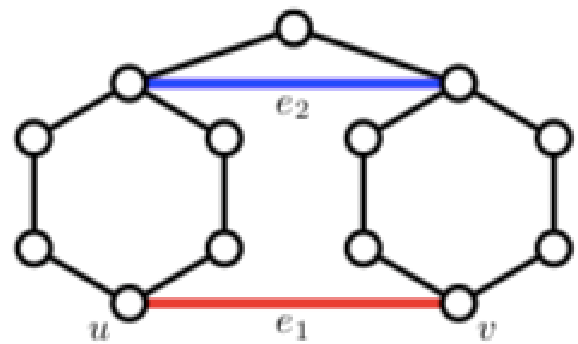
Scalable Algorithmic Primitives for Data Science

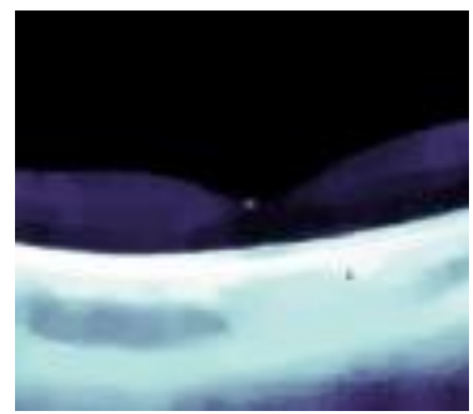
Richard Peng

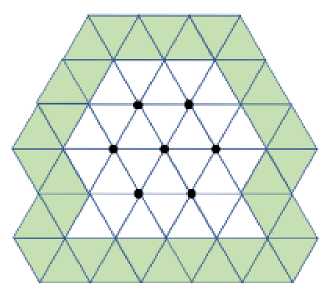
Sep 25, 2018

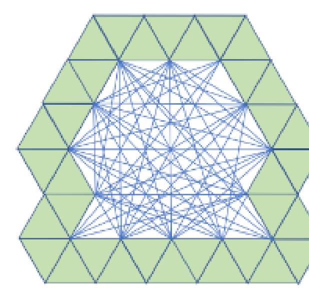
Large Scale graphs / matrices

- Network science:
  centrality, partitioning …
- Image/video processing:
  segmentation, denoising …
- Scientific computing:
  stress/strain, heat/fluid, waves …









