



Facebook Friends and Matrix Functions

Graduate Research Day



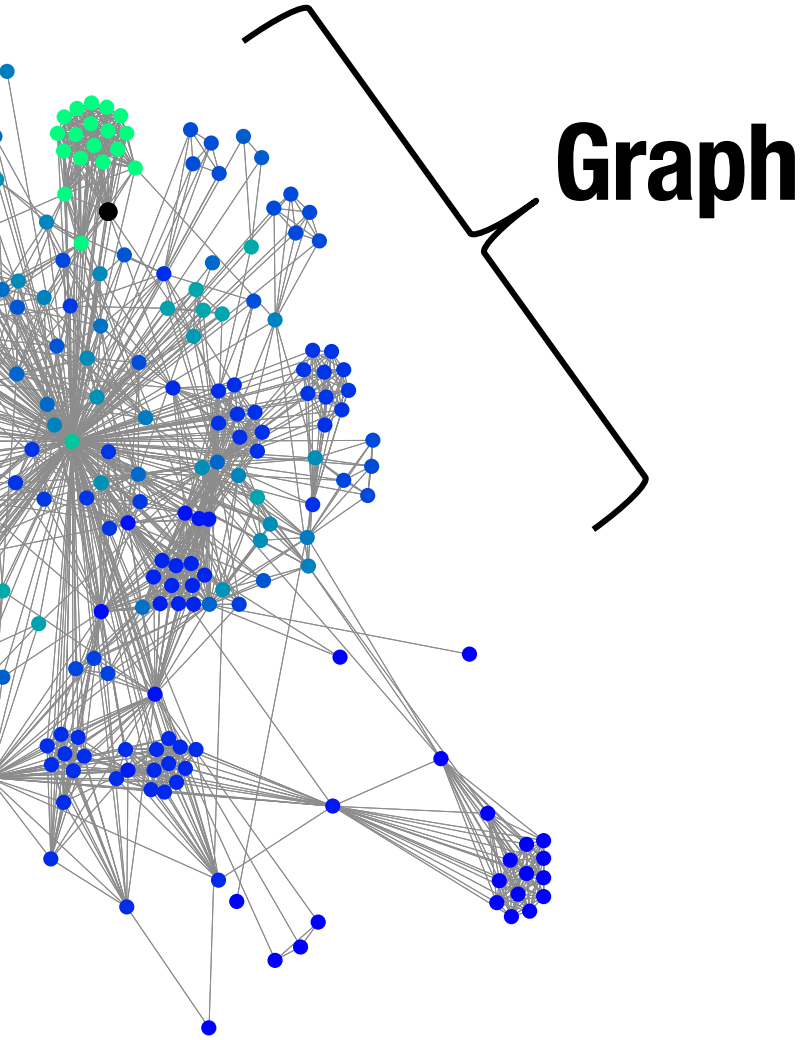
Joint with
David F. Gleich,
(Purdue), supported by
NSF CAREER
1149756-CCF

PURDUE
UNIVERSITY

Kyle Kloster
Purdue University

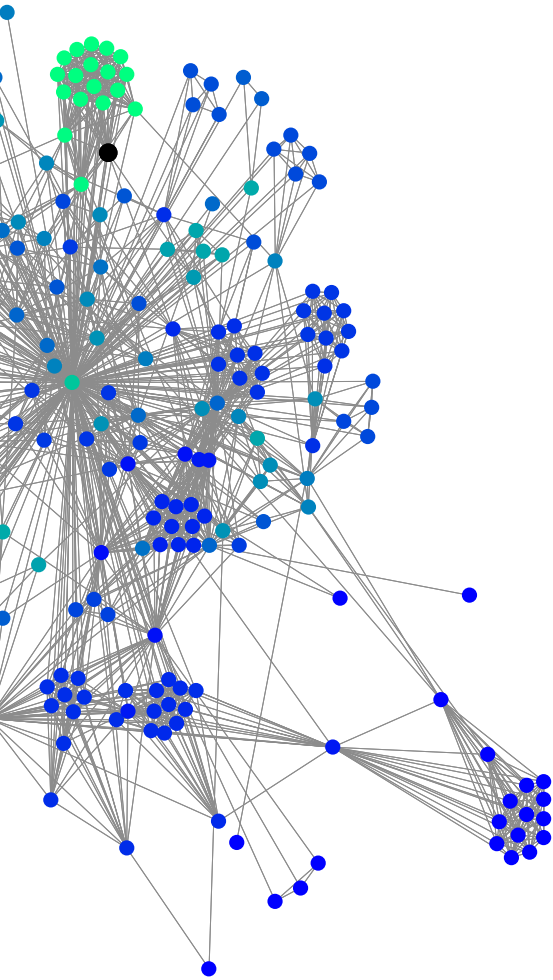
Network Analysis

Use linear algebra to study graphs



Network Analysis

Use linear algebra to study graphs



Graph, G

V , vertices (nodes)

