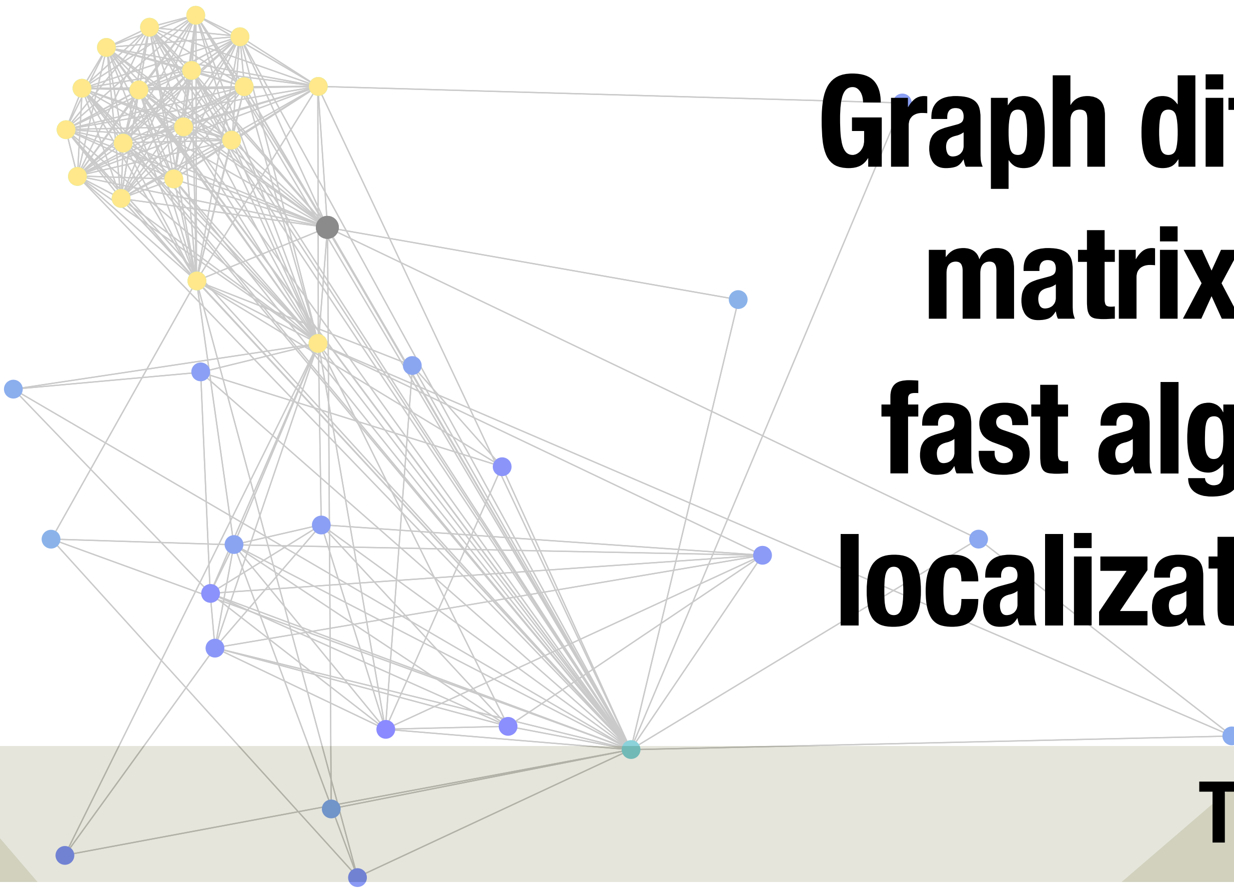


Graph diffusions and matrix functions: fast algorithms and localization results

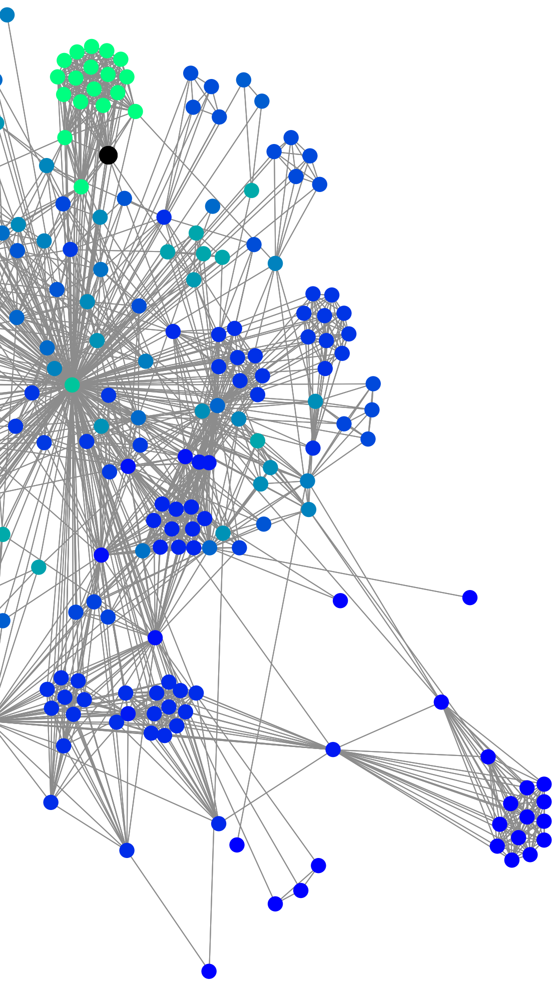
Thesis defense

Advised by
David F. Gleich
Supported by
NSF CAREER
1149756-CCF

Kyle Kloster
Purdue University

Network Analysis

Graphs can model everything!



