Parallel, high-performance min-cut solvers and tall-and-skinny matrix-based data analysis

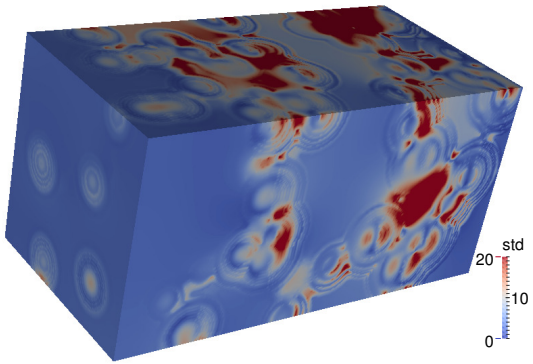
Joint work with
Austin Benson, Jason Lee (Stanford) Bartek Rawja (Purdue), Paul Constantine (Col. Mines) Jeremy Templeton (Sandia), Yangyang Hou (Purdue), Yao Zhu (Purdue), Michael Mahoney (Berkeley) Supported by NSF CAREER 1149756-CCF, and DOE ASCR award

David F. Gleich
Purdue University

My mission *Design the best models and associated algorithms for fast and robust analysis of data.*

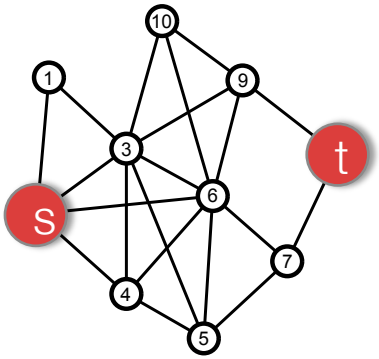
Big data methods

ICASSP '12, BigData '13, SISC' 14, NIPS '14



Parallel methods for graph algs

Submitted



Massive matrix computations

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

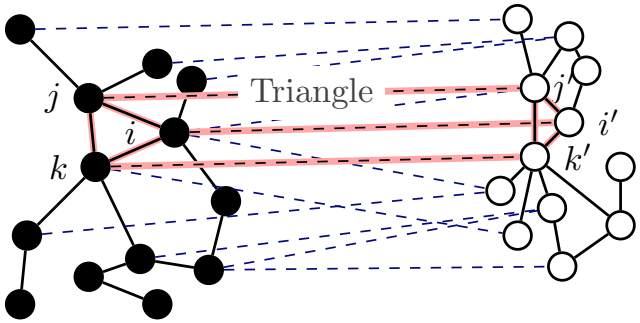
$$\min \|\mathbf{Ax} - \mathbf{b}\|$$

$$\mathbf{Ax} = \lambda \mathbf{x}$$

on multi-threaded and distributed architectures

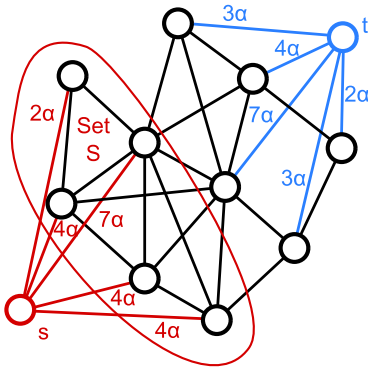
Network alignment

ICDM '09, SC '11, TKDE '13, More in prep



Robust graph-based learning

ICML '14, KDD '15

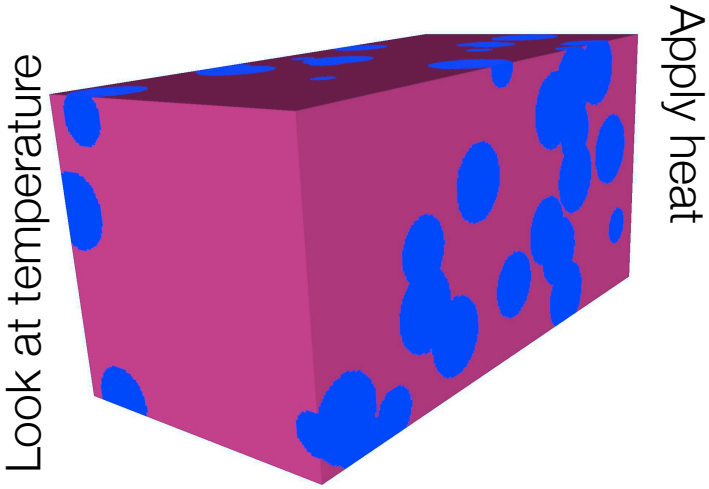


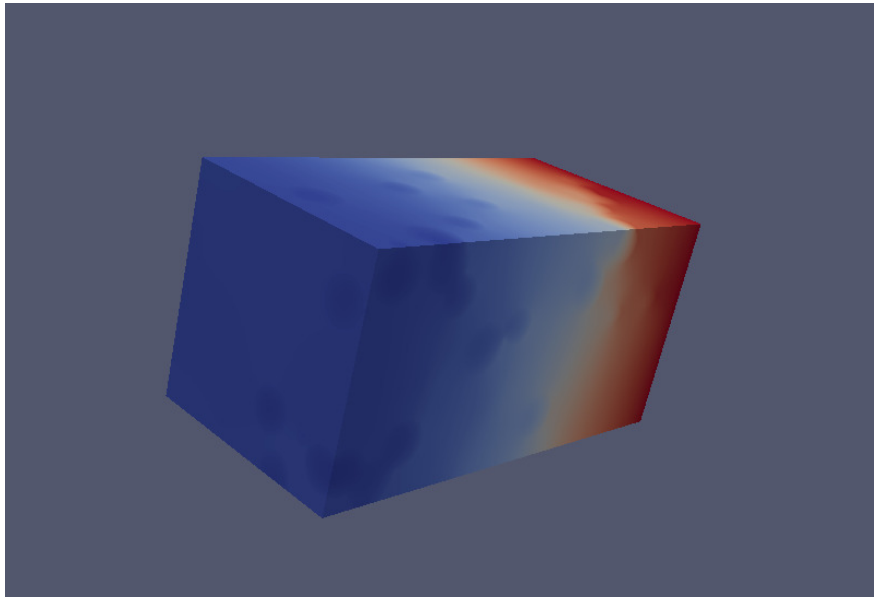
David Gleich · Purdue

Simulation analysis and large-scale tall-and-skinny matrix factorizations

Nonlinear heat transfer model in random media

Each run takes 5 hours on 8 processors, outputs 4M (node) by 9 (time-step) simulation