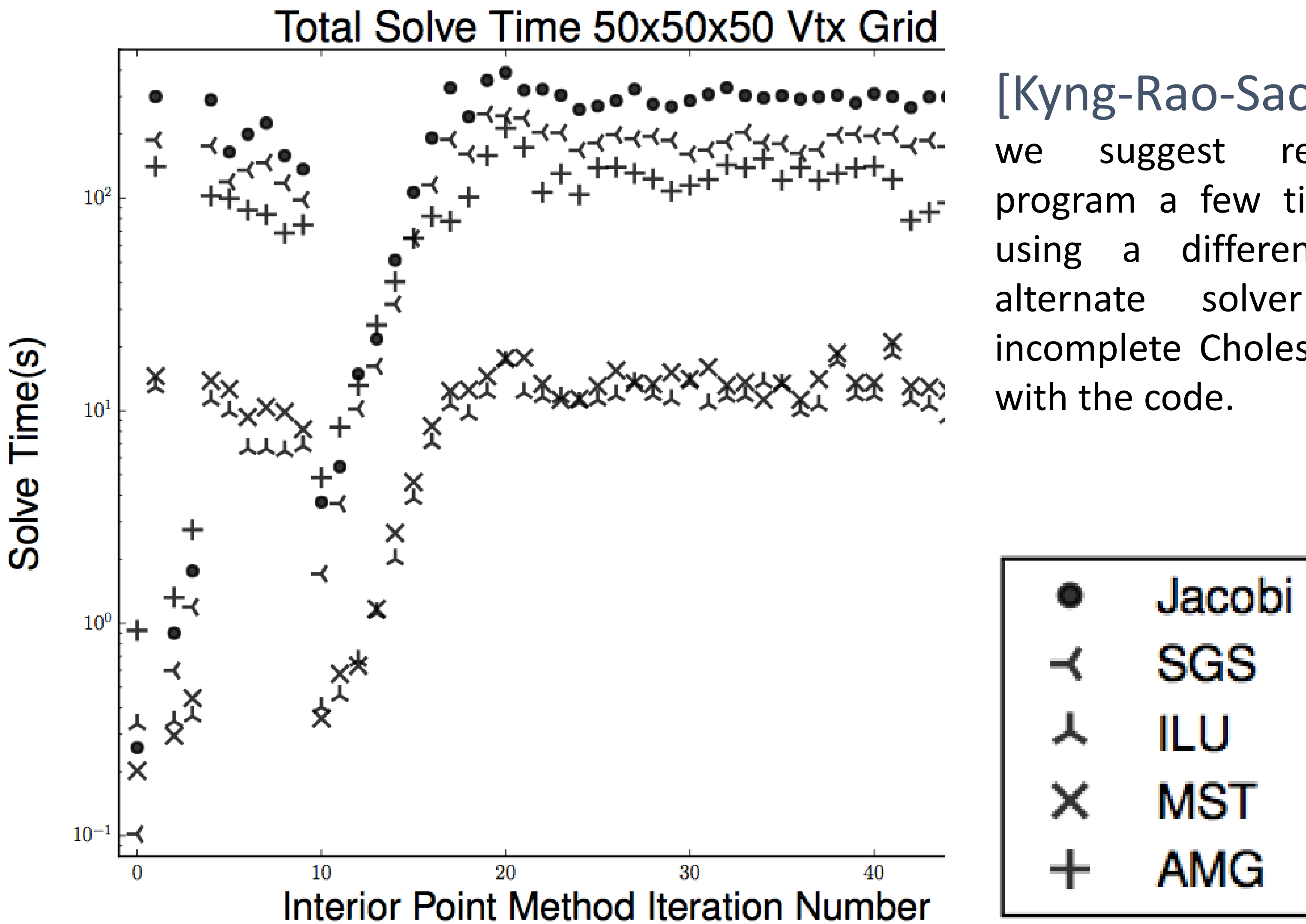
Tools for large graphs / matrices

- \ (linear system solve)
- CVX (convex optimization)
- Eigenvector / SVD / spectral algorithms

Most basic: solving **Ax = b**

- Optimization: interior point method
- Eigenvector: inverse power method

x=Solve(**A**, **b**) vs. sorting



[Kyng-Rao-Sachdeva `15]
we suggest rerunning the program a few times and / or using a different solver. An alternate solver based on incomplete Cholesky is provided with the code.

Works on solving **Ax = b**

from *SIAM News*, Volume 33, Number 4

The Best of the 20th Century: Editors Name Top 10 Algorithms

- matrix decompositions,
- QR factorization,
- Krylov space methods (e.g. conjugate gradient)

On a laptop: many instance with $m \approx 10^6$ solvable in seconds

Open: provably solve ALL graphs / matrices problems this fast

This talk: recent progresses and the central role of high dimensional concentration

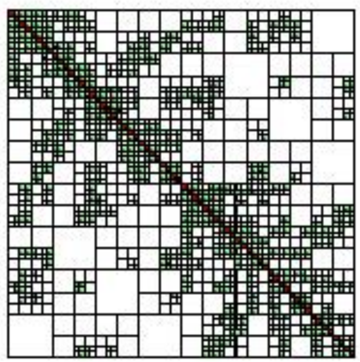
Focus: linear case, but most have non-linear extensions

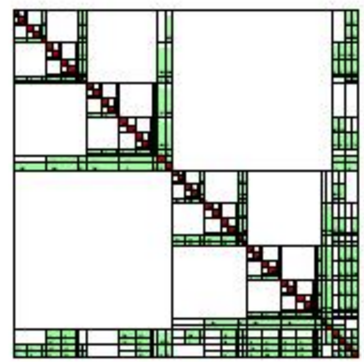
Direct Methods (combinatorial)