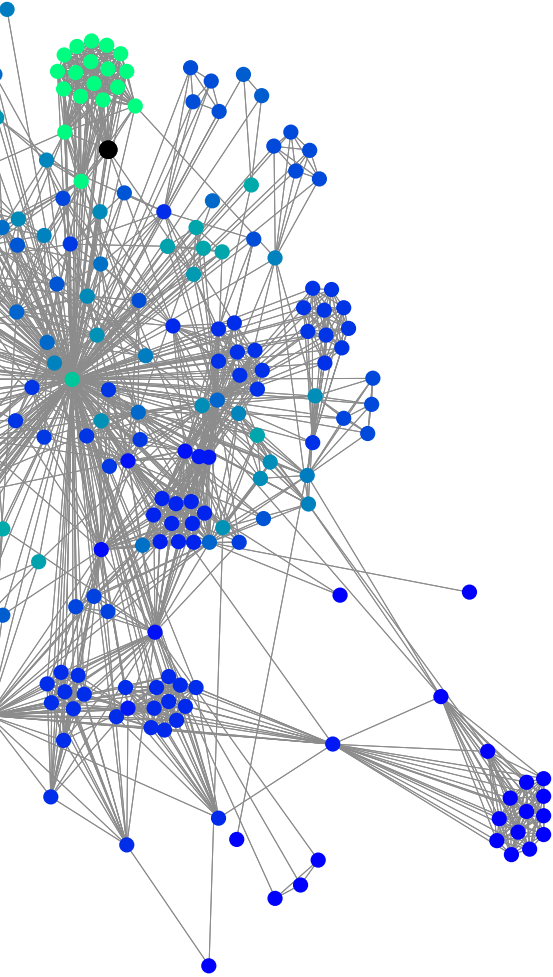
E , edges (links)

degree of a node =
edges incident to it.

nodes sharing an
edge are **neighbors**.

Network Analysis

Use linear algebra to study graphs



Graph, G

V , vertices (nodes)

E , edges (links)

Erdős Number

Facebook friends

Twitter followers

Search engines

Amazon/Netflix rec.

Protein interactions

Power grids

Google Maps

Air traffic control

Sports rankings

Cell tower placement

Scheduling

Parallel programming

Everything

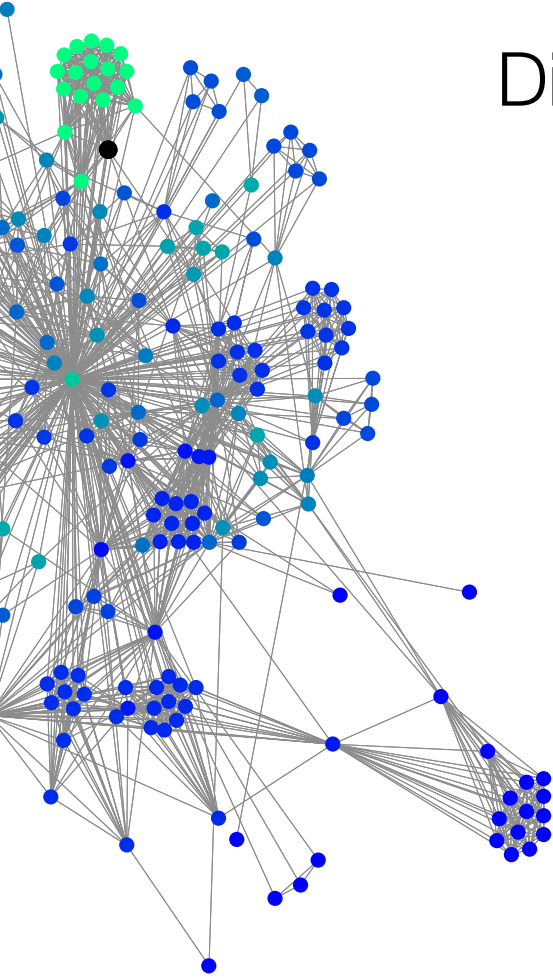
Kevin Bacon

Network Analysis

Use linear algebra to study graphs

Graph Properties

Diameter



Is everything just
a few hops away
from everything else?