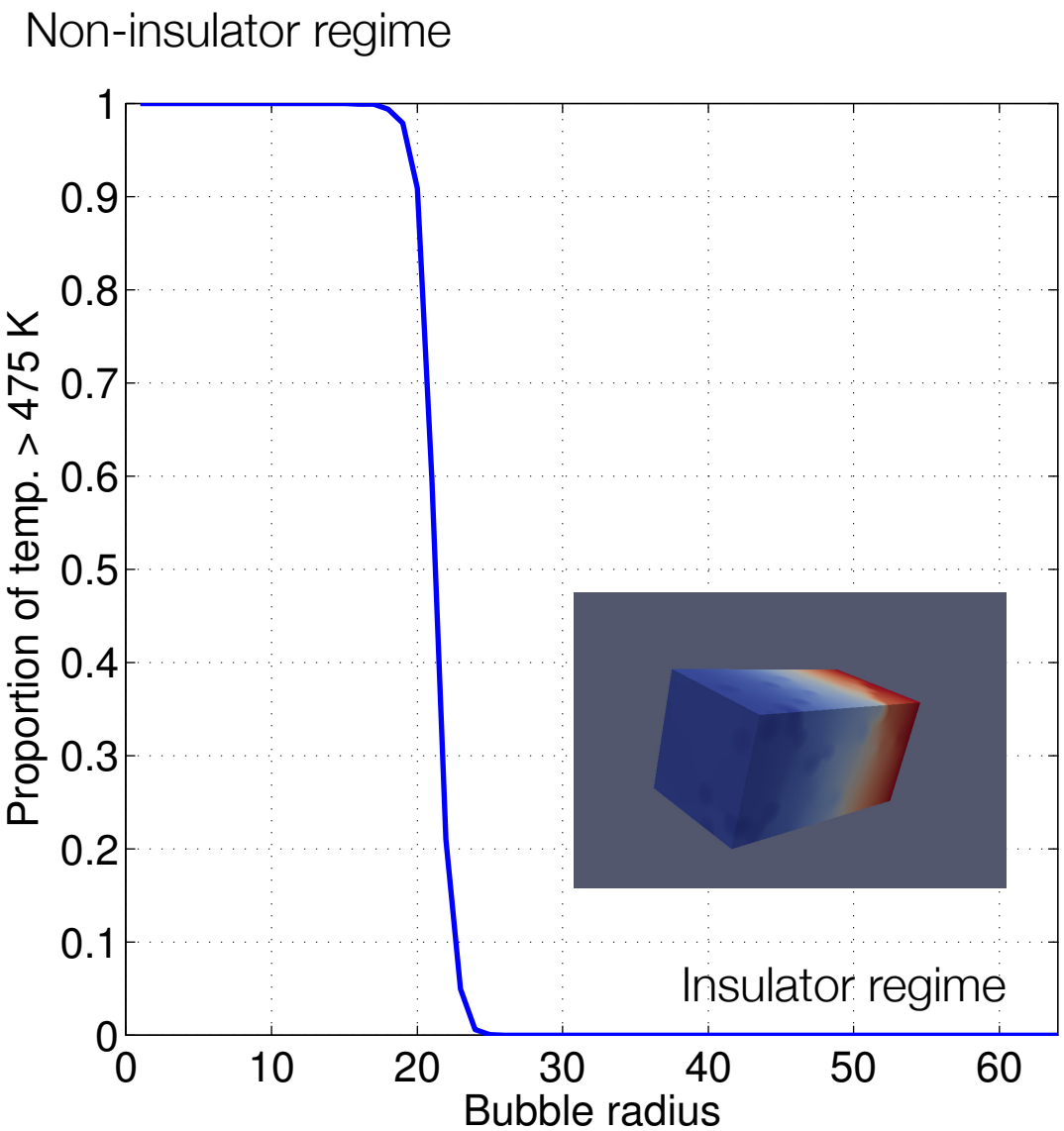


We did 8192 runs (128 samples of bubble locations, 64 bubble radii)

4.5 TB of data in Exodus II (NetCDF)

https://www.opensciencedatacloud.org/publicdata/heat-transfer/

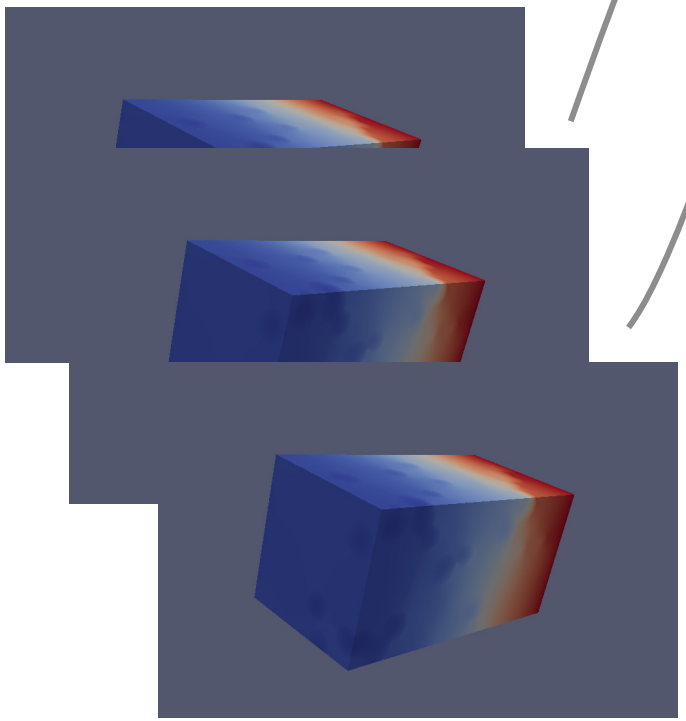
Constantine et al. SISC 2014

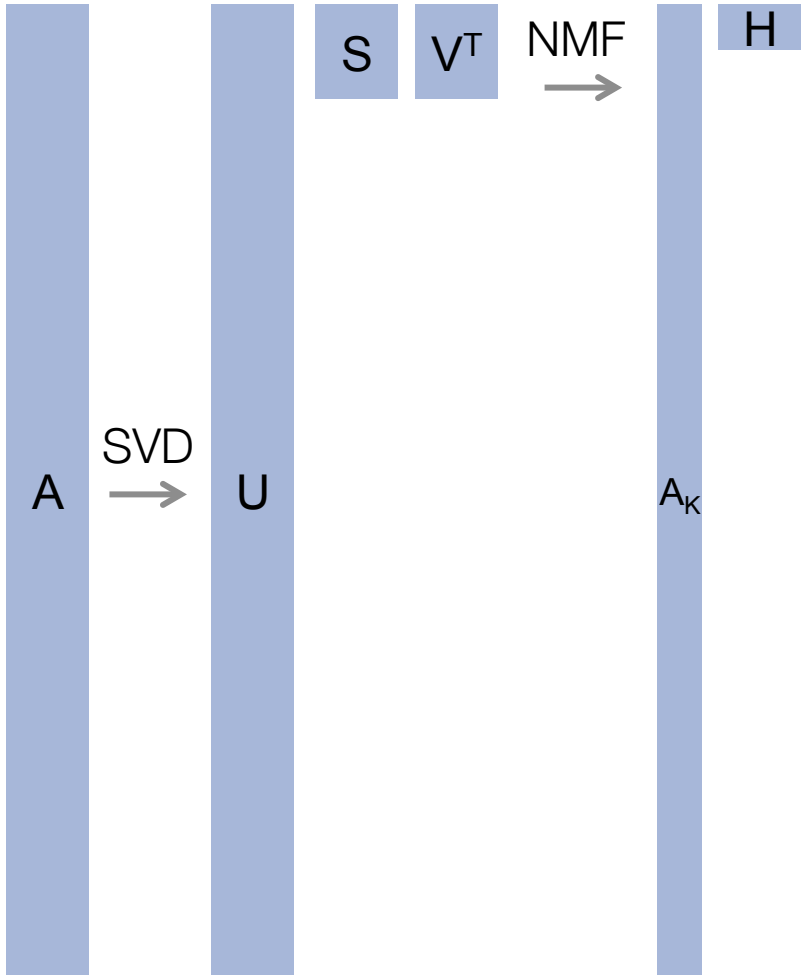


We used tools like the ones I'll describe to build a data-driven surrogate that is about 1000-times faster than a simulation in bulk.