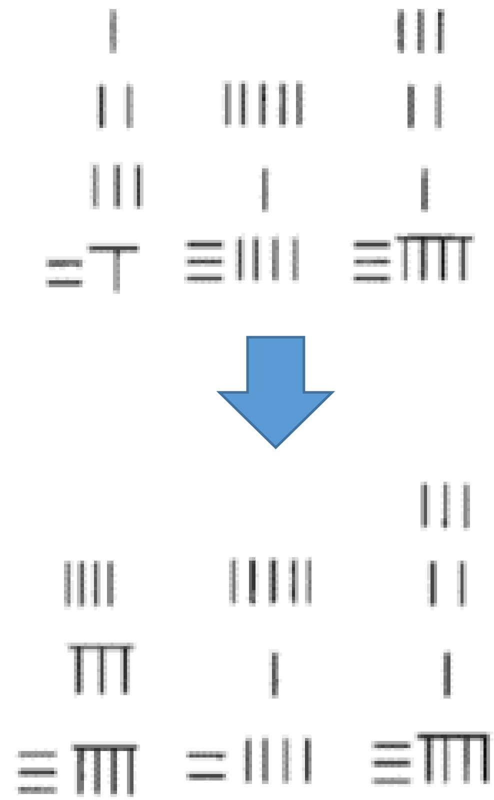
$M^{(2)} \leftarrow \text{Eliminate}(M^{(1)}, x1)$

$M^{(3)} \leftarrow \text{Eliminate}(M^{(2)}, x1)$

…

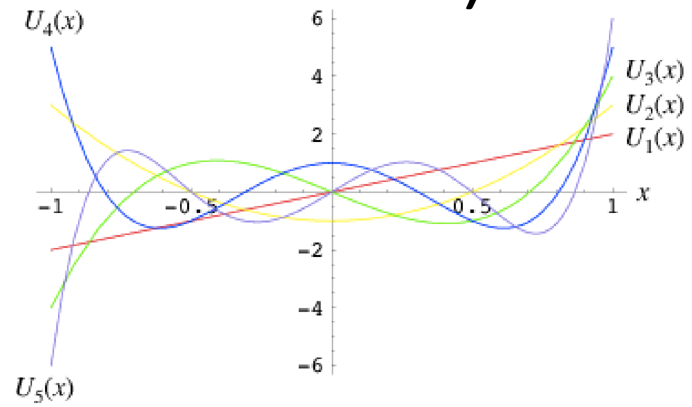






- Combinatorial scientific computing
- Matrix multiplication
- Parallel graph algorithms
- Sparsified squaring (e.g. connectivity in L)

Iterative Methods (numerical)



Preconditioning:

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

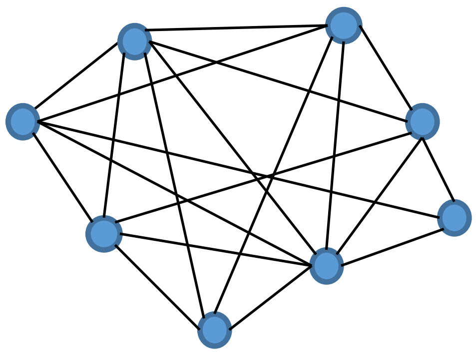
$\mathbf{x} \leftarrow \mathbf{x} + \mathbf{B}^{-1}(\mathbf{A}\mathbf{x} - \mathbf{b})$

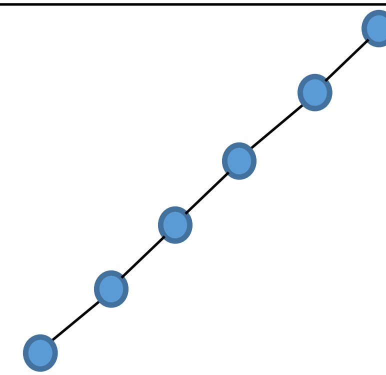
- Fixed point: $\mathbf{A}\mathbf{x} - \mathbf{b} = 0$
- Simple **B**: $\mathbf{B} = \mathbf{I}$, many iterations
- $\mathbf{B} = \mathbf{A}$: 1 iteration, but same problem

- Conjugate gradient (pcg)
- Convex optimization algorithms
- Krylov space methods

