights

d-Regular graphs: $\mathbf{L} = d\mathbf{I} - \mathbf{A}$, $\mathbf{A}$: adjacency matrix

Graph cuts: $\mathbf{x}^T\mathbf{L}\mathbf{x} = \sum_{u\sim v} \mathbf{w}_{uv}(\mathbf{x}_u - \mathbf{x}_v)^2$



x indicator vector of cut ➔ weight of cut

$\mathbf{L} = \mathbf{B}^T\mathbf{W}\mathbf{B}$ where $\mathbf{B}$ is edge-vertex incidence matrix

Origin of the Laplacian Paradigm

[Spielman Teng `04]
Input: graph Laplacian **L**

vector **b**

Output: vector x s.t. $\mathbf{Lx} \cong \mathbf{b}$

Runtime: $O(m\log^{O(1)} n \log(1/\varepsilon))$

Origin of the Laplacian Paradigm

[Spielman Teng `04]
Input: graph Laplacian **L**
vector **b**
Output: vector x s.t. $\mathbf{Lx} \cong \mathbf{b}$
Runtime: $O(m\log^{\boxed{O(1)}} n \log(1/\varepsilon))$

[Cohen-Kyng-Miller-Pachocki-P-Rao-Xu `14]: ≤ 1/2

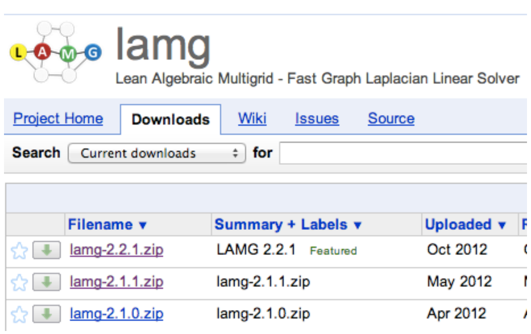
Origin of the Laplacian Paradigm

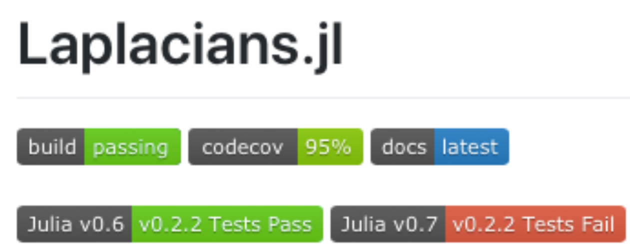
[Spielman Teng `04]
Input: graph Laplacian **L**
vector **b**

Output: vector x s.t. **Lx** ≅ **b**

Runtime: $O(m\log^{\boxed{O(1)}} n \log(1/\varepsilon))$

[Cohen-Kyng-Miller-Pachocki-P-Rao-Xu `14]: ≤ 1/2



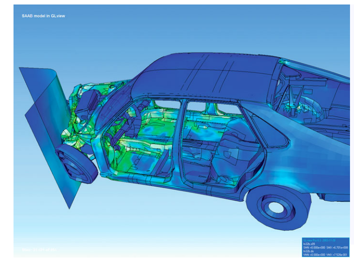


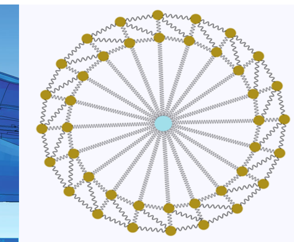
Wall clock: $m \leq 10^7$ in ≤ 20s

The Laplacian Paradigm

Directly related:

Elliptic systems



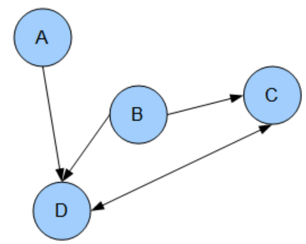


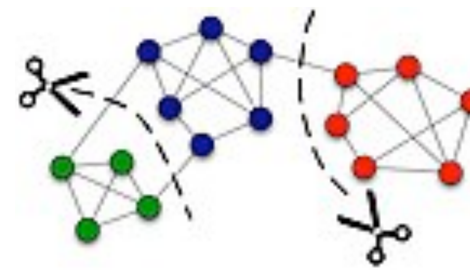
Lx=b

Few iterations:

Eigenvectors,

Heat kernels

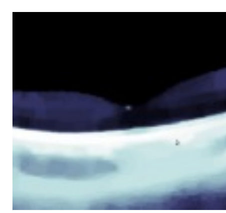


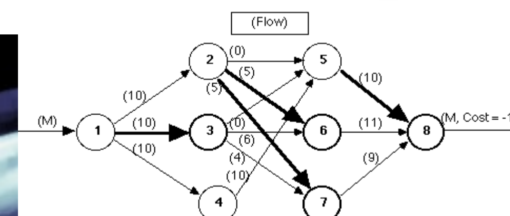


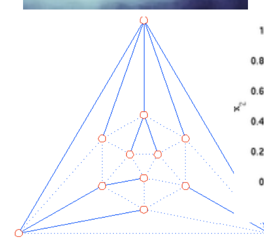
Many iterations / modify algorithm

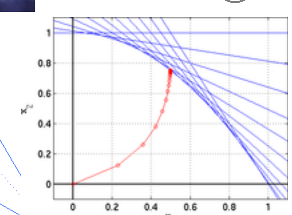
Graph problems

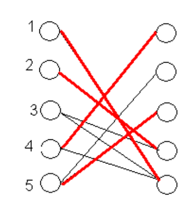
Image processing











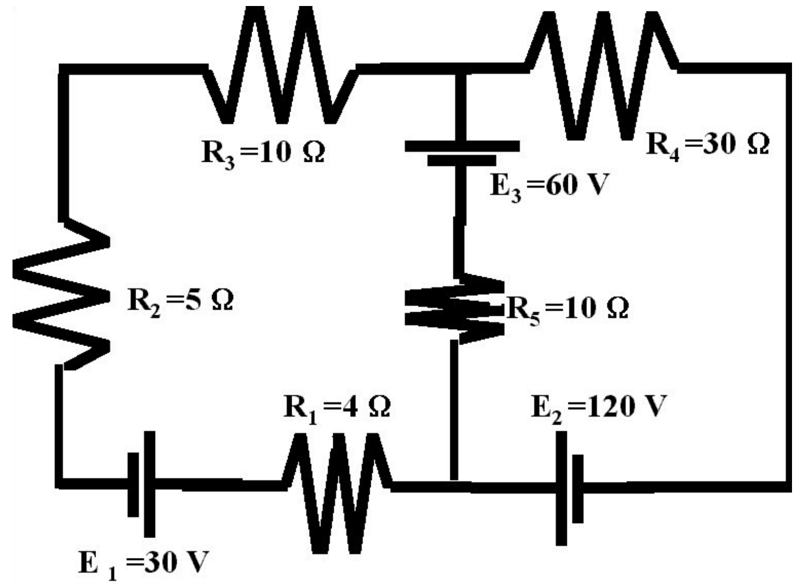
Outline

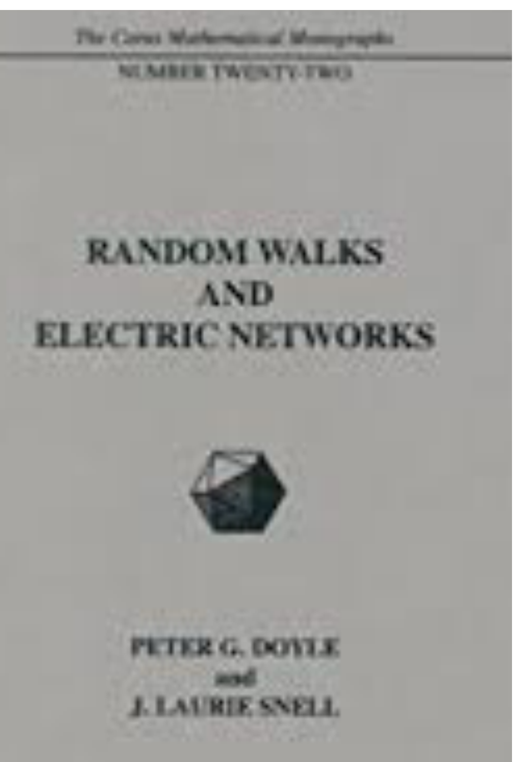
- Graphs and Laplacians
- **Building Blocks**
- Laplacian Paradigm 2.0