Constantine et al. SISC 2014

Run a "standard" NMF algorithm on SV^T





Each simulation is a column

5B-by-64 matrix
2.2TB

Benson et al. NIPS 2014

Tall-and-Skinny matrices

A

(m ≫ n)

Many rows (like a billion)

A few columns (under 10,000)

From tinyimages collection

Used in

regression and general linear models with many samples

customer data analysis?

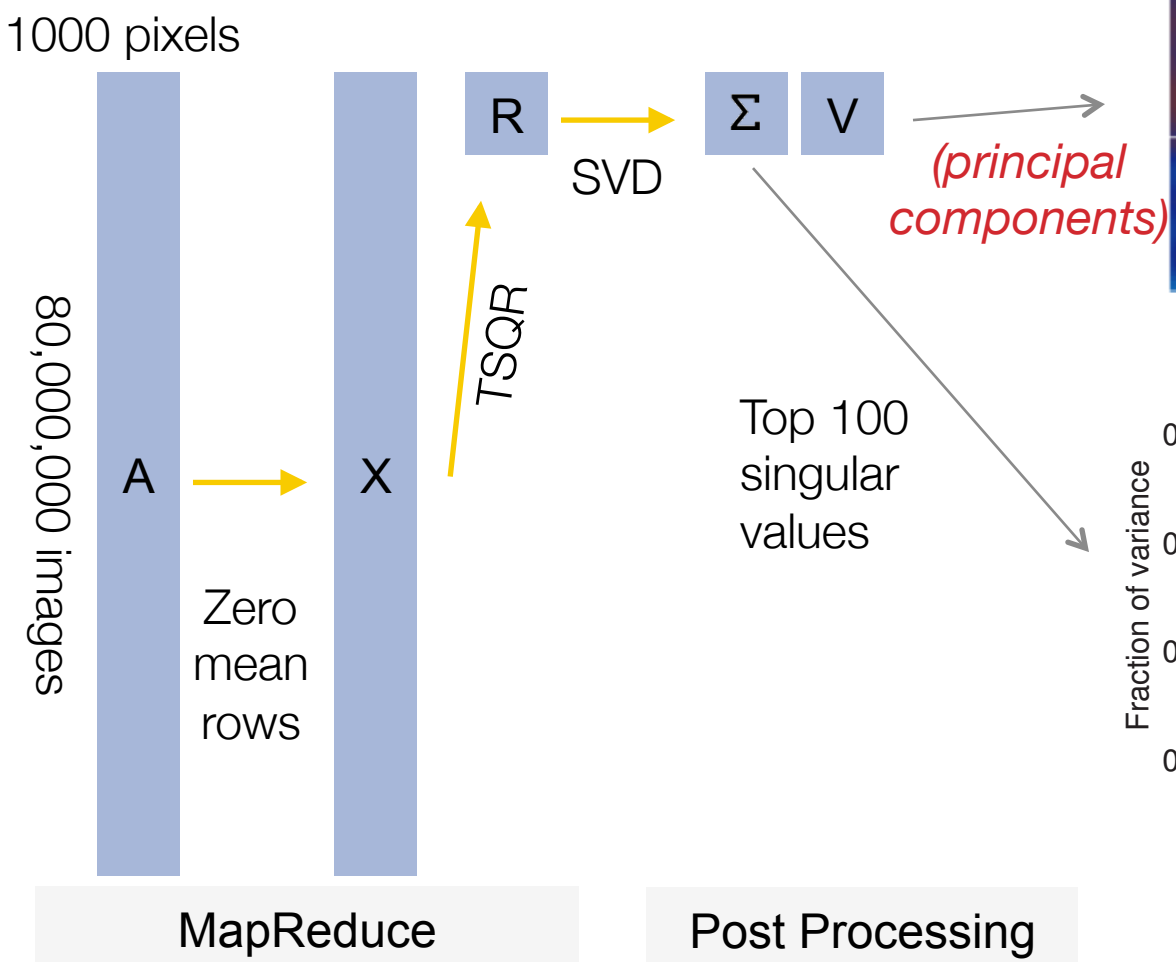
big simulation data

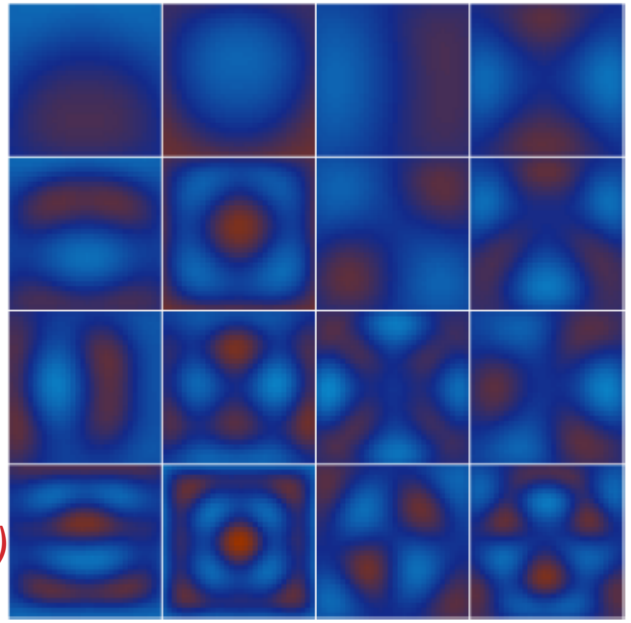
Nyström kernel approximations

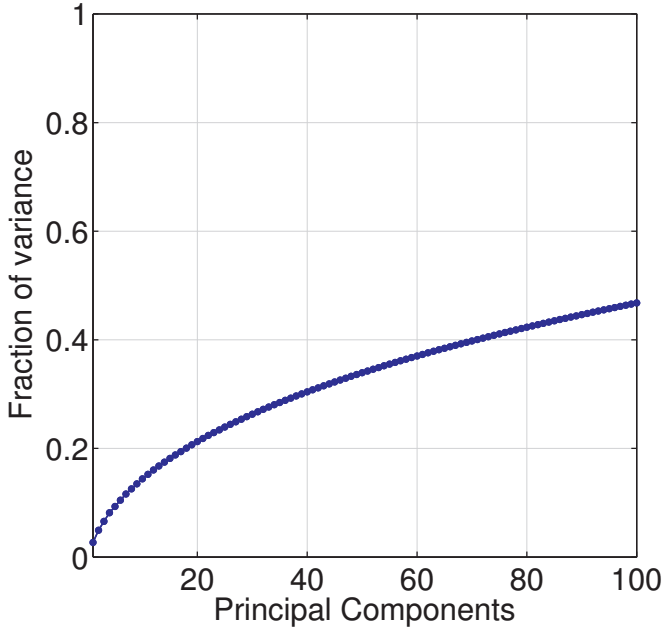
tensor methods too

PCA of 80,000,000 images

Constantine & Gleich, MapReduce 2011.