Difficulties in scaling

High performance computing: nonzeros $\Leftrightarrow$ edges





Highly connected,
need global steps

Long paths / trees,
need many steps

Each easy on its own

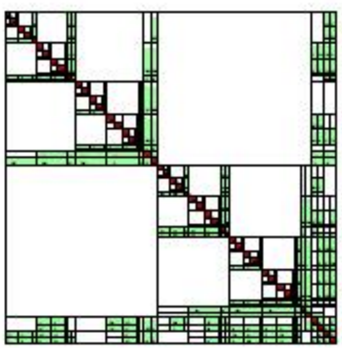
Iterative methods

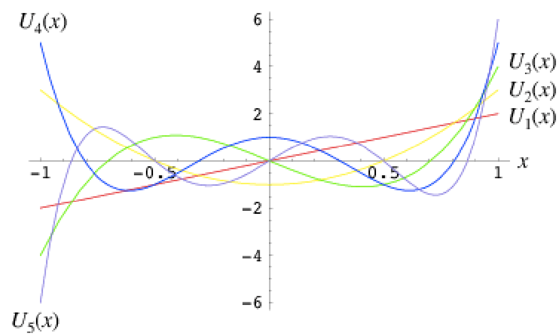
Direct methods

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

Hybrid algorithms

- Approximate Gaussian elimination (pcg + ichol)
- [Vaidya `89] precondition with graphs





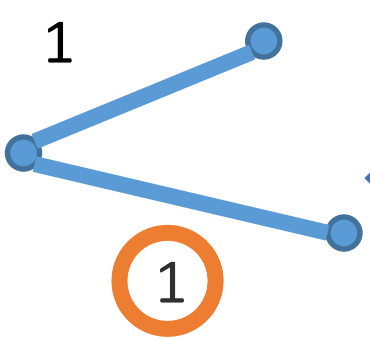
Algorithmic view:

- Operator error as another resource
- Fined grained coupling of discrete/continuous

Graph Structured Matrices

graph Laplacian Matrix **L**

- Diagonal: degree
- Off-diagonal: -edge weights



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

Laplacians arise in:

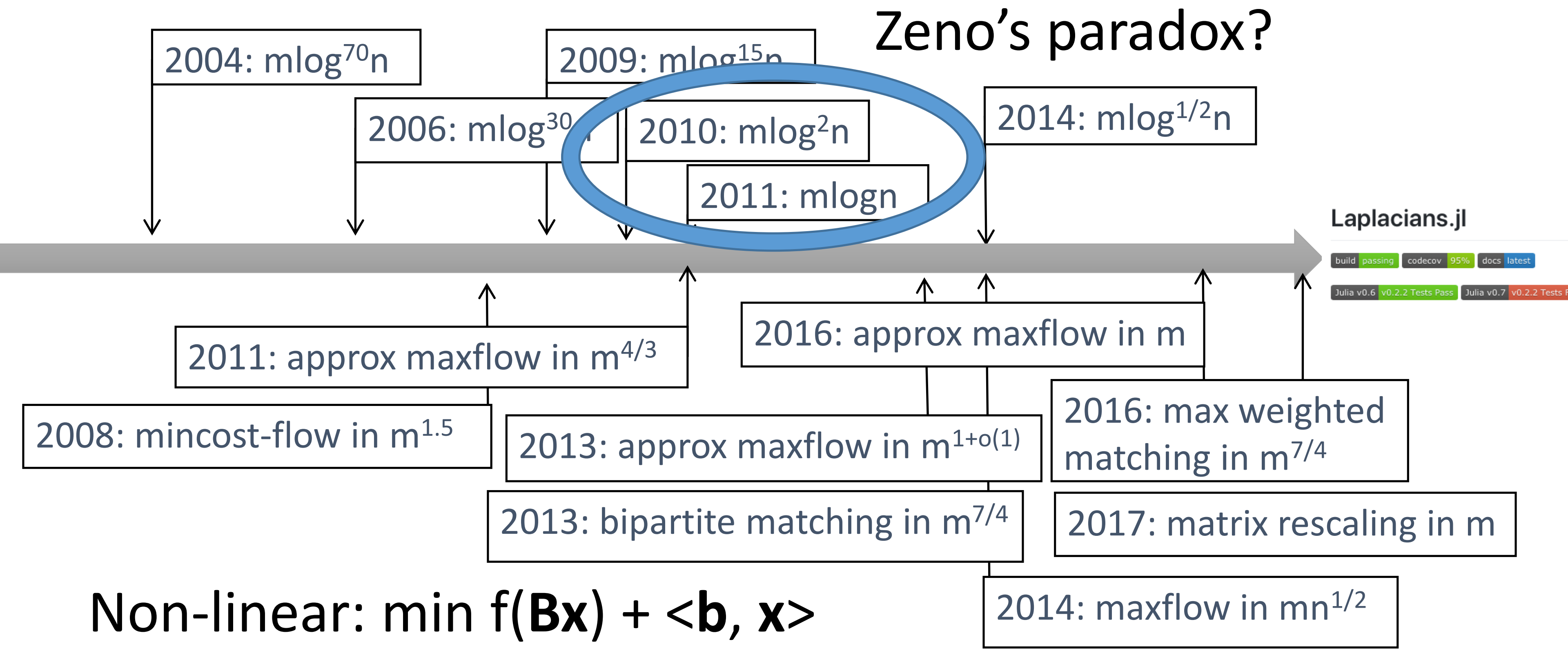
- Spectral algorithms
- Inference on graphs
- Hessian matrices of IPM