Lx = b as a graph problem

x: voltage vectors

Dual: electrical flow **f**



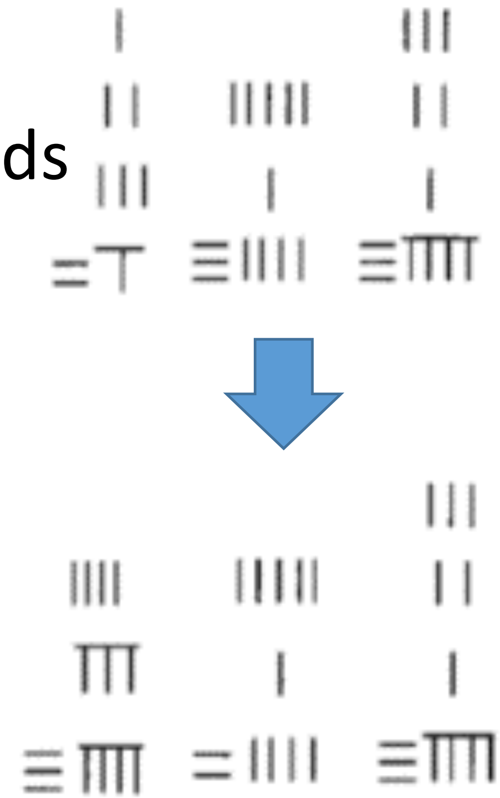


Unified formulation: $\min_{\mathbf{f}\text{ with resdiual }\mathbf{b}} \|f\|_p$:

- $p = 2$: solving **Lx** = **b**
- $p = 1$: shortest path / transshipment
- $p = \infty$: max-flow/min-cut

Direct Methods (combinatorial)

Repeatedly remove vertices by creating equivalent graphs on their neighborhoods



$\mathbf{M}^{(2)} \leftarrow \text{Eliminate}(\mathbf{M}^{(1)}, i_1)$

$\mathbf{M}^{(3)} \leftarrow \text{Eliminate}(\mathbf{M}^{(2)}, i_2)$

…

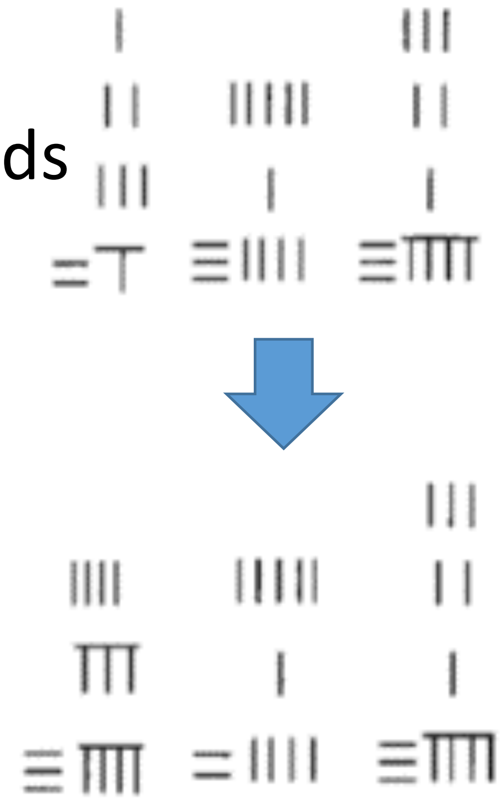
Direct Methods (combinatorial)

Repeatedly remove vertices by creating equivalent graphs on their neighborhoods

$$\mathbf{M}^{(2)} \leftarrow \text{Eliminate}(\mathbf{M}^{(1)}, i_1)$$

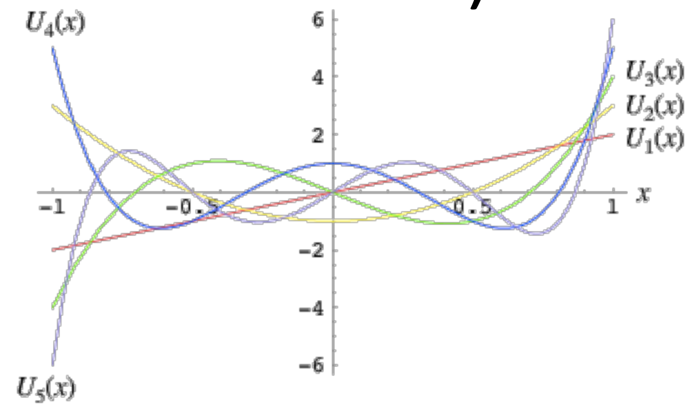
$$\mathbf{M}^{(3)} \leftarrow \text{Eliminate}(\mathbf{M}^{(2)}, i_2)$$

…



- Parallel graph algorithms
- Matrix multiplication / dense solves
- Sparsified squaring

Iterative Methods (numerical)

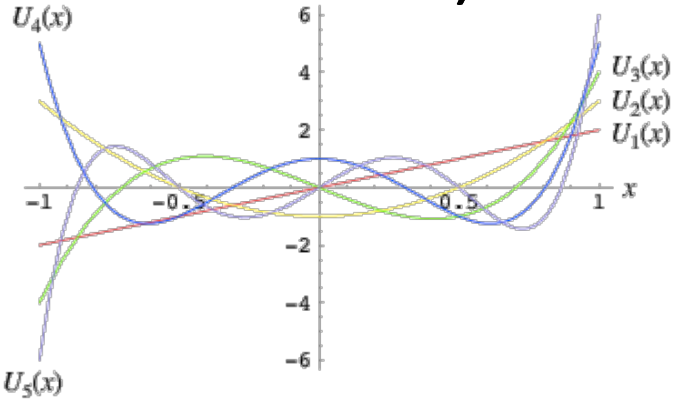


Solve $\mathbf{Ax} = \mathbf{b}$ by

$\mathbf{x} \leftarrow \mathbf{x} - (\mathbf{Ax} - \mathbf{b})$

Fixed point: $\mathbf{Ax} - \mathbf{b} = 0$

Iterative Methods (numerical)



Preconditioning:

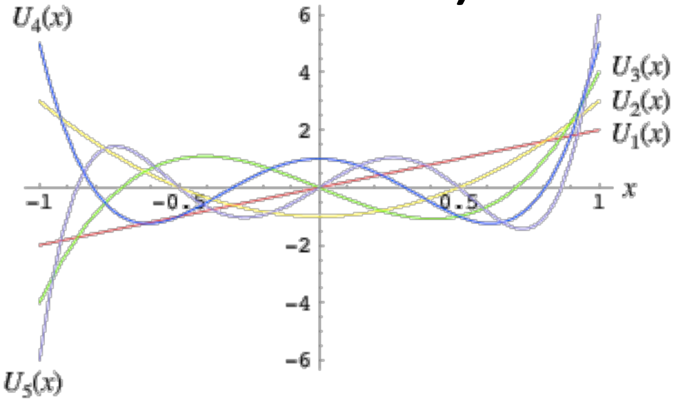
Solve $\mathbf{B}^{-1}\mathbf{A}\mathbf{x} = \mathbf{B}^{-1}\mathbf{b}$ by:

$$\mathbf{x} \leftarrow \mathbf{x} - \mathbf{B}^{-1}(\mathbf{A}\mathbf{x} - \mathbf{b})$$

Fixed point: $\mathbf{A}\mathbf{x} - \mathbf{b} = 0$

- Simple **B**: $\mathbf{B} = \mathbf{I}$, many iterations
- $\mathbf{B} = \mathbf{A}$: 1 iteration, but same problem

Iterative Methods (numerical)



Preconditioning:

Solve $\mathbf{B}^{-1}\mathbf{A}\mathbf{x} = \mathbf{B}^{-1}\mathbf{b}$ by:

$\mathbf{x} \leftarrow \mathbf{x} - \mathbf{B}^{-1}(\mathbf{A}\mathbf{x} - \mathbf{b})$

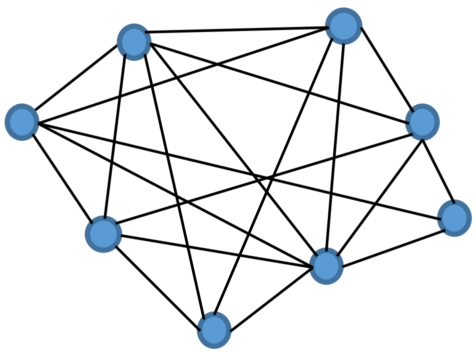
Fixed point: $\mathbf{A}\mathbf{x} - \mathbf{b} = 0$

- Simple **B**: **B** = **I**, many iterations
- **B** = **A**: 1 iteration, but same problem

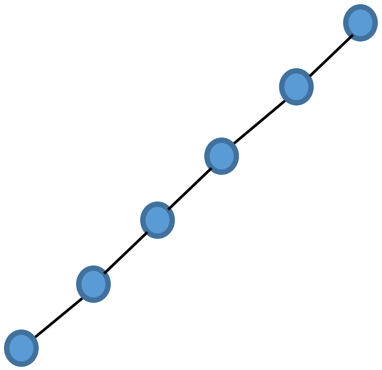
- Krylov space methods / PCG
- Convex optimization algorithms

Hard instances

Direct methods create too much fill on highly connected graphs

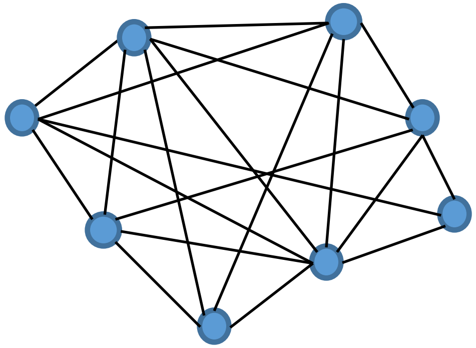


Iterative methods take too many iterations paths

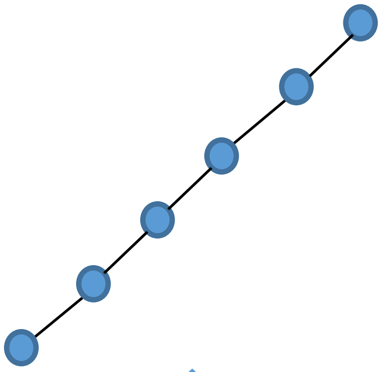


Hard instances

Direct methods create too much fill on highly connected graphs



Iterative methods take too many iterations paths



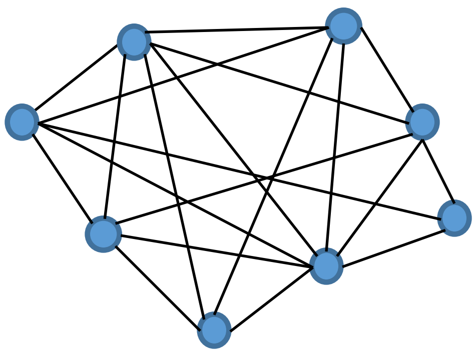
Still `easy' by themselves

Easy for iterative methods

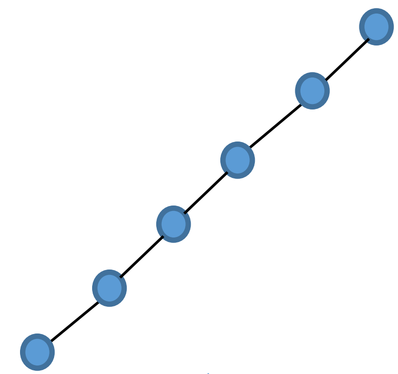
Easy for direct methods

Hard instances

Direct methods create too much fill on highly connected graphs



Iterative methods take too many iterations paths



Still `easy' by themselves

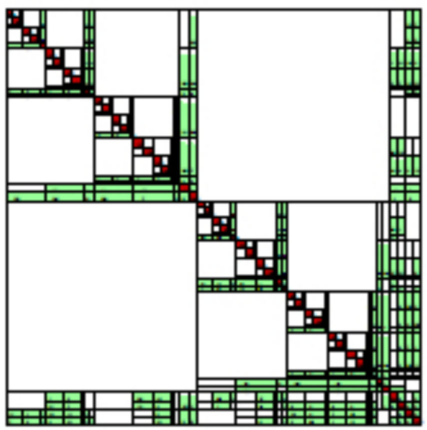
Easy for iterative methods

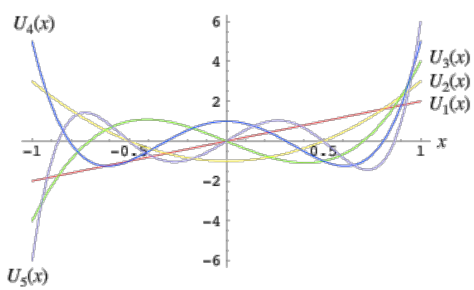
Easy for direct methods

Must handle both simultaneously, but avoid paying n iterations $\times$ m per iteration

Hybrid algorithms (aka. v1.0)