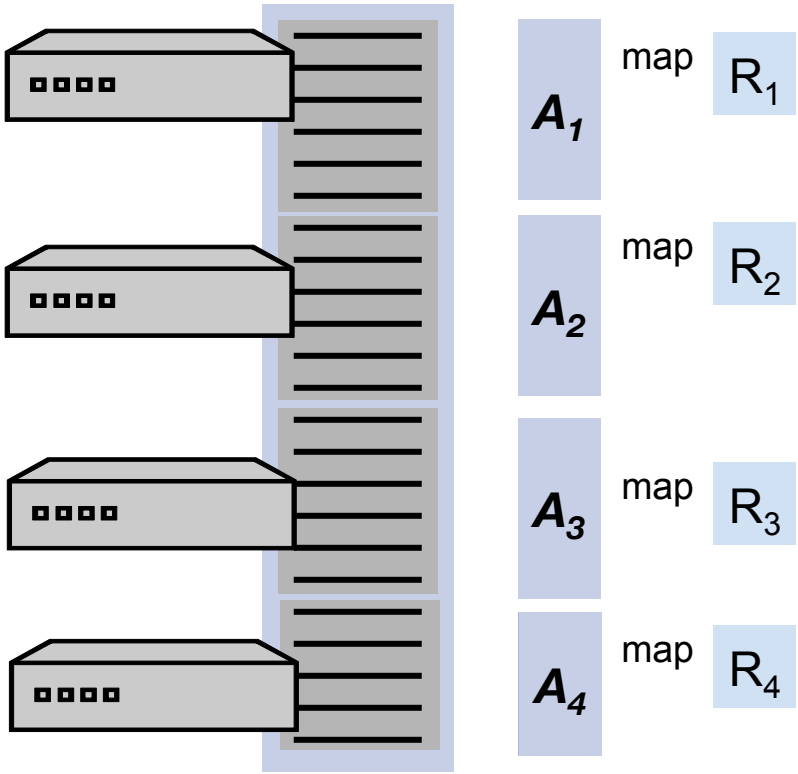


How to store and process tall-and-skinny matrices in MapReduce

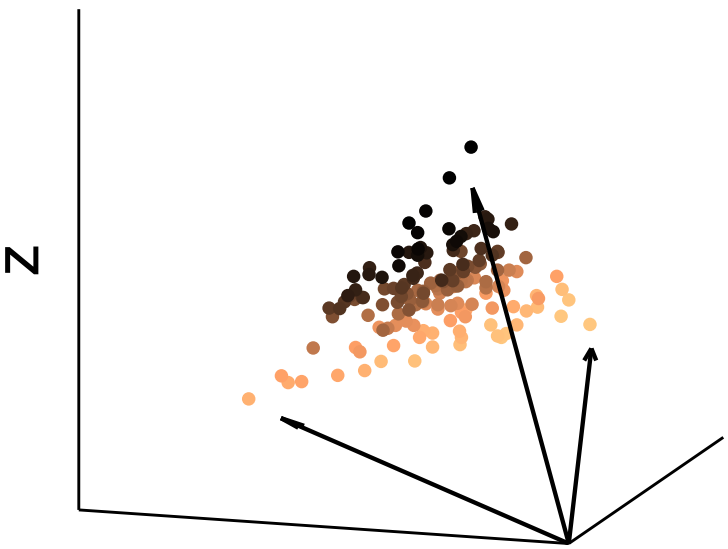
$\boldsymbol{A}$: m x n, m $\gg$ n

Key is an arbitrary row-id
Value is the 1 x n array for a row (or b x n block)

Each submatrix $\boldsymbol{A}_i$ is an the input to a map task.



Separable NMF algorithms are really geometric



Separable NMF

$$\text{Find } \mathbf{H} \geq 0, \mathbf{A}(:, \mathcal{K})$$
$$\text{where } \mathbf{A} \approx \mathbf{A}(:, \mathcal{K})\mathbf{H}$$

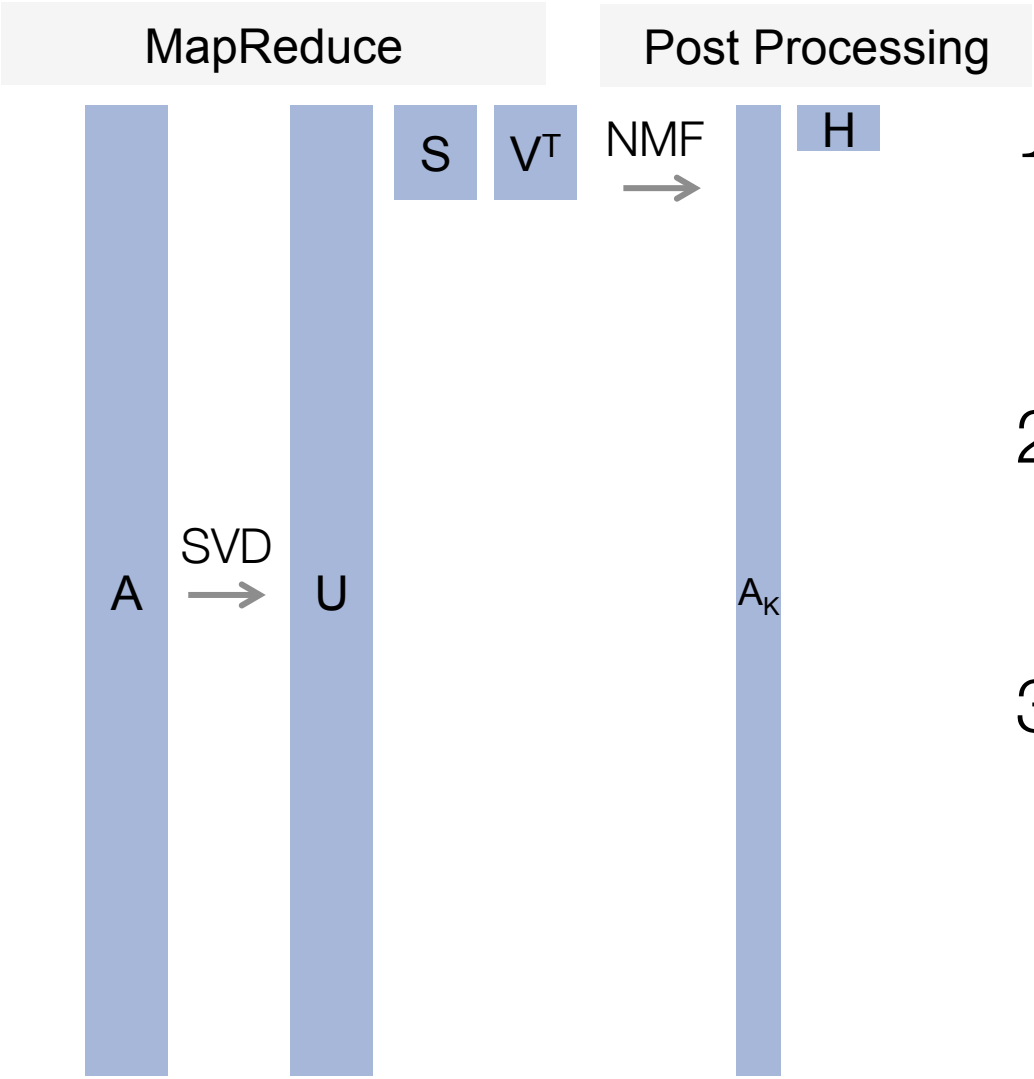
1. Find the columns of A. Equiv. to "Find the extreme points of a convex set."

   These columns are preserved under linear transformations!