n vertices
m edges

n rows / columns
O(m) non-zeros

Laplacian Paradigm

[Spielman-Teng `04] find **x** s.t. **Lx** = **b** in nearly-linear time

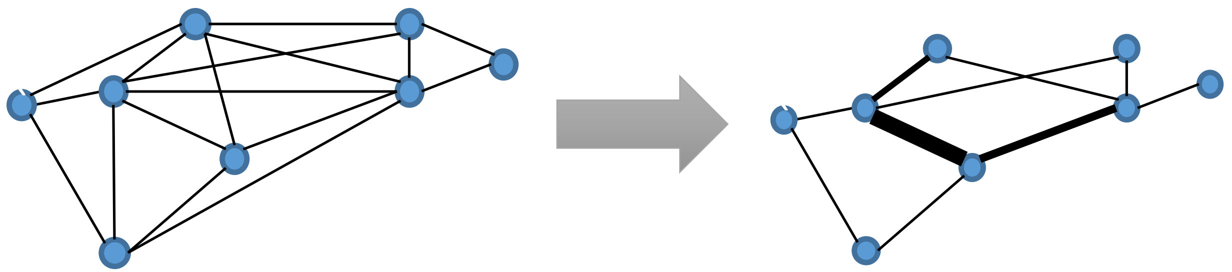


Non-linear: min f(**Bx**) + <**b**, **x**>

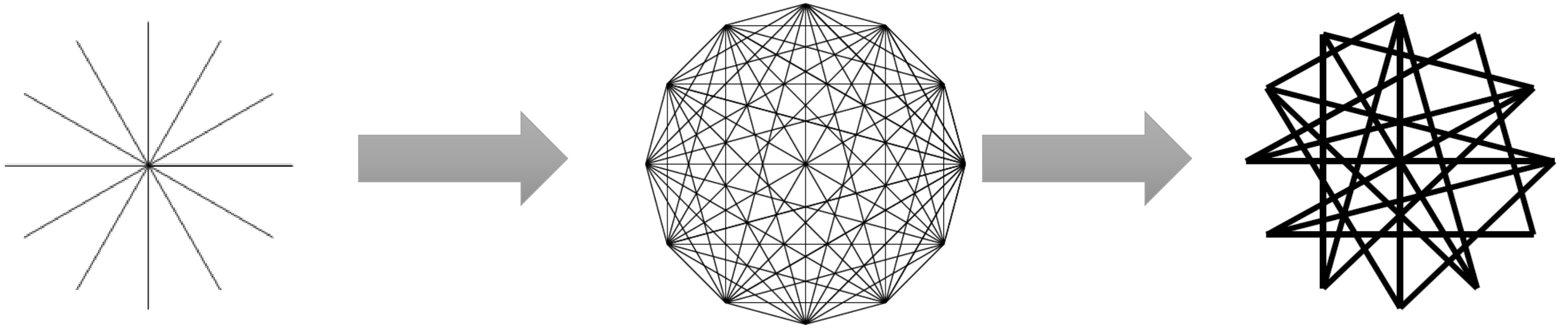
Where **B** = edge-vertex incidence matrix