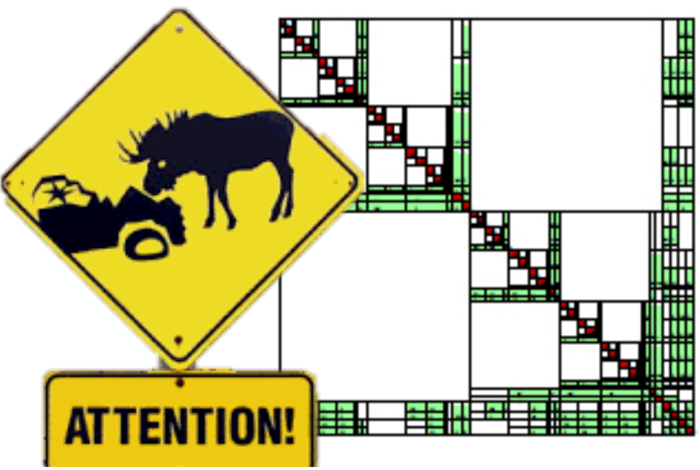
- Scientific computing: iChol, multigrid
- [Vaidya `89] precondition with graphs

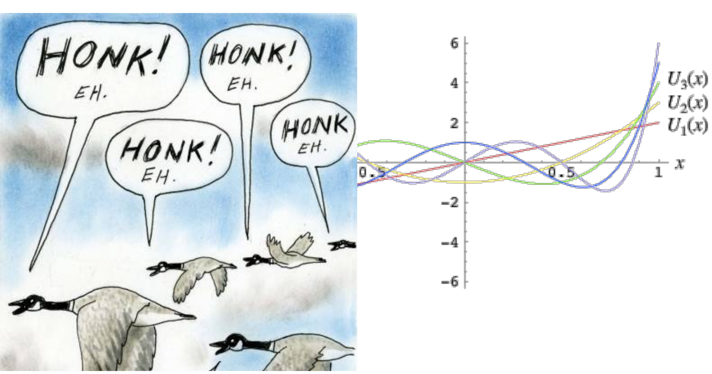




Hybrid algorithms (aka. v1.0)

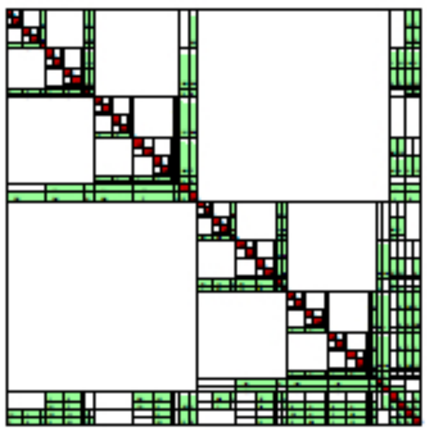
- Scientific computing: iChol, multigrid
- [Vaidya `89] precondition with graphs

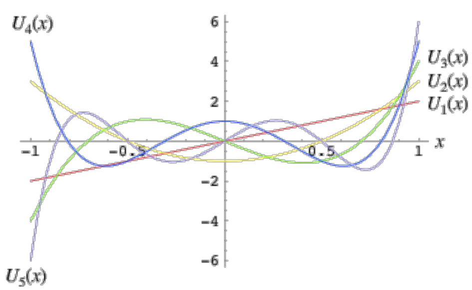




Hybrid algorithms (aka. v1.0)

- Scientific computing: iChol, multigrid
- [Vaidya `89] precondition with graphs



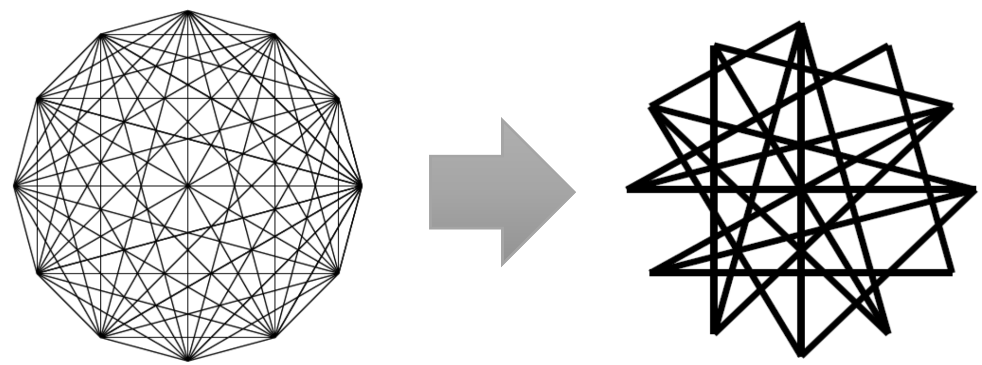


Focus: how to combine

- [Gemban-Miller `96]: spectral graph theory
- [Spielman-Teng `04]: spectral (ultra-)sparsify

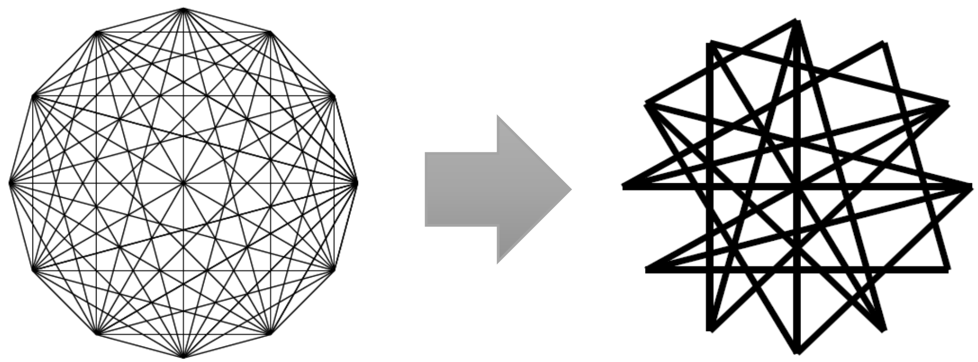
Key “glue”: sparsification

[Spielman-Teng `04]: for any G, can find H with $O(n\log^{O(1)}n)$ edges s.t. $\mathbf{x}^T\mathbf{L}_G\mathbf{x} \approx \mathbf{x}^T\mathbf{L}_H\mathbf{x}\ \ \forall\ \mathbf{x}$



Key "glue": sparsification

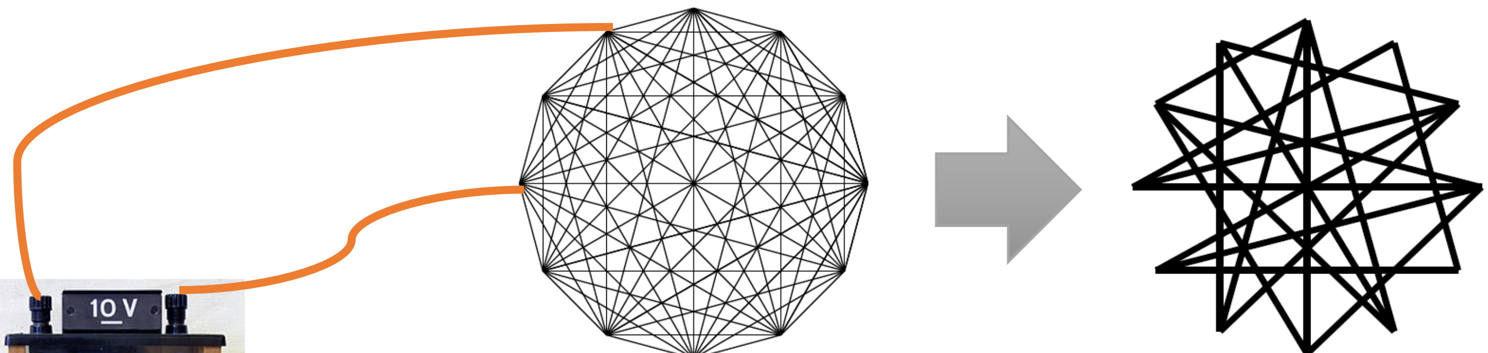
[Spielman-Teng `04]: for any G, can find
H with $O(n\log^{O(1)} n)$ edges s.t. $\mathbf{x}^T\mathbf{L}_G\mathbf{x} \approx \mathbf{x}^T\mathbf{L}_H\mathbf{x} \;\; \forall\, \mathbf{x}$