2. Then find H

   (local non-neg least squares)

Benson et al. NIPS 2014



Sidenote Easy to use any hierarchical memory system here.

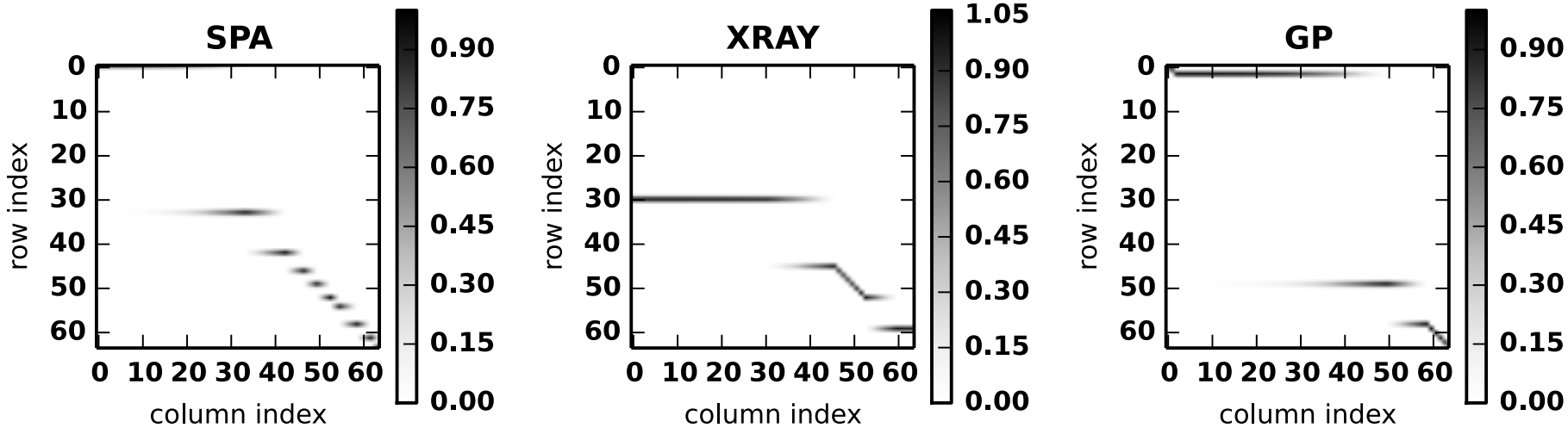
1. Compute QR/SVD using TSQR method
2. Run a separable NMF method on SV^T
3. Find H by solving a small non-negative least-squares problem in each column. These are tiny.

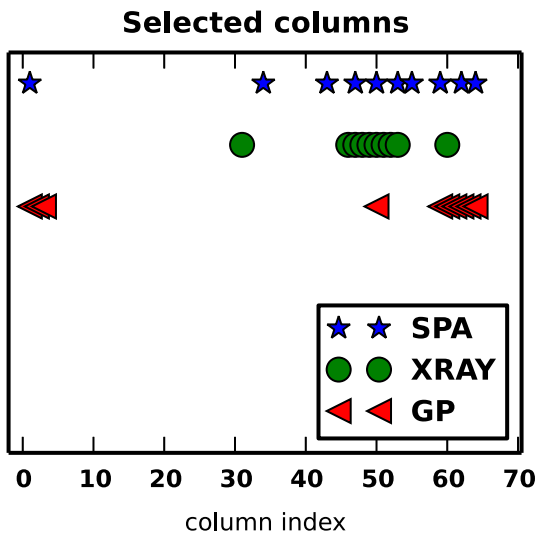
Benson et al. NIPS 2014
Benson et al. BigData 2013

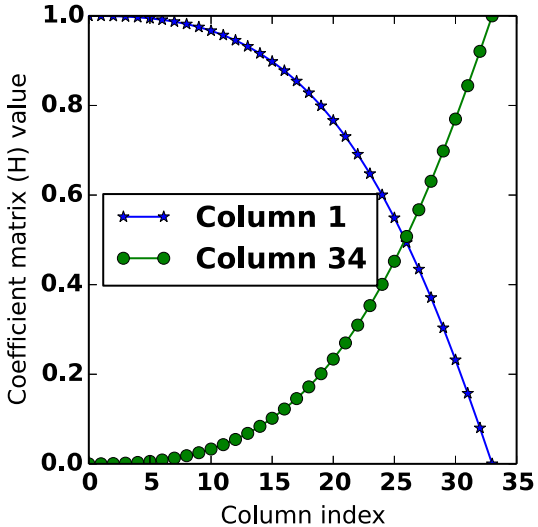
All of the hard analysis is on the small dimension of the matrix, which makes this very useful in practice and effectively “linear-time”

Benson et al. NIPS 2014

Rank 10 Sep NMF



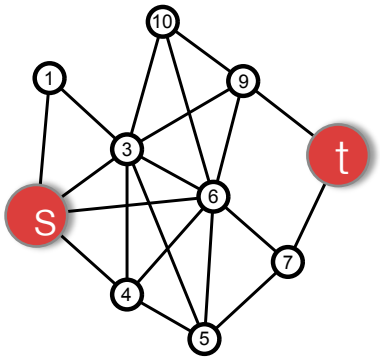




All other simulations are convex combinations of picked simulations!

Benson et al. NIPS 2014

Some algorithms on graphs are hard, even for modestly sized graphs.



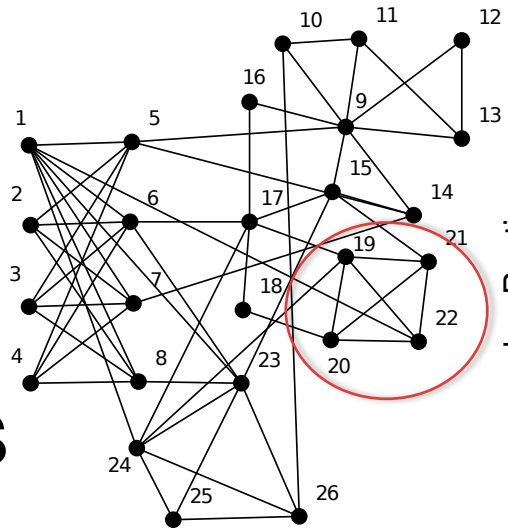
$O(|\text{data}|^{(3/2)})$

Push-relabel

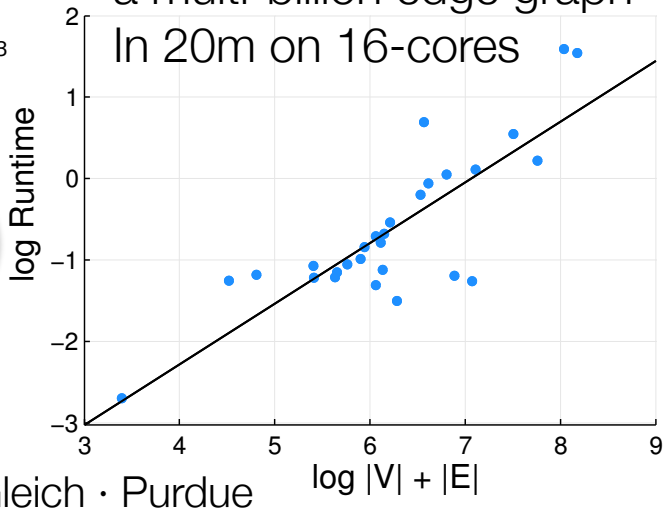
Nearly-linear time stuff soon?