Laplacian Paradigm(s)

- [Vaidya `89, Spielman-Teng `04, Koutis-Miller-P `10, `11…, Kelner-Orecchia-Sidford-Zhu `13, Lee-Sidford `14…]
  Turn graph into tree by removing off-tree edges



- [Gremban-Miller `96, P-Spielman `14, Kyng-Lee-P-Sachdeva-Spielman `16, Kyng-Sachdeva `16, Cohen et al. `16, `17, `18]:
  Turn graph into clique(s) while eliminating vertices



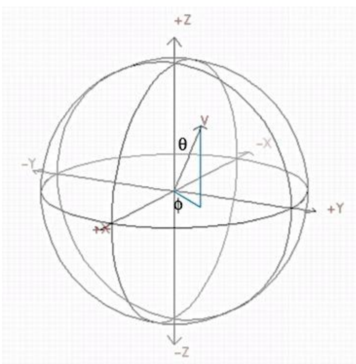
Hybrid algorithms at a glance

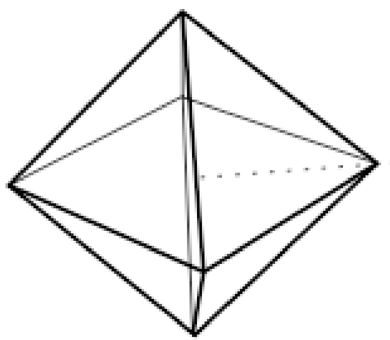
	Ultra-sparsifiers	Elimination
End goal	Tree	Expander
Progress	#edges	Condition number
Reduction / step	Factor of k	Factor of 2
Error / step	$O(k \log^2 n)$	$1/O(\log n)$
Objects sampled	Off-tree edges	Walks
Building upon	DFS / BFS / MST Global Min-Cut	Multi-grid, Connectivity in L

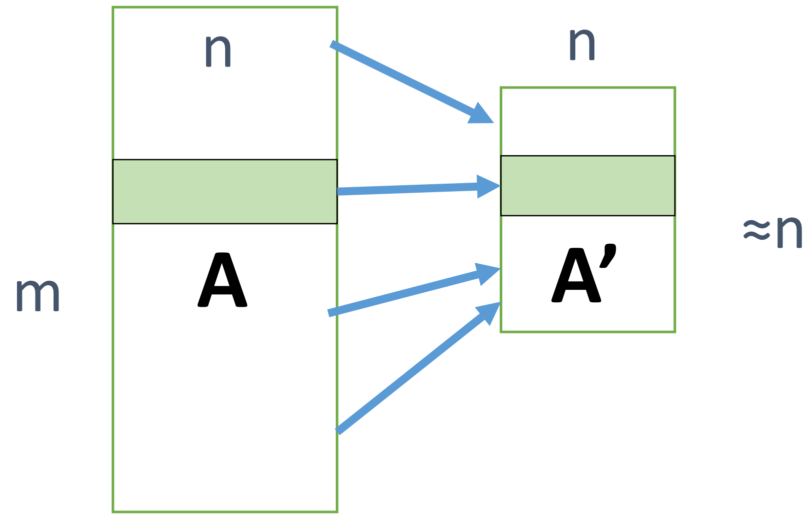
Core step: gradual transfer between sizes and numerical complexities

Dimensionality Reduction

f(**Ax**): Banach space







Can work on **A'** instead: f(**A'x**) ≈ f(**A**x) ∀**x**:
⇔ min f(**Ax**) + <**b**, **x**> ≈ min f(**A'x**) + <**b**, **x**>