- Combinatorial parameter: #edges
- Numerical parameter : approximations

Key “glue”: sparsification

[Spielman-Teng `04]: for any G, can find
H with $O(n\log^{O(1)} n)$ edges s.t. $\mathbf{x}^T\mathbf{L}_G\mathbf{x} \approx \mathbf{x}^T\mathbf{L}_H\mathbf{x} \;\; \forall\, \mathbf{x}$



- Combinatorial parameter: #edges
- Numerical parameter : approximations

[Spielman-Srivatava`08]: sample by effective resistances gives H with O(nlogn) edges

Outline

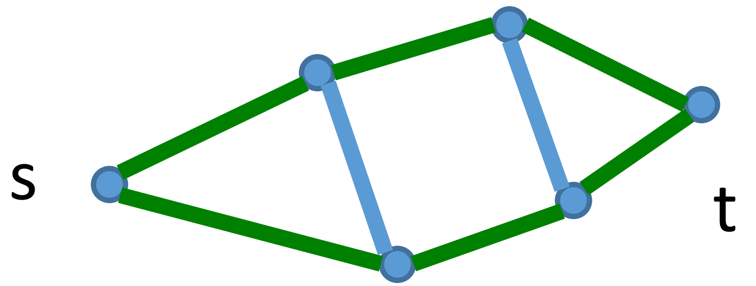
- Graphs and Laplacians
- Building Blocks
- **Laplacian Paradigm 2.0**

Max-Flow problem

Maximum number of disjoint s-t paths

Applications:

- Routing
- Scheduling



Recall: $\min_{\mathbf{f} \text{ with resdiual } \mathbf{b}} \|f\|_p$:

- $p = 2$: solving $\mathbf{Lx} = \mathbf{b}$
- $p = \infty$: max-flow/min-cut

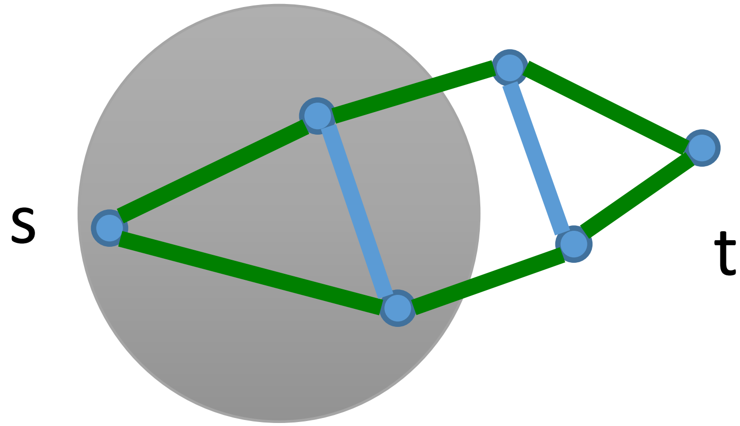
Max-Flow problem

Maximum number of disjoint s-t paths

Dual: separate s and t by removing fewest edges

Applications:

- Routing
- Scheduling



Applications:

- Partitioning
- Clustering

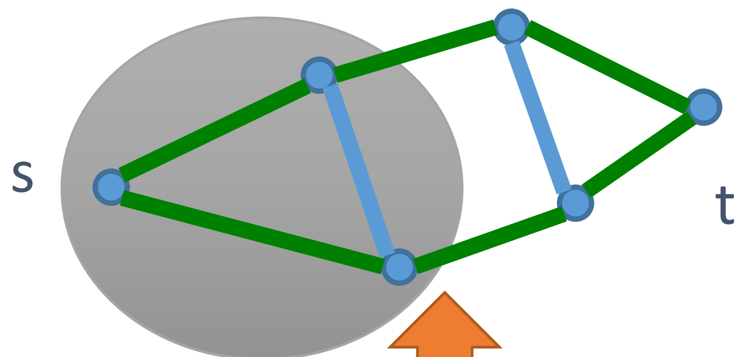
Recall: $\min_{\mathbf{f} \text{ with resdiual } \mathbf{b}} \|f\|_p$:

- $p = 2$: solving $\mathbf{Lx} = \mathbf{b}$
- $p = \infty$: max-flow/min-cut

Hybrid Algorithms for Max-Flow

[Daitch-Spielman `08][Christiano-Kelner-Madry-Spielman-Teng `10]:

[Lee-Sidford `14] Max-flow/Min-cut via (several) electrical flows



Repeat about $m^{1/3}$ iters

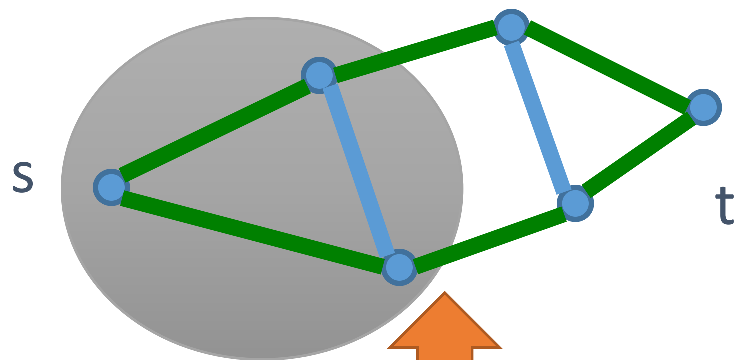
- Solve linear systems
- Re-adjust edge weights

$Lx=b$

Hybrid Algorithms for Max-Flow

[Daitch-Spielman `08][Christiano-Kelner-Madry-Spielman-Teng `10]:

[Lee-Sidford `14] Max-flow/Min-cut via (several) electrical flows



Repeat about $m^{1/3}$ iters

- Solve linear systems
- Re-adjust edge weights

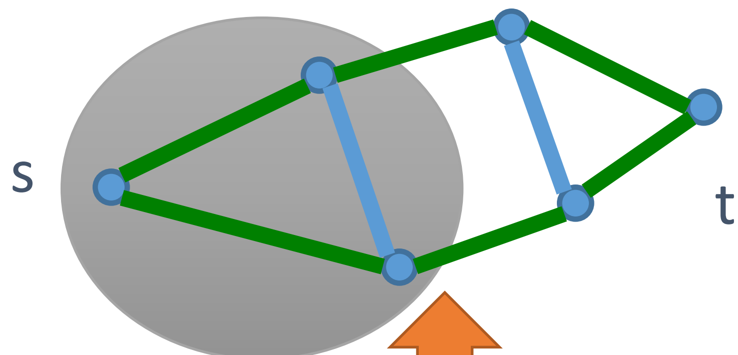
Lx=b

[Madry `10] [Racke-Shah-Taubig `14]: cut approximator / oblivious routing $O(n^{o(1)})$-approx. in $O(m^{1+o(1)})$