Network Analysis

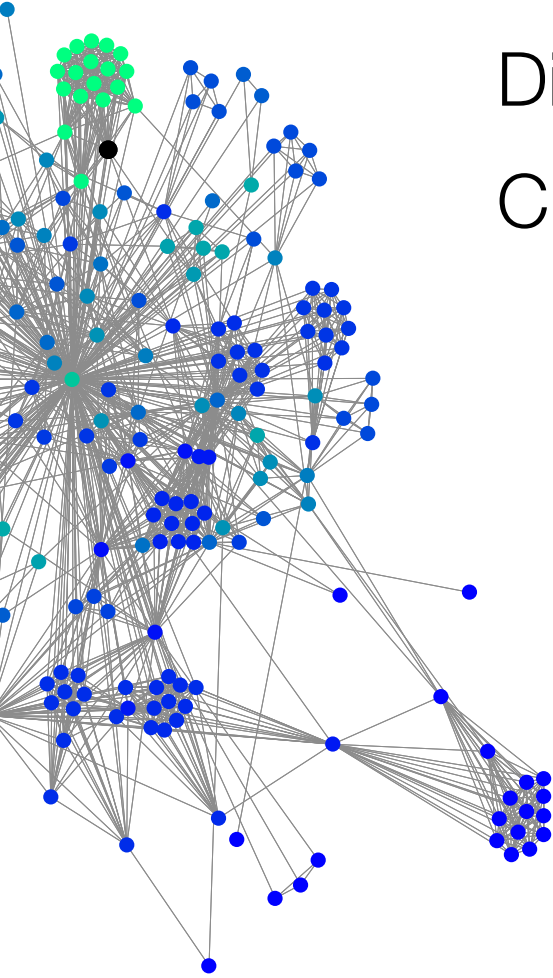
Use linear algebra to study graphs

Graph Properties

Diameter

Clustering

Are there tightly-knit
groups of nodes?



Network Analysis

Use linear algebra to study graphs

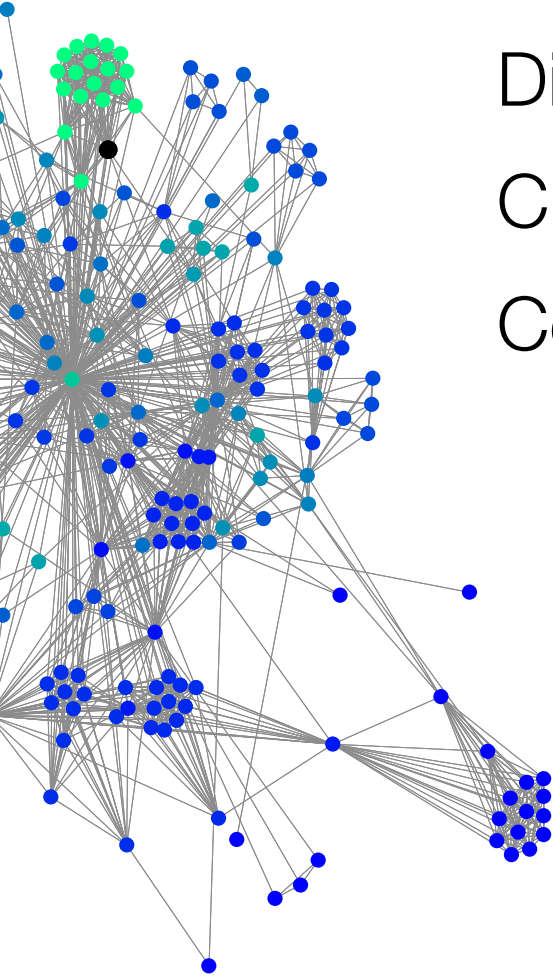
Graph Properties

Diameter

Clustering

Connectivity

How well can each
node reach every
other node?



Network Analysis

Use linear algebra to study graphs

Graph Properties

Diameter

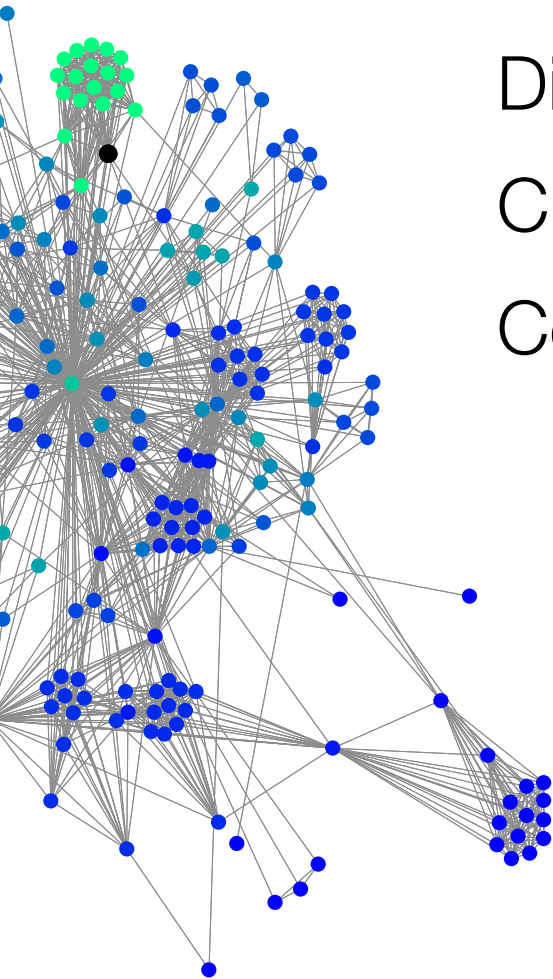
Clustering

Connectivity

Linear Algebra

Eigenvalues and matrix functions shed light on all these questions.

These tools require a matrix related to the graph...