Functional analysis (e.g. (Talagrande `90]): for many f, including p-norms with 1 <= p <= 2, any n-dimensional Banach space embeds with constant error into $R^{O(n\log n)}$

p-norm: $\| \mathbf{y} \|_q := (\Sigma |\mathbf{y}_i|^p)^{1/p}$

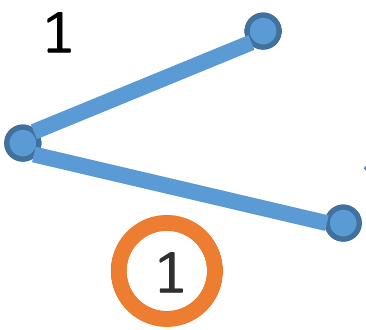
Edge-vertex incidence matrix

graph Laplacian Matrix **L**

- Diagonal: degree
- Off-diagonal: -edge weights

edge-vertex incidence matrix

$\mathbf{B}_{eu}$ = -1/1 for endpoints u
0 everywhere else



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

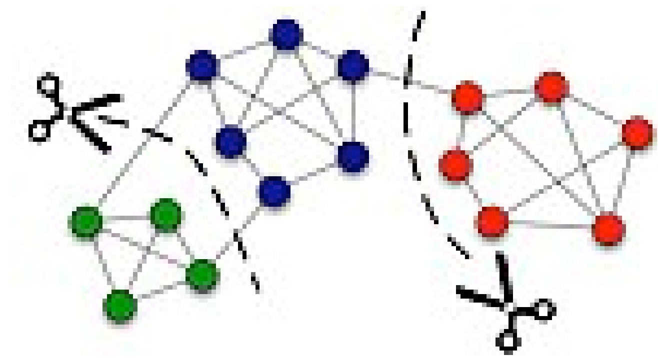
n vertices
m edges

n rows / columns
O(m) non-zeros

m rows
n columns

L is the Gram matrix of **B**, $\mathbf{L} = \mathbf{B}^T\mathbf{B}$

Implications of $||B_G x||_2 \approx ||B_H x||_2$



G ≈ **H** on all cuts: $\mathbf{x} = \{0, 1\}^V$:

- For edge e = uv, $(\mathbf{B}_{e:}\mathbf{x})^2 = (\mathbf{x}_u - \mathbf{x}_v)^2$
- $||\mathbf{B}_G\mathbf{x}||_2^2$ = size of cut given by **x**

Operation approximations:

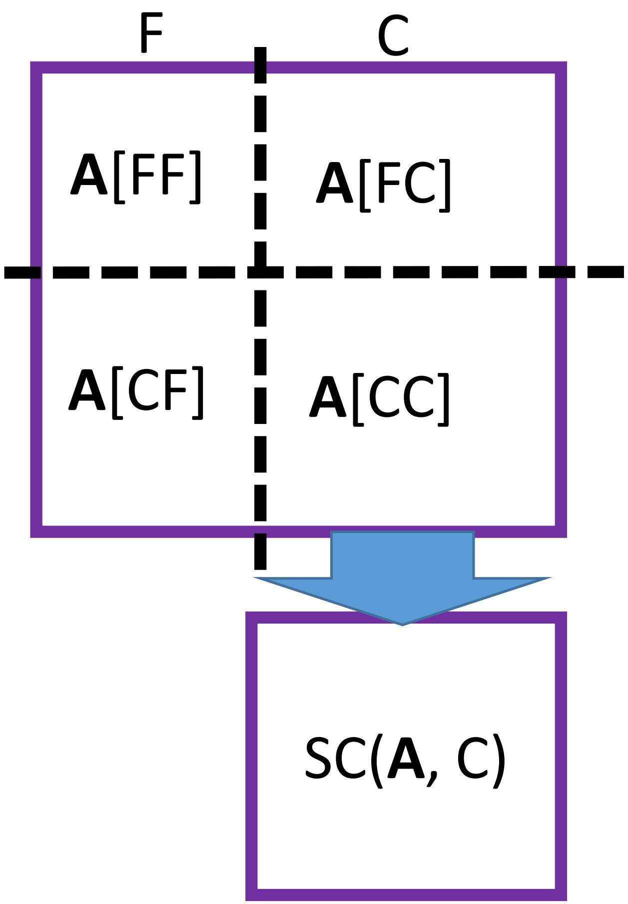
- $\mathbf{L}_G = \mathbf{B}_G{}^T\mathbf{B}_{G:}$
- $\mathbf{x}^T\mathbf{L}_G\mathbf{x} = ||\mathbf{B}_G\mathbf{x}||_2^2 \Leftrightarrow \mathbf{L}_G \approx \mathbf{L}_H$

- Undirected graph have sparse approximations
- Key: outputs of randomized methods structured

Fine Grained Incorporation of ≈

Schur complement, SC(**A**, C) **A**[CC] – **A**[CF] **A**[FF]$^{-1}$**A**[FC]

eliminate all variables in F = V \ C



[Strassen `69]: suffices to invert:

- **A**[FF]:|F|-by-|F|
- SC(**A**, C):|C|-by-|C|

SC(**A**, C) is another graph Laplacian!