Graph, G

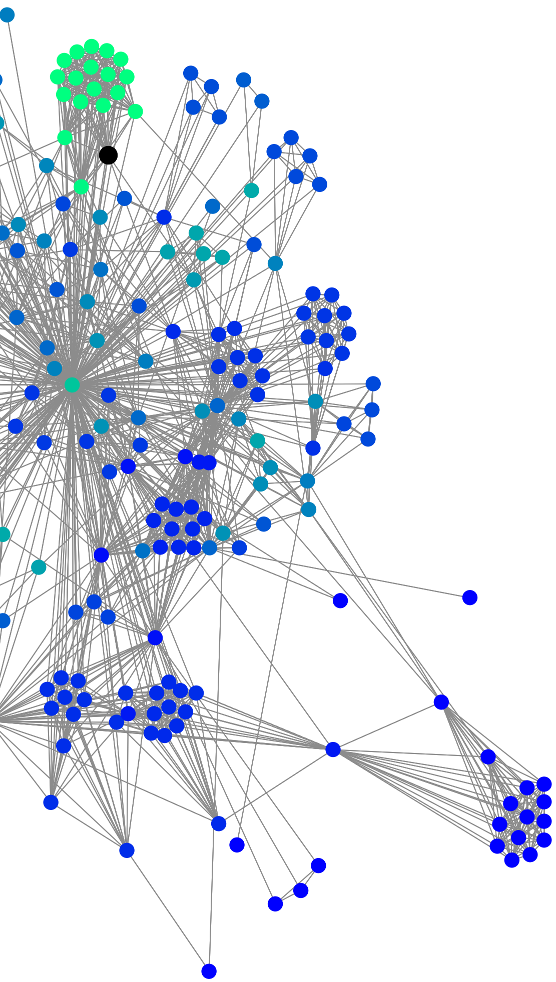
V, nodes

E, edges

- 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

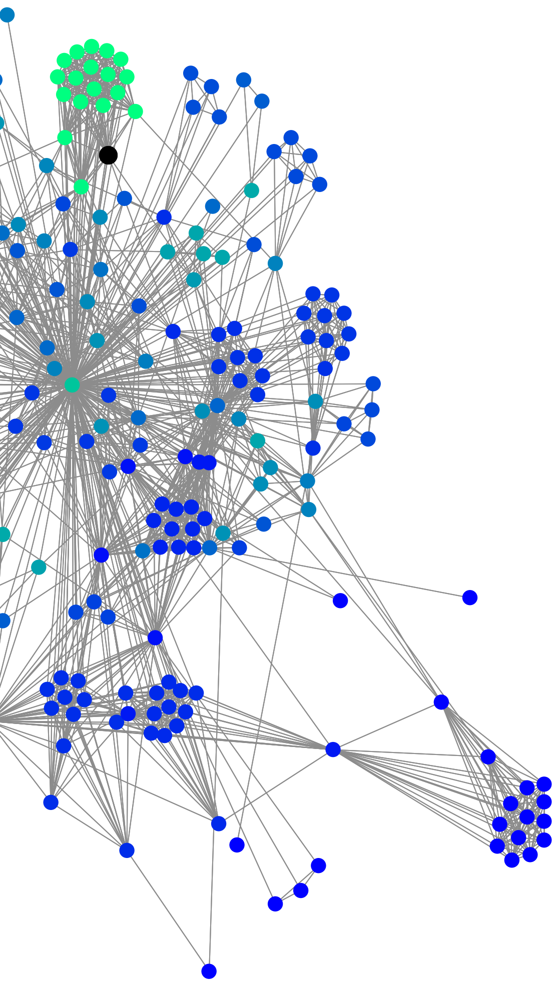
Recommending facebook friends



Each node is a user, the graph has edges between facebook friends.

How should Facebook determine which users to recommend as new friends to the node colored black?

Network Analysis



PageRank

One of the best methods for determining FB friends / Twitter followers is “seeded PageRank”

A diffusion process that leaks dye from target node (seed) to the rest of the graph.

More dye = higher probability that node is your friend!

Mo' data, mo' problems

Nonzero dye on every node (nonzero probability you are friends with each person) -> must look at whole graph to be accurate!