Hybrid Algorithms for Max-Flow

[Daitch-Spielman `08][Christiano-Kelner-Madry-Spielman-Teng `10]:

[Lee-Sidford `14] Max-flow/Min-cut via (several) electrical flows



[Lee-Rao-Srivastava `13][Sherman `13, `17][Kelner-Lee-Orecchia-Sidford `14]: Preconditioning, (1+ε)-approx

Repeat about $m^{1/3}$ iters

- Solve linear systems
- Re-adjust edge weights

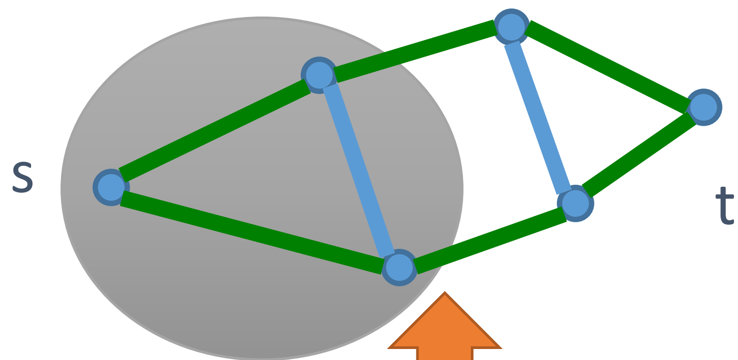
$Lx=b$

[Madry `10] [Racke-Shah-Taubig `14]: cut approximator / oblivious routing $O(n^{o(1)})$-approx. in $O(m^{1+o(1)})$

Hybrid Algorithms for Max-Flow

[Daitch-Spielman `08][Christiano-Kelner-Madry-Spielman-Teng `10]:

[Lee-Sidford `14] Max-flow/Min-cut via (several) electrical flows



Repeat about $m^{1/3}$ iters

- Solve linear systems
- Re-adjust edge weights

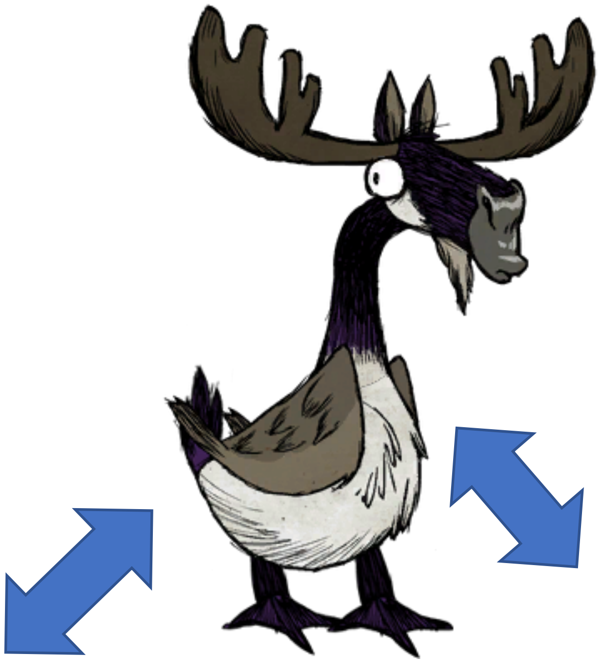
$Lx=b$

[Lee-Rao-Srivastava `13][Sherman `13, `17][Kelner-Lee-Orecchia-Sidford `14]: Preconditioning, $(1+\varepsilon)$-approx

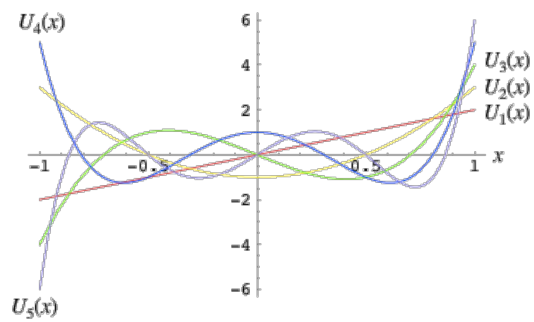
[P`16]: recurse them into each other: $O(m\log^{41}n)$, optimistically $m\log^6 n$

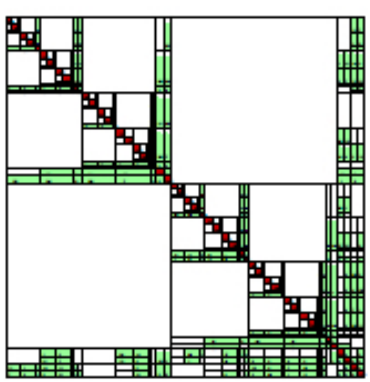
[Madry `10] [Racke-Shah-Taubig `14]: cut approximator / oblivious routing $O(n^{o(1)})$-approx. in $O(m^{1+o(1)})$

Laplacian Paradigm 2.0



Motivated by the goal of hybrid algorithms, modify direct and iterative methods

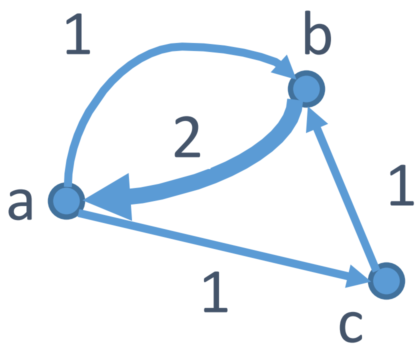


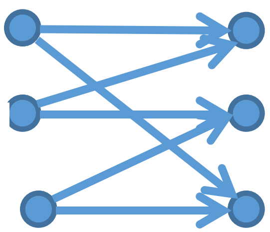


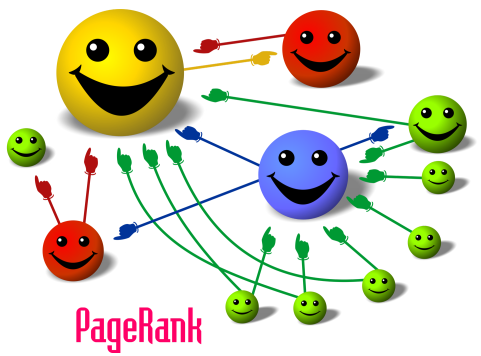
New Intermediate structures / theorems motivated by the overall algorithms