[Kyng-Lee-P-Sachdeva-Spielman `16]: Sparsify(SC(**L**, C)) without building it

- O(mlogn + nlog^{2}n) overall
- extensions to connection Laplacians

Algorithmic issues

- How to construct / sample SC(L, C) efficiently
- Similar issues in the graph ➔ tree approach

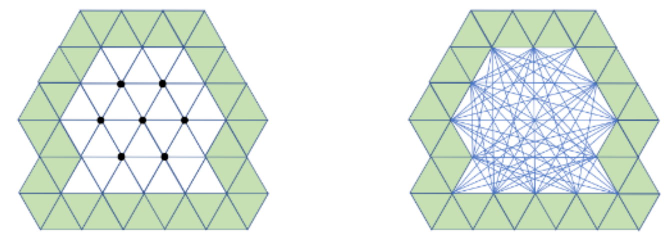
Algorithmic kitchen sink applicable:

- <u>Embeddings</u>: **Lx** = **b**, max-flow, sketching,
- <u>Spanners</u>: **Lx** = **b**, max-flow, sketching
- <u>Data structures</u>: **Lx** = **b**, streaming settings
- <u>Matrix martingales</u>: **Lx = b**, directed graphs
- <u>Recursion</u>: **Lx** = **b**, max-flow, row sampling, directed graphs, connection Laplacians

ELIMINATING MORE

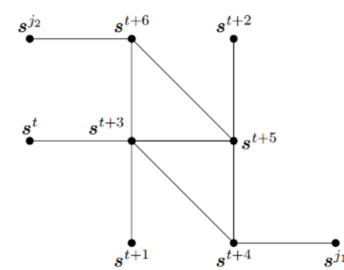
Linear elasticity problems: physical forces on trusses

[Kyng-P-Schweiterman-Zhang STOC`18]: O($n^{5/3}$) time for trusses on well-shaped simplicial complexes



[Kyng-Zhang FOCS`17]: **any** PSD matrix is the partially eliminated state of:

- A generalized 2-D truss matrix
- A 2-commodity flow matrix



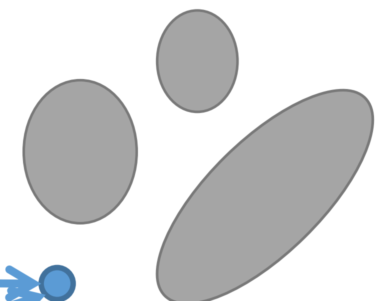
Reversibility of eliminations → trusses are 'complete' for all **Mx** = **b**!

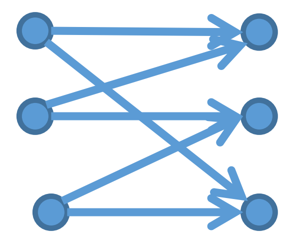
DIRECTED SPECTRAL METHODS

[Cohen-Kelner-Peebles-P-Rao-Sidford-Vladu `16, 17]
sparse approx. of directed graph,
and solving directed **Lx** = **b** in nearly-linear time

Difficult in general:

- Undirected : connected components
- Directed reachability: $\Omega(n^2)$ bits





Key: use states of iterative methods to restrict the information that need to be preserved

Questions / Future directions

- Generalizations of high-dimensional concentration?
- How much of these work in non-linear cases?
- Dynamic / streaming via. adaptive sampling?

Property	Direct	Iterative	Hybrid
Convex functions	?	☺	☺☺??
Arbitrary values	☺	?	?☹☺!
Dynamic / streaming	☺	☹	☺???
Sparse / low memory	☹	☺	☺
Parallelizable	☺	☹	☺
☺ on trees	☺	☹	☺
☺ on well-connected	☹	☺	☺