Sidenote 3-loop GEMM, LINPACK are $O(|\text{data}|^{3/2})$ too!

Some hard problems are easy on real-world instances

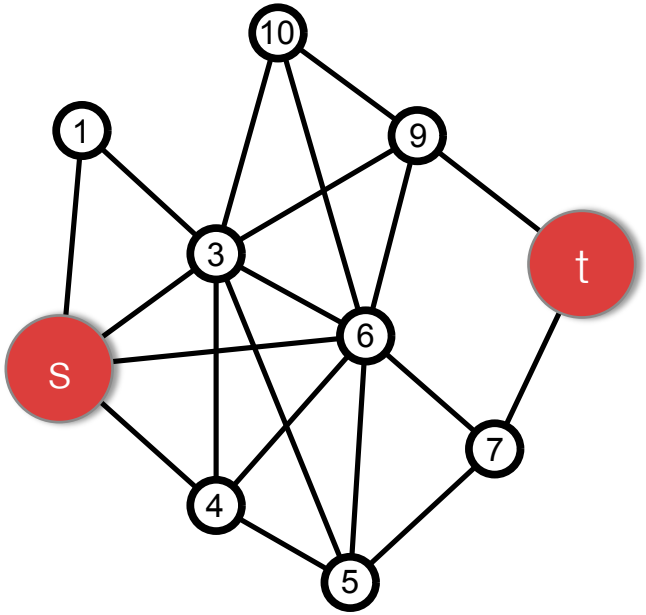


Find the largest clique on a multi-billion edge graph In 20m on 16-cores



Rossi et al. SISC. To appear.

Fast min-cut solvers for large graphs



minimize $\sum_{ij \in E} C_{i,j} |x_i - x_j|$

subject to $x_s = 1, x_t = 0.$

MINCUT LP

s,t-mincut

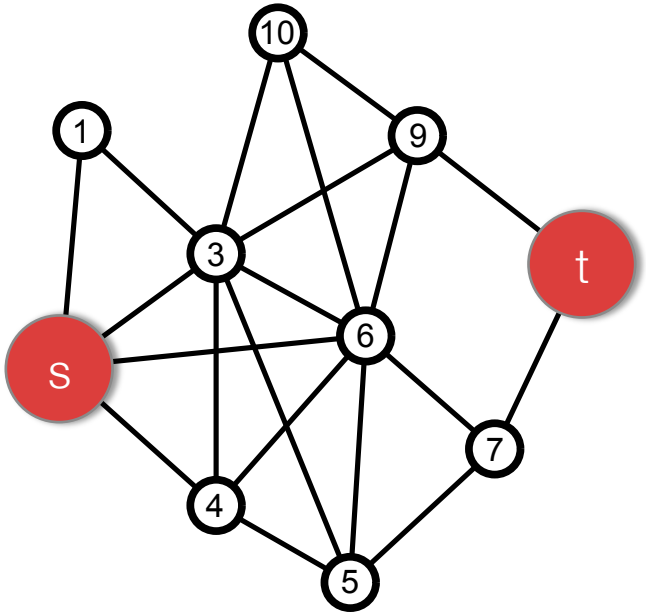
Given an undirected graph, two marked nodes, find the best cut separating them.

- Separating water-from-fat in MRI
- Improving graph partitions
- Object extraction, image restoration, etc.

Often have a sequence of probs

Zhu and Gleich, arXiv: 1501.03105

Fast min-cut solvers for large graphs



s,t-mincut

Given an undirected graph, two marked nodes, find the best cut separating them.

- Best algorithms use max-flow relationships and are serially bound.

$$\begin{array}{ll} \text{minimize} & \sum_{ij \in E} C_{i,j} |x_i - x_j| \\ \text{subject to} & x_s = 1, x_t = 0. \end{array}$$

MINCUT LP

Zhu and Gleich, arXiv: 1501.03105

Mincut to sequence of Laplacian linear systems

$$\begin{aligned} \text{minimize} \quad & \|\boldsymbol{CBx}\|_1 \\ \text{subject to} \quad & x_s = 1, x_t = 0 \end{aligned}$$

Variational representation of 1-norm

$$\|\mathbf{z}\|_1 = \min_{\mathbf{w} \geq 0} \frac{1}{2} \sum_i \left(\frac{z_i^2}{w_i} + w_i \right)$$

$$\begin{aligned} \underset{\mathbf{x},\mathbf{w}}{\text{minimize}} \quad & \sum_{i,j \in E} \frac{1}{2} \left(\frac{C_{i,j}^2 (x_i - x_j)^2}{w_{i,j}} + w_{i,j} \right) \\ \text{subject to} \quad & \mathbf{w} \geq 0, x_s = 1, x_t = 0 \end{aligned}$$

Jointly convex optimization over

- The solution x
- An edge-weight vector w

Now, use an alternating algorithm over x and w

Zhu and Gleich, arXiv: 1501.03105

Mincut to sequence of Laplacian linear systems

1. Compute $w_i^{(k)} = \sqrt{C_{i,j}^2 (x_i^{(k-1)} - x_j^{(k-1)})^2 + \varepsilon^2}$
2. Solve $\text{minimize}_{\mathbf{x}} \ \frac{1}{2}\mathbf{x}^T \boldsymbol{B}^T \boldsymbol{C}(\boldsymbol{W}^{(k)})^{-1} \boldsymbol{C}\boldsymbol{B}\mathbf{x}$ with $x_s = 1, x_t = 0$

The hard step! But this is a Laplacian lin. sys.

$$\begin{aligned} \underset{\mathbf{x},\mathbf{w}}{\text{minimize}} \quad & \sum_{i,j \in E} \frac{1}{2} \left(\frac{C_{i,j}^2 (x_i - x_j)^2}{w_{i,j}} + w_{i,j} \right) \\ \text{subject to} \quad & \mathbf{w} \geq 0, x_s = 1, x_t = 0 \end{aligned}$$

Jointly convex optimization over

- The solution x
- An edge-weight vector w

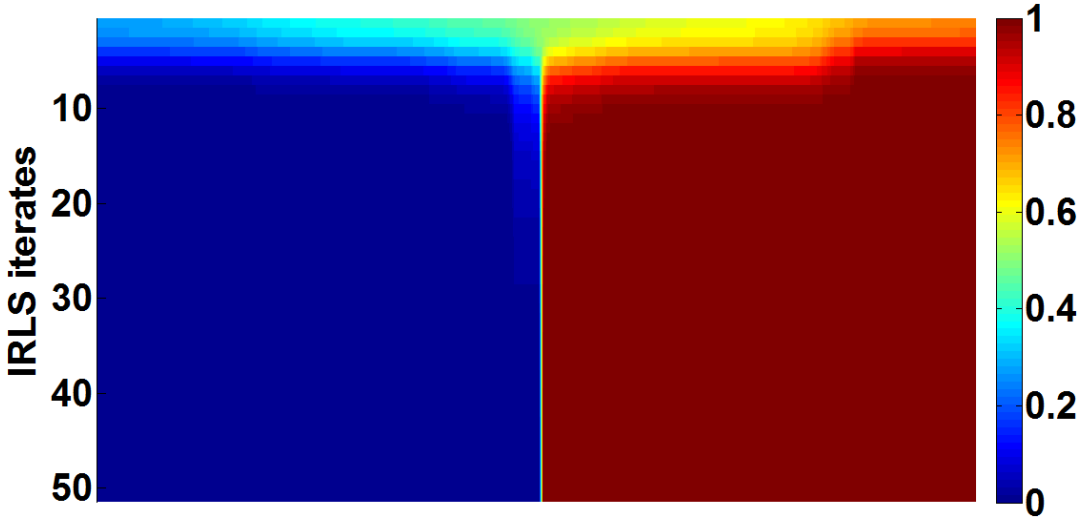
Now, use an alternating algorithm over x and w

Zhu and Gleich, arXiv: 1501.03105

Solving the big linear system

The algorithm is now a sequence of big linear systems which are weighted Laplacians.