Examples

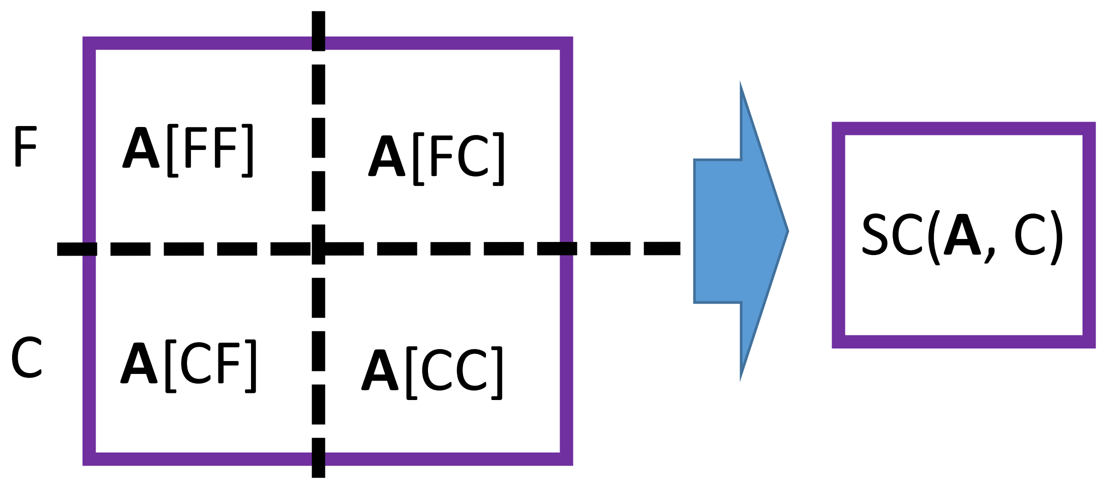
Directed graphs / asymmetric matrices





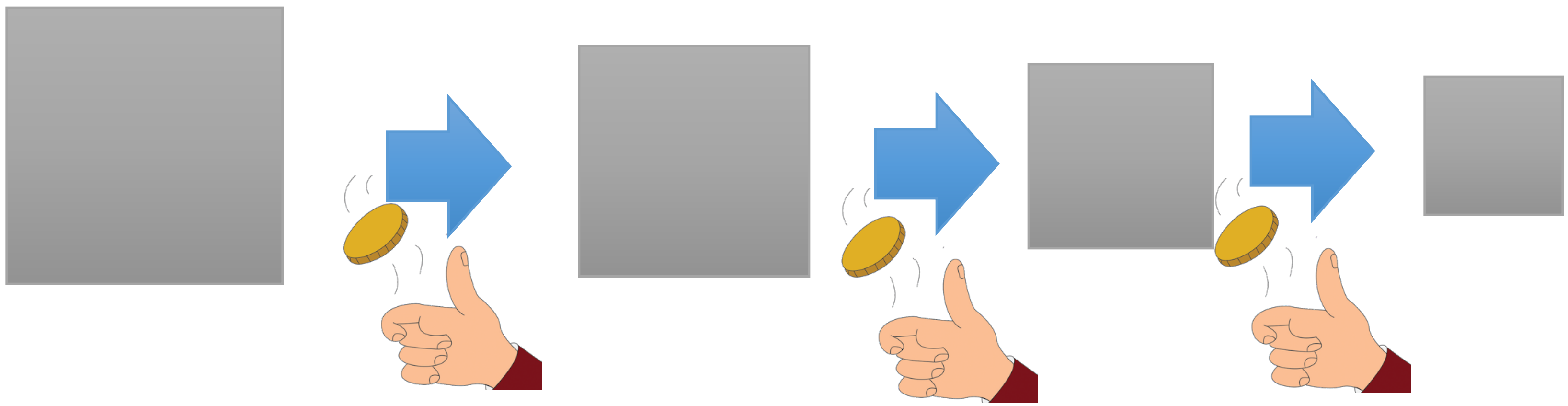


Sparsified/Approximate Gaussian Elimination

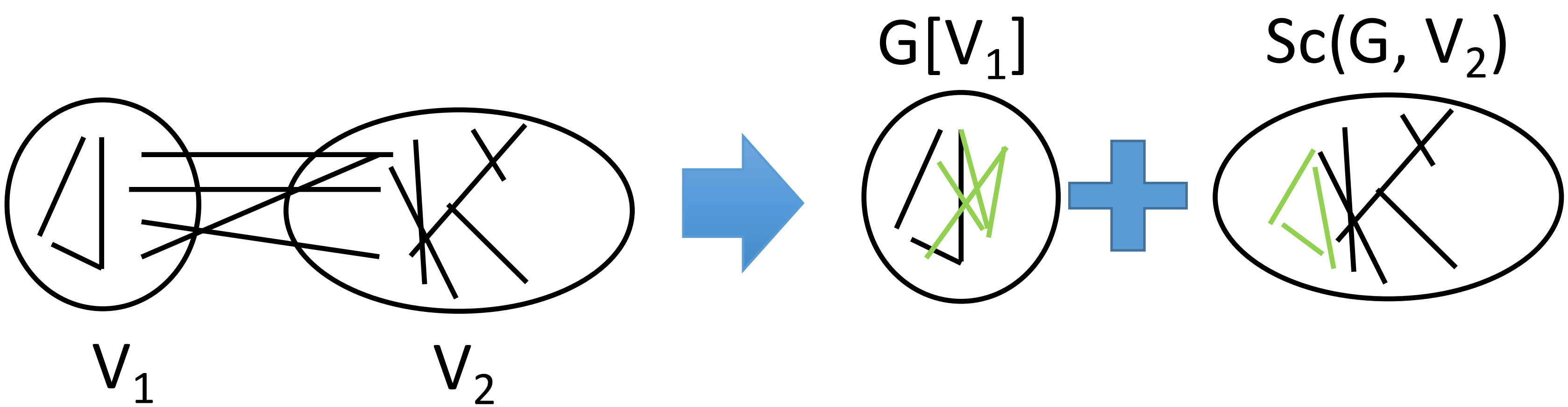


Under the hood

Matrix (martingale) concentration

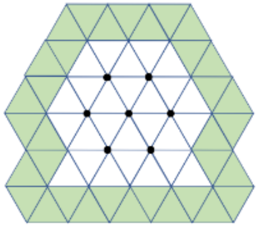


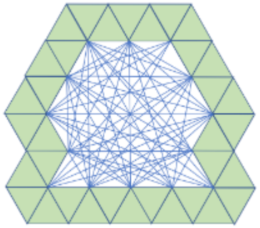
Partitioning / Localizations of Random Walks

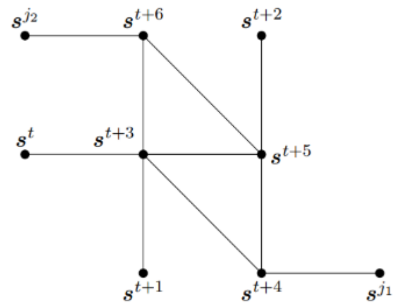


Not covered ☹

Matrix Zoo from Scientific Computing

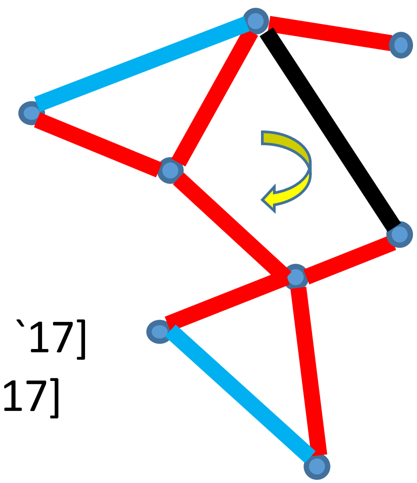


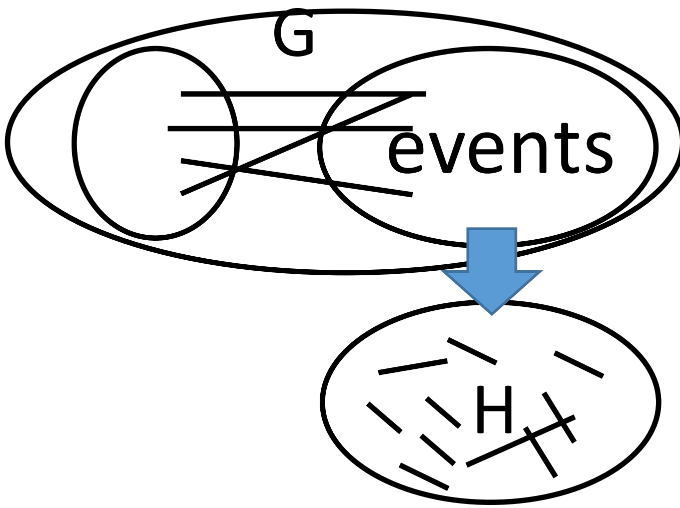




[Boman-Hendrickson-Vavasis `04]
[Kyng-Lee-P-Sachdeva-Spielman `16]
[Kyng-Zhang `17][Kyng-P-Schweiterman-Zhang `18]