Graph Matrices

Adjacency matrix, **A**

$$\mathbf{A}_{ij} = \begin{cases} 1, & \text{if nodes } i, j \text{ share an edge (are adjacent)} \\ 0 & \text{otherwise} \end{cases}$$

Random-walk transition matrix, **P**

$$\mathbf{P}_{ij} = \mathbf{A}_{ij} / d_j \quad \text{where } d_j \text{ is the degree of node } j.$$

Stochastic! i.e. column-sums = 1

Network analysis via **Heat Kernel**

Uses include

- Local clustering
- Link prediction
- Node centrality

Heat kernel is...

a graph diffusion
a function of a matrix

$$\exp(\mathbf{G}) = \sum_{k=0}^{\infty} \frac{1}{k!} \mathbf{G}^k$$

For \mathbf{G} , a network's

random-walk,	\mathbf{P}
adjacency,	\mathbf{A}
Laplacian,	\mathbf{L}

matrix

Heat Kernel describes node connectivity

$(\mathbf{A}^k)_{ij} = \#$ walks of length k from node i to j

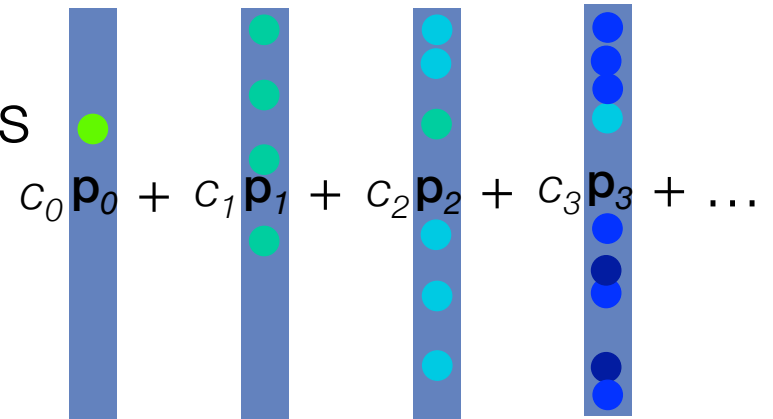
$$\exp(\mathbf{A})_{ij} = \sum_{k=0}^{\infty} \frac{1}{k!} (\mathbf{A}^k)_{ij}$$

“sum up” the walks between i and j

For a small set of seed nodes, \mathbf{s} , $\exp(\mathbf{A}) \mathbf{s}$ describes nodes most relevant to \mathbf{s}

Diffusion score

“diffusion scores” of a graph =
weighted sum of probability vectors



diffusion score vector = \mathbf{f}

$$\mathbf{f} = \sum_{k=0}^{\infty} c_k \mathbf{P}^k \mathbf{s}$$

\mathbf{P} = random-walk
transition matrix

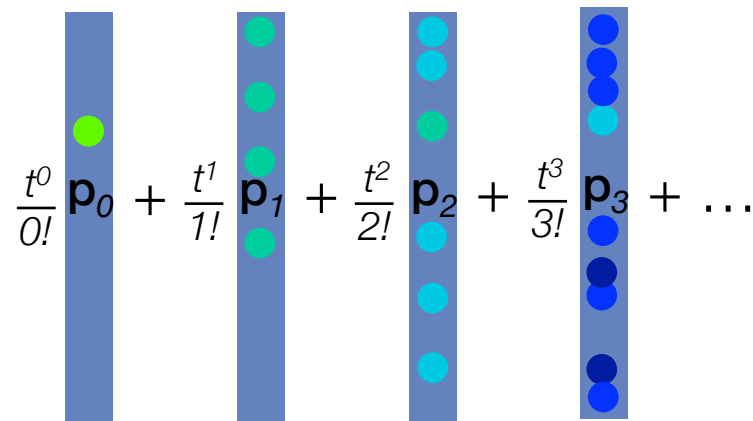
\mathbf{s} = normalized
seed vector

c_k = weight on
stage k

Heat Kernel vs. PageRank Diffusions

Heat Kernel uses $t^k/k!$

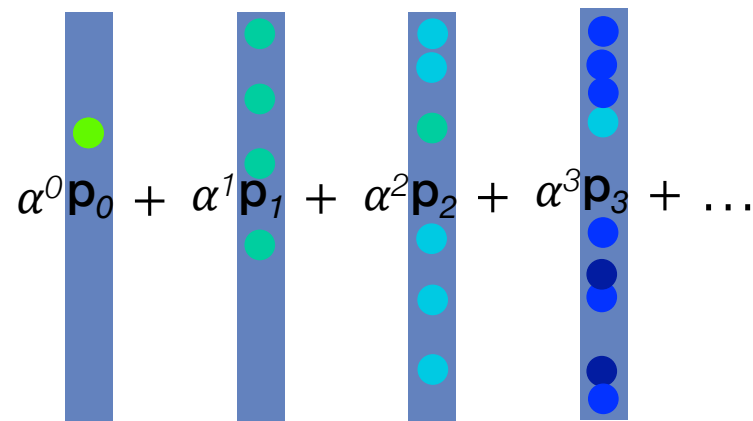
Our work is new analysis and algorithms for this diffusion.

$$\frac{t^0}{0!} \mathbf{p}_0 + \frac{t^1}{1!} \mathbf{p}_1 + \frac{t^2}{2!} \mathbf{p}_2 + \frac{t^3}{3!} \mathbf{p}_3 + \dots$$


PageRank uses α^k at stage k.