- Use PETSc / and Block Jacobi preconditioned CG to solve the Laplacians
- Exploit “discrete” info in the continuous solution to round faster.
- Convergence theory too.



Zhu and Gleich, arXiv: 1501.03105

Performance on benchmark graphs

System 8 nodes,
24 cores (E5-4617)

	nodes	edges
asia_osm	11,950,757	12,711,603
euro_osm	50,912,018	54,054,660
adhead.n26c100	12,582,912	163,577,856
bone.n26c100	7,798,784	101,384,192
liver.n26c100	4,161,600	54,100,800

Graph	PIRMCut	B-K Solver	Speedup	Number of Cores
asia_osm	49	1465	29.8	128
euro_osm	301	3102	10.3	128
adhead.n26c100	174	555	3.2	64
bone.n26c100	101	215	2.1	64
liver.n26c100	95	294	3.1	64

Zhu and Gleich, arXiv: 1501.03105

Challenges for the next 5 years!

1. Extend “e-linear-time” matrix analysis to thick matrices (100k + columns).
2. Parallelize more “hard” graph algorithms.
3. New compression for reducing IO bottlenecks (not new)
4. Take advantage of properties of real-world computational problems, e.g. low-rank objective structure, real-world graph props for faster algorithms
5. Understand what we should compute on large graphs in light of increased noise/data
6. Statistical significance of graph features/patterns
7. Algorithmic resilience without system overhead.
8. Quad precision for accurate big-data matrix computations.

PageRank on the Twitter graph on a Raspberry Pi

Raspberry Pi 4 core ARM CPU, 1GB RAM

Twitter graph 40m nodes, 1.4B edges. 6GB on disk

PageRank iteration with USB on-disk graph

4m 14s (25 MB/sec) = 72% of peak USB speed

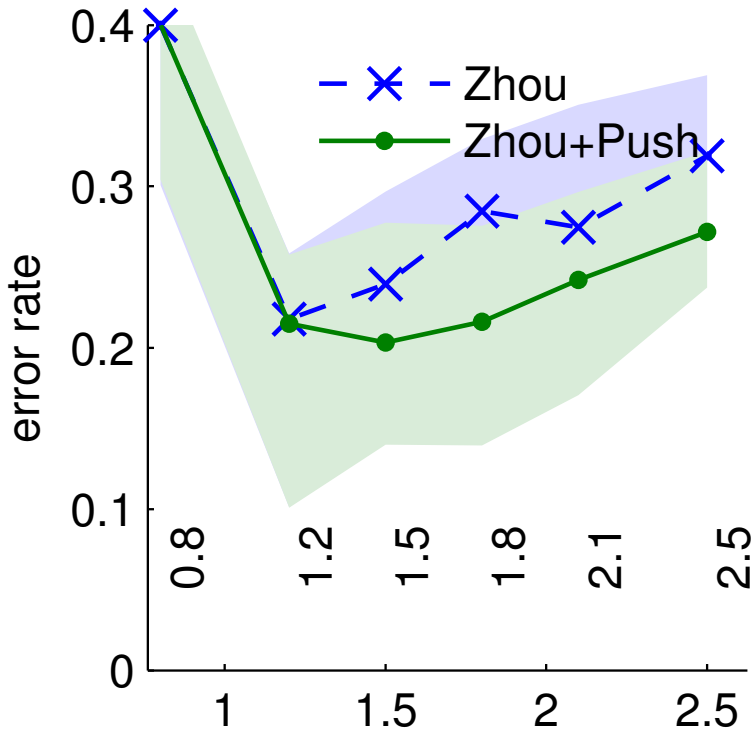
Streaming the graph alone 3m 30s (82% of peak)

2% of energy of HPC solutions (Big Data frameworks).

Goal 2 min per iteration with compression (142% of peak)

Rainey and Gleich, In progress

Using a local algorithm implicitly regularizes a mathematical model



Zhou is an "exact" algorithm involving solving a linear system.

Zhou + Push is a fast approximation. It outperforms Zhou due to implicit regularization properties.

This is a win-win! Faster algorithm, better performance!

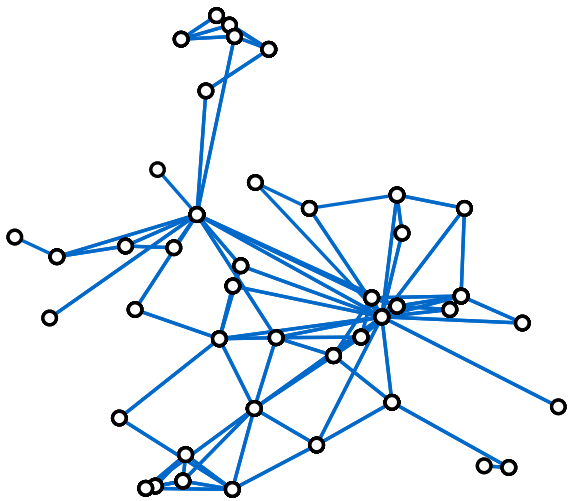
$$\text{minimize}^{\sigma} \quad \boxed{\sum_{ij \in E} C_{i,j} |x_i - x_j|^2} + \boxed{\tau \sum_{i \in V} d_i x_i}$$

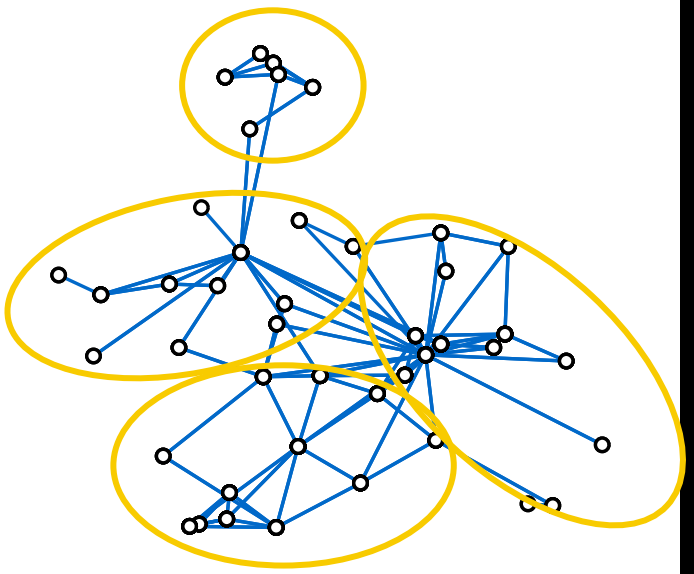
$$\text{subject to} \quad x_s = 1, x_t = 0, x_i \geq 0.$$

Gleich and Mahoney, KDD 2015

The graph-based data analysis pipeline

1	0	0	0	1	0	0	1	
0	1	0	1	0	0	1	1	
0	1	0	1	0	0	0	1	
1	0	0	0	0	0	1	1	
1	1	0	1	1	1	0	1	
1	0	1	1	0	0	0	1	
1	0	1	1	1	0	1	0	
1	1	1	1	0	1	0	0	
1	1	1	0	0	1	1	1	
1	1	0	1	1	1	1	1	





Raw data

- Relationships
- Images
- Text records
- Etc.