Interactions with data structures

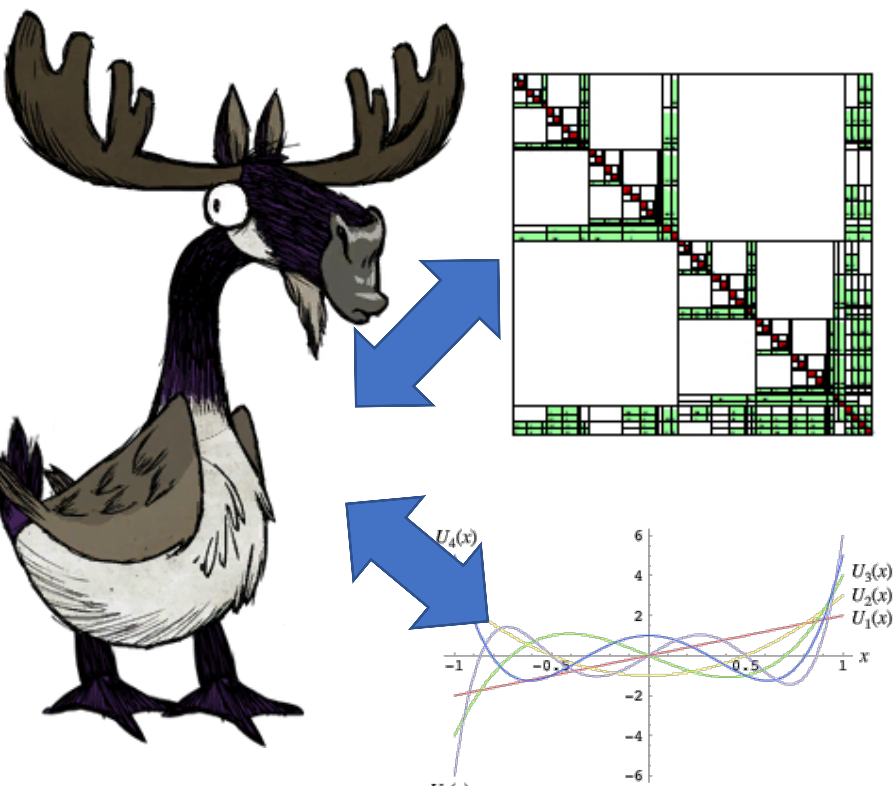




[Kelner-Orecchia-Sidford-Zhu `13]
[Nanongkai-Saranuk `17][Wulff-Nilsen `17]
[Durfee-Kyng-Peebles-Rao-Sachdeva `17]

Questions

Wist list	Direct	Iterative	Hybrid
Convex functions	?	☺	☺☺??
Arbitrary values	☺	?	?☹☺!
Dynamic/streaming	☺	☹	☺???



Approximate eliminations beyond spectral condition #

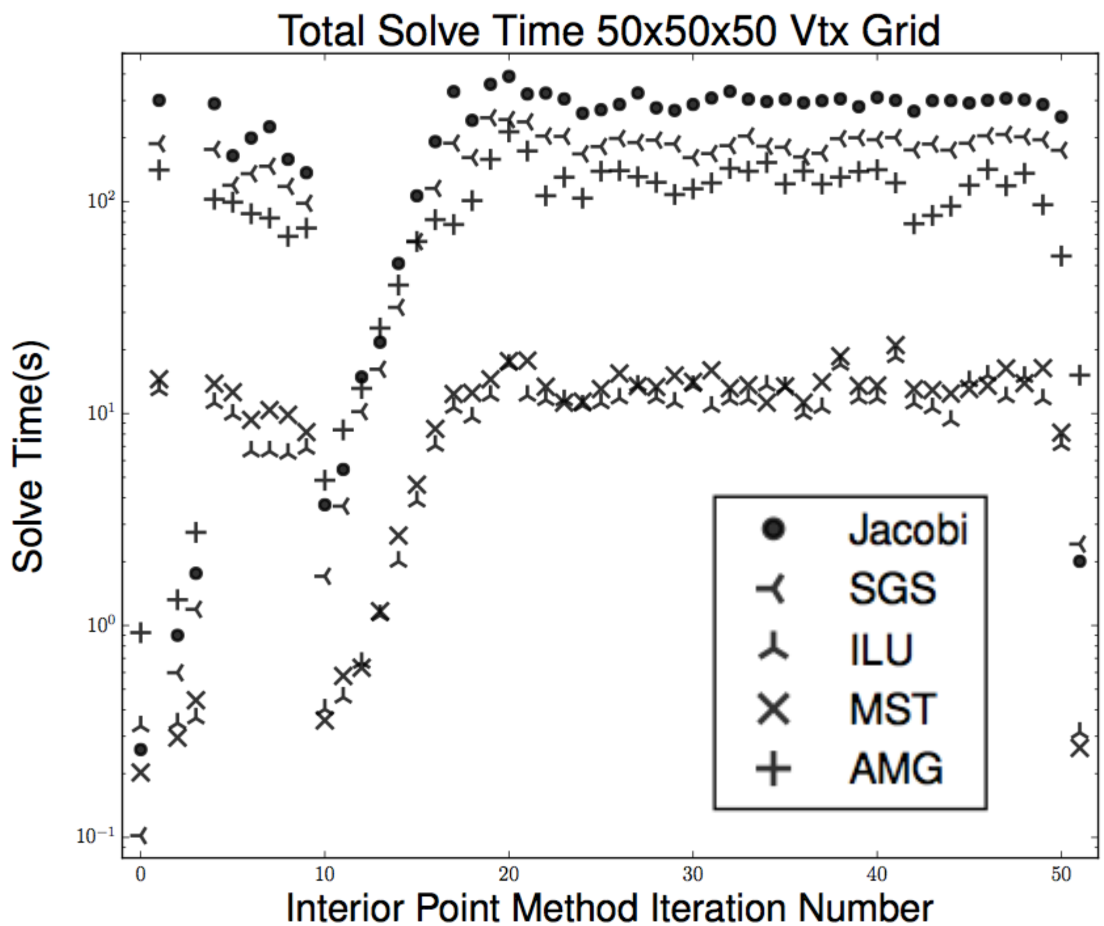
Unreasonable effectiveness of pcg(ichol(A), b), multigrid

Non-linear (preconditioned) iterative methods

[Adil-Kyng-P-Sachdeva `19]: p-norm iterative refinment

Solvers in Practice

[Kyng-Rao-Sachdeva `15] we suggest rerunning the program a few times… An alternate solver based on iChol is provided…



julia

Questions:

- Precision
- (pseudo) deterministic