Standard, widely-used diffusion we use for comparison.

Linchpin of Google's original success!

$$\alpha^0 \mathbf{p}_0 + \alpha^1 \mathbf{p}_1 + \alpha^2 \mathbf{p}_2 + \alpha^3 \mathbf{p}_3 + \dots$$


Heat Kernel vs. PageRank Theory

good
clusters

fast
algorithm

PR

Local Cheeger Inequality:
“PR finds near-optimal
clusters”

existing constant-time
algorithm
[Andersen Chung Lang 06]

HK

Heat Kernel vs. PageRank Theory

good
clusters

fast
algorithm

PR

Local Cheeger Inequality:
“PR finds near-optimal
clusters”

existing constant-time
algorithm
[Andersen Chung Lang 06]

HK

Local Cheeger Inequality
[Chung 07]

Heat Kernel vs. PageRank Theory

good
clusters

fast
algorithm

PR

Local Cheeger Inequality:
“PR finds near-optimal
clusters”

existing constant-time
algorithm
[Andersen Chung Lang 06]

HK

Local Cheeger Inequality
[Chung 07]

Our work

Algorithm outline

$$\hat{\mathbf{x}} \approx \exp(\mathbf{P}) \mathbf{s}$$

- (1) Approximate with a polynomial
- (2) Convert to linear system (Details in paper)
- (3) Solve with sparse linear solver

Algorithm outline

$$\hat{\mathbf{x}} \approx \exp(\mathbf{P}) \mathbf{s}$$

- (1) Approximate with a polynomial
- (2) Convert to linear system (Details in paper)
- (3) Solve with sparse linear solver

$$\mathbf{A}\mathbf{x}^{(k)} \approx \mathbf{b}$$

**Gauss-Southwell
Sparse solver**

$$\mathbf{r}^{(k)} := \mathbf{b} - \mathbf{A}\mathbf{x}^{(k)}$$

$$\mathbf{x}^{(k+1)} := \mathbf{x}^{(k)} + \mathbf{A}\mathbf{r}_{big}^{(k)}$$


“relax” largest
entry in \mathbf{r}

Algorithm outline

$$\hat{\mathbf{x}} \approx \exp(\mathbf{P}) \mathbf{s}$$

- (1) Approximate with a polynomial
- (2) Convert to linear system (Details in paper)
- (3) Solve with sparse linear solver

Key: We avoid doing these full matrix-vector products

$$\exp(\mathbf{P}) \mathbf{s} \approx \sum_{k=0}^N \frac{1}{k!} \mathbf{P}^k \mathbf{s}$$


Algorithm outline

$$\hat{\mathbf{x}} \approx \exp(\mathbf{P}) \mathbf{s}$$

(1) Approximate with a polynomial

(2) Convert to linear system

(Details in paper)

(3) Solve with sparse linear solver

Key: We avoid doing these full matrix-vector products

$$\exp(\mathbf{P}) \mathbf{s} \approx \sum_{k=0}^N \frac{1}{k!} \mathbf{P}^k \mathbf{s}$$

(All my work was showing this actually can be done with bounded error.)

Algorithms & Theory for $\hat{\mathbf{x}} \approx \exp(\mathbf{P}) \mathbf{s}$

Algorithm 1, Weak Convergence

- *constant time* on any graph, $\tilde{O}\left(\frac{e^1}{\varepsilon}\right)$
- outperforms [PageRank](#) in clustering
- accuracy: $\|\mathbf{D}^{-1} \mathbf{x} - \mathbf{D}^{-1} \hat{\mathbf{x}}\|_{\infty} < \varepsilon$

Algorithms & Theory for $\hat{\mathbf{x}} \approx \exp(\mathbf{P}) \mathbf{s}$

$$\|\mathbf{D}^{-1} \mathbf{x} - \mathbf{D}^{-1} \hat{\mathbf{x}}\|_{\infty} < \varepsilon$$

Conceptually

Diffusion vector quantifies node's connection to each other node. Divide each node's score by its degree, delete the nodes with score $< \varepsilon$.

Only a constant number of nodes remain in G !