Convert to a graph

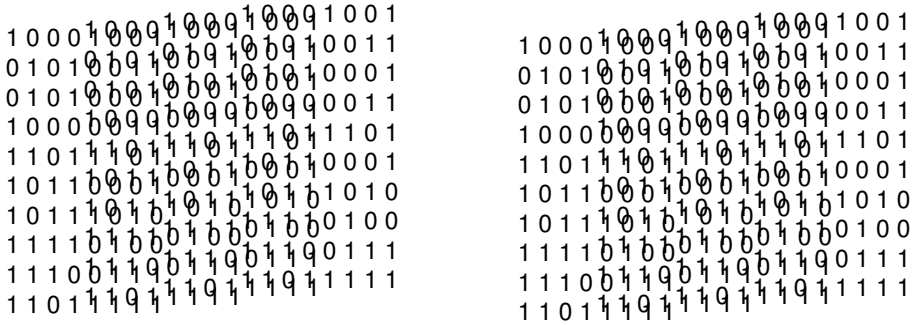
- Nearest neighs
- Kernels
- 2-mode to 1-mode
- Etc.

Algorithm/Learning

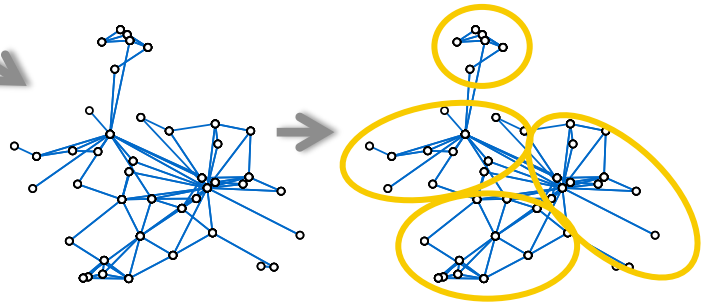
- Important nodes
- Infer features
- Clustering
- Etc.

Gleich and Mahoney, KDD 2015

Graph-based learning is usually only one component of a big pipeline



Many databases over genes with survival rates for various cancers



THIS STEP SHOULD BE ROBUST TO VARIATIONS ABOVE

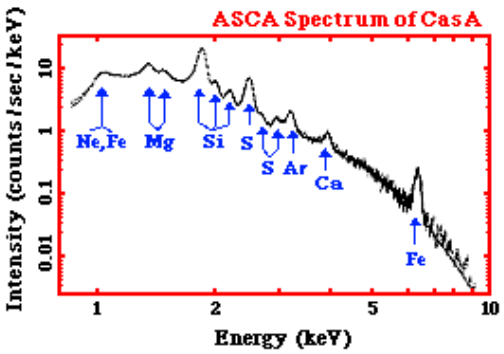
Cluster analysis

Reinteration of data

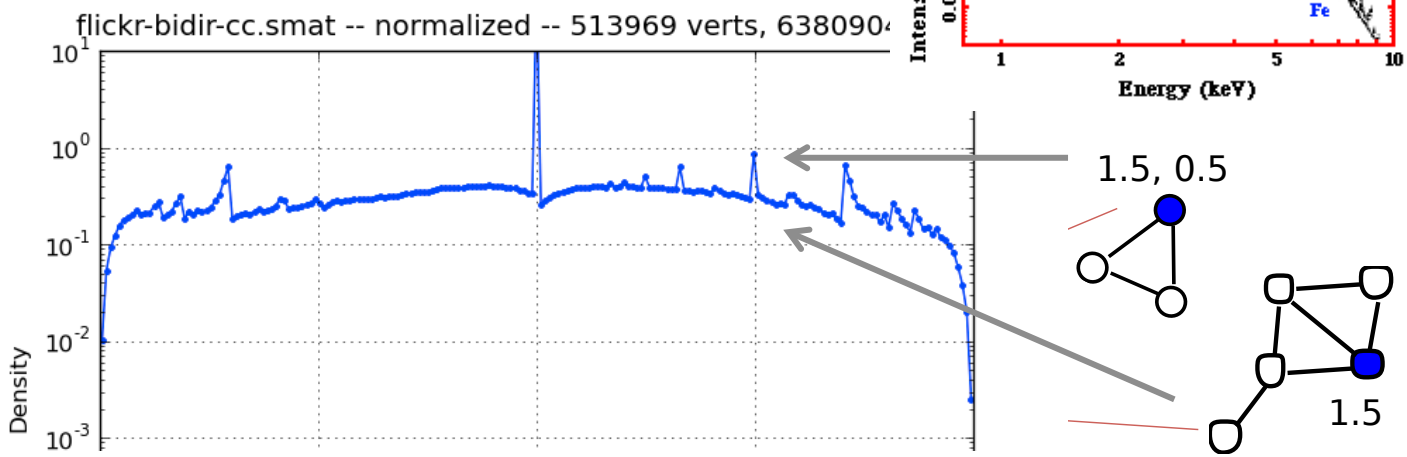
List of possible genes responsible for survival

Gleich and Mahoney, KDD 2015

I found something in a graph. Is it significant?



Eigenvalue spectrum of a 500k node graph, 7400 cores (925 nodes) RedSky for 10 hours



What is a prior for a graph?

- Erdős–Rényi
- Weighted (skewed) Erdős–Rényi
- Kronecker (R-mat)

Will result in huge computational issues

We can develop computational analogs of erasure coding (RAID)!

Raw system	$\boldsymbol{A}\mathbf{x} = \mathbf{b}$	$n \times n$
Encoded system	$\tilde{\boldsymbol{A}}\tilde{\mathbf{x}} = \tilde{\mathbf{b}}$	$(n+k) \times (n+k)$

We can build $\tilde{\mathbf{A}}$ with only a small amount of reliable work

If we run CG on the encoded system, then we can ignore up to k faults.

We can recover $\mathbf{x}$ from $\tilde{\mathbf{x}}$ with k arbitrary components with a small amount of reliable work.

Does this work for other problems? Real-world feasible?

Zhu, Gleich, Grama arXiv:1412.7364

References

dgleich@purdue.edu

www.cs.purdue.edu/homes/dgleich (Google Gleich)

Constantine & Gleich, MapReduce 2011 (my website)

Benson, Gleich, & Demmel, BigData '13 arXiv:1301.1071

Benson, Lee, Gleich, Rawja, NIPS '14 arXiv:1402.6964

Rossi, Gleich, Gebremedhin, SISC To Appear arXiv:1302.6256

Zhu & Gleich, Submitted arXiv:1501.03105

Zhu, Gleich, & Grama, Submitted arXiv:1412.7364

Gleich & Mahoney, ICML '14 (my website)

Gleich & Machine, KDD '15 (my website)

Sidenote Two students graduating within 1 year.

Supported by NSF, DOE

CODE www.cs.purdue.edu/homes/dgleich/codes/robust-diffusions