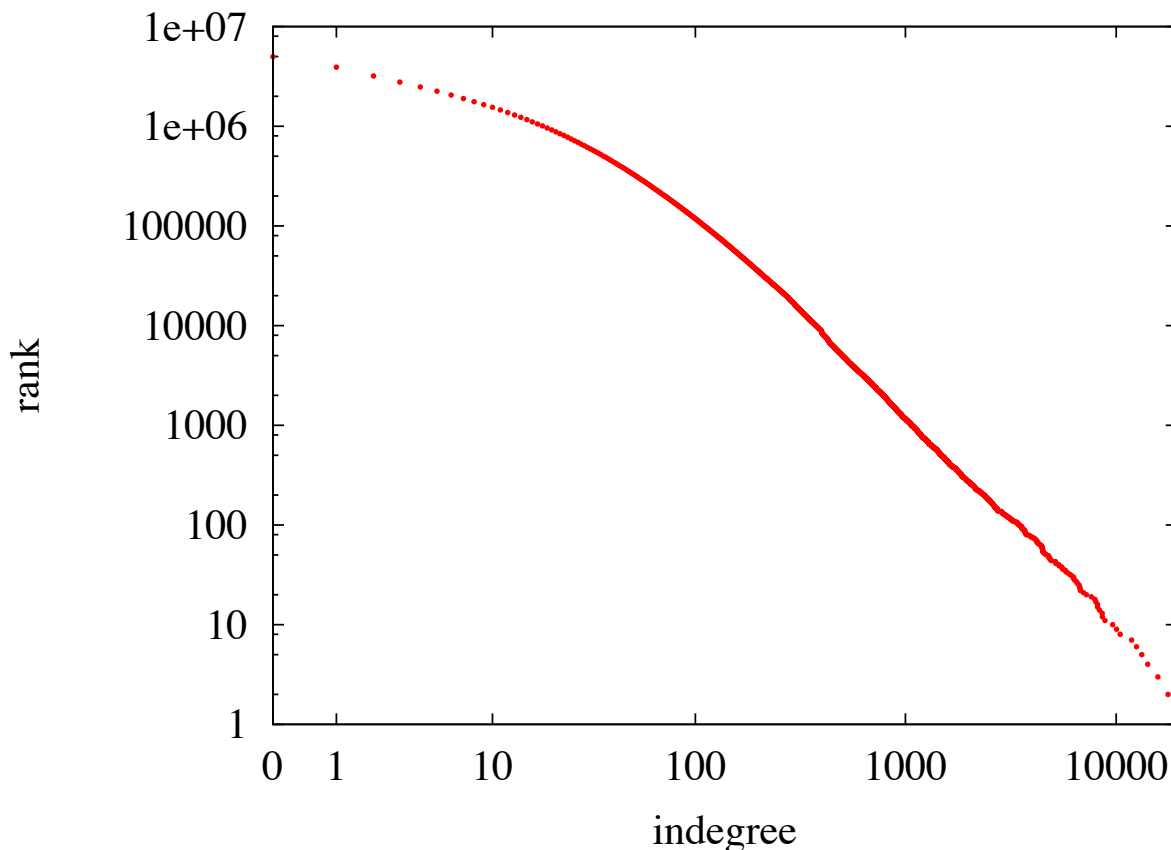
Users spend “reciprocated time” with $O(1)$ others.

Algorithms & Theory for $\hat{\mathbf{x}} \approx \exp(\mathbf{P}) \mathbf{s}$

Algorithm 2, Global Convergence (conditional)

Power-law Degrees

Realworld graphs have degrees distributed as follows. This causes diffusions to be **localized**.



Power-law degrees

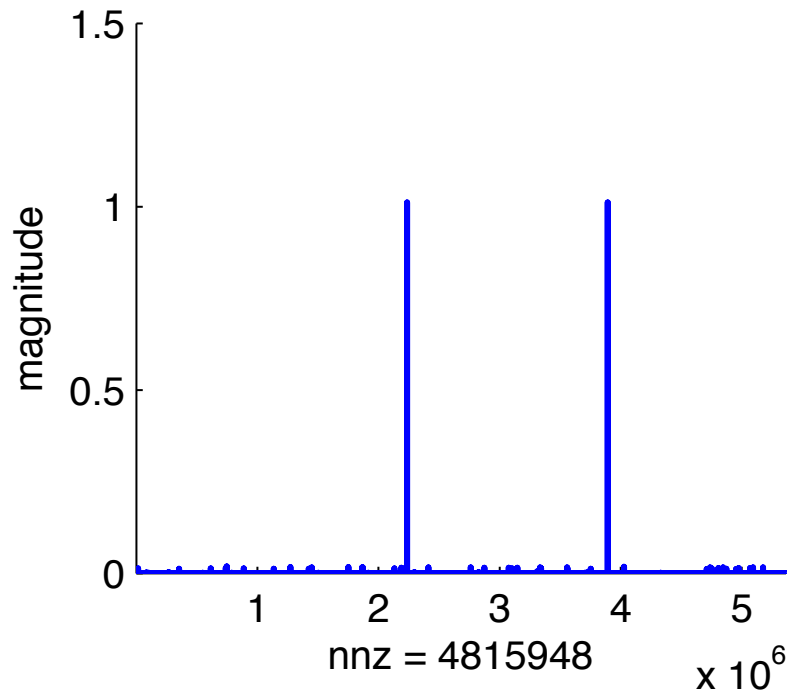
Degrees of nodes
in Ljournal-2008

Log-log scale

[Boldi et al., Laboratory for Web Algorithmics 2008]

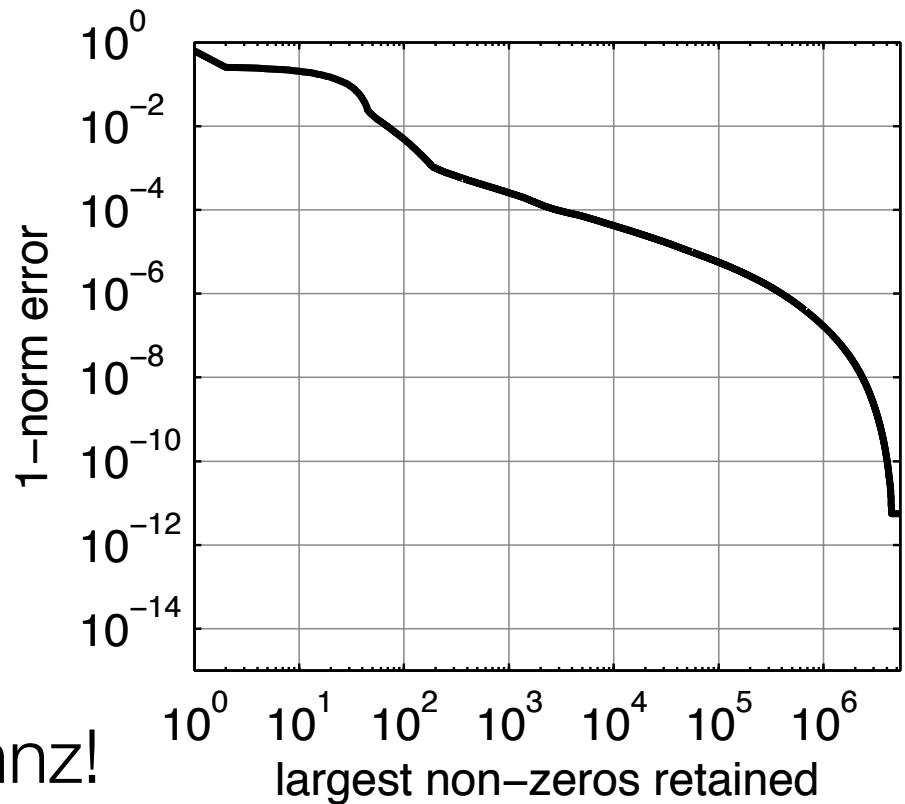
Local solutions

Magnitude of entries in solution vector



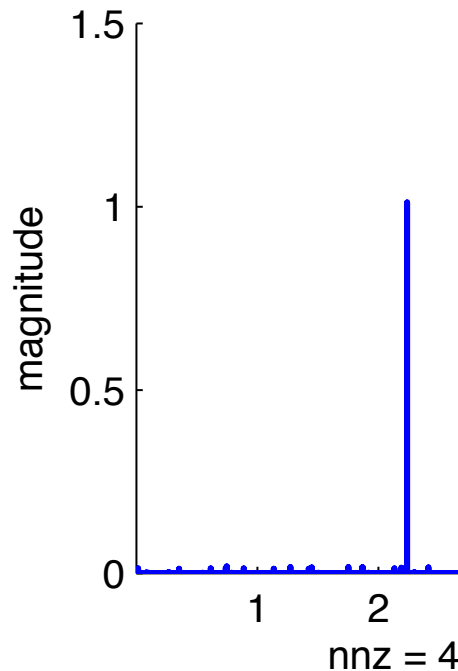
$\sum_{k=0}^{\infty} \frac{1}{k!} \mathbf{A}^k \mathbf{s}$ has ~ 5 million nnz!

Accuracy of approximation using only large entries

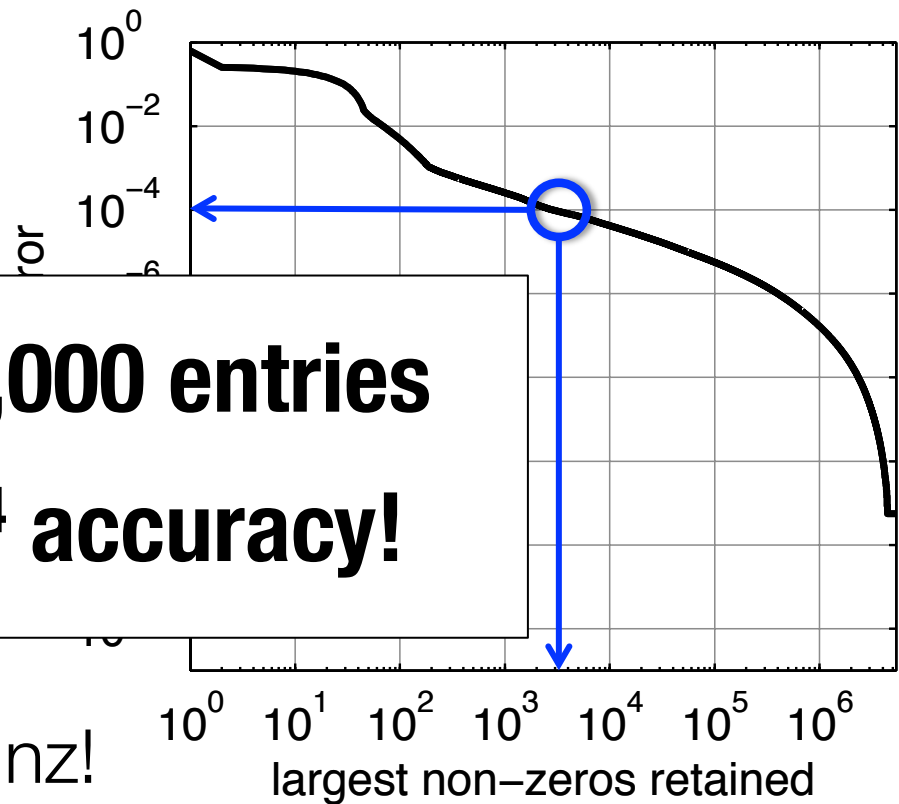


Local solutions

Magnitude of entries in solution vector



Accuracy of approximation using only large entries



**Only ~3,000 entries
For 10^{-4} accuracy!**

$$\sum_{k=0}^{\infty} \frac{1}{k!} \mathbf{A}^k \mathbf{s} \text{ has } \sim 5 \text{ million nnz!}$$

Algorithms & Theory for $\hat{\mathbf{x}} \approx \exp(\mathbf{P}) \mathbf{s}$

Algorithm 2, Global Convergence (conditional)

- *sublinear* (power-law) $\tilde{O}(d \log d (1/\varepsilon)^C)$

- accuracy: $\|\mathbf{x} - \hat{\mathbf{x}}\|_1 < \varepsilon$

Algorithms & Theory for $\hat{\mathbf{x}} \approx \exp(\mathbf{P}) \mathbf{s}$

$$\|\mathbf{x} - \hat{\mathbf{x}}\|_1 < \varepsilon$$

Conceptually

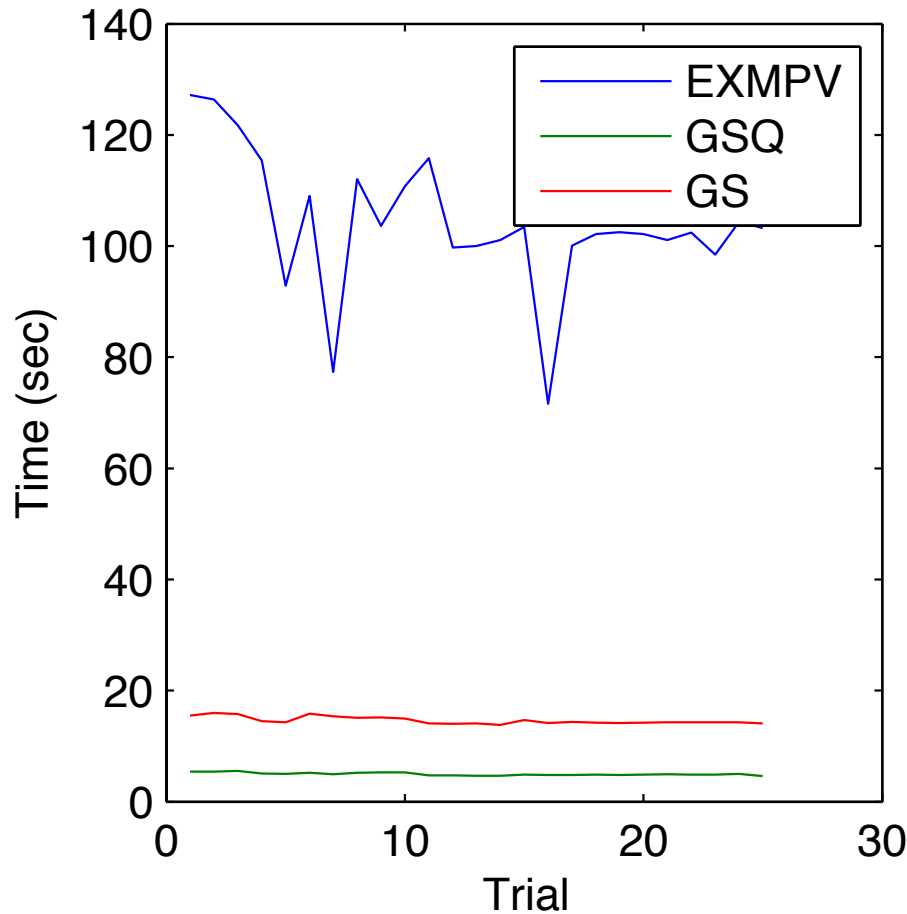
A node's diffusion vector can be approximated with **total** error $< \varepsilon$ using only $O(d \log d)$ entries.

In realworld networks (i.e. with degrees following a power-law), no node will have nontrivial connection with more than $O(d \log d)$ other nodes.

Experiments

Runtime on the web-graph

A particularly sparse graph benefits us best



$$|V| = O(10^8)$$
$$|E| = O(10^9)$$

GSQ, GS: our methods
EXPMV: MatLab

Thank you

Local clustering via heat kernel code available at

`http://www.cs.purdue.edu/homes/dgleich/codes/hkgrow`

Global heat kernel code available at

`http://www.cs.purdue.edu/homes/dgleich/codes/nexpokit/`

Questions or suggestions? Email Kyle Kloster at `kkloste-at-purdue-dot-edu`