And real-world networks are

- ~ $O(10^9)$ #nodes
- ~ $O(10^{10})$ #edges = $|E|$

Big networks pose a big problem for applications that need fast answers (like "which users should I befriend?")

State-of-the-art c. 2012: "Wild West"

There exist "fast" methods for seeded PageRank, but they were "compute first, ask questions later"* (or not at all!)

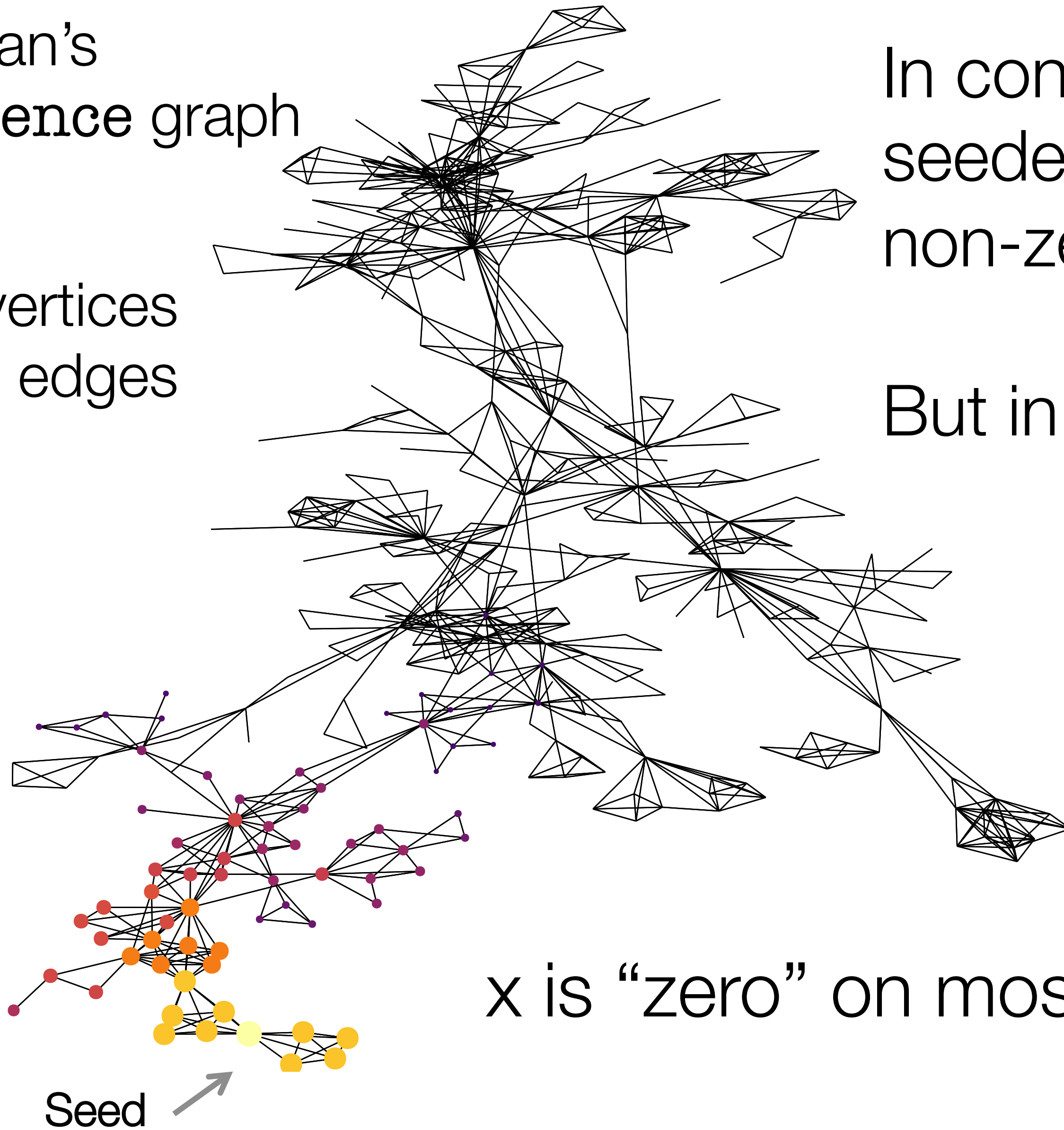
They lacked principled mathematical theory guaranteeing these fast approximations would be **accurate**.

But fast approximate methods "seemed to work"

Localization in seeded PageRank

Newman's `netscience` graph

379 vertices
924 edges



In connected graphs seeded PageRank is non-zero everywhere.

But in practice…

x is "zero" on most of the nodes!

Solution to Big Data: localization

Local algorithms look at just the graph region near the nodes of interest

Localization occurs when a global object can be approximated accurately by being precise in only a small region

Weak and strong localization

Weak localization: an approximation that is sparse, and accurate enough to use in applications that tolerate low accuracy (clustering!)

Strong localization: an approximation that is sparse, and accurate enough for use in **any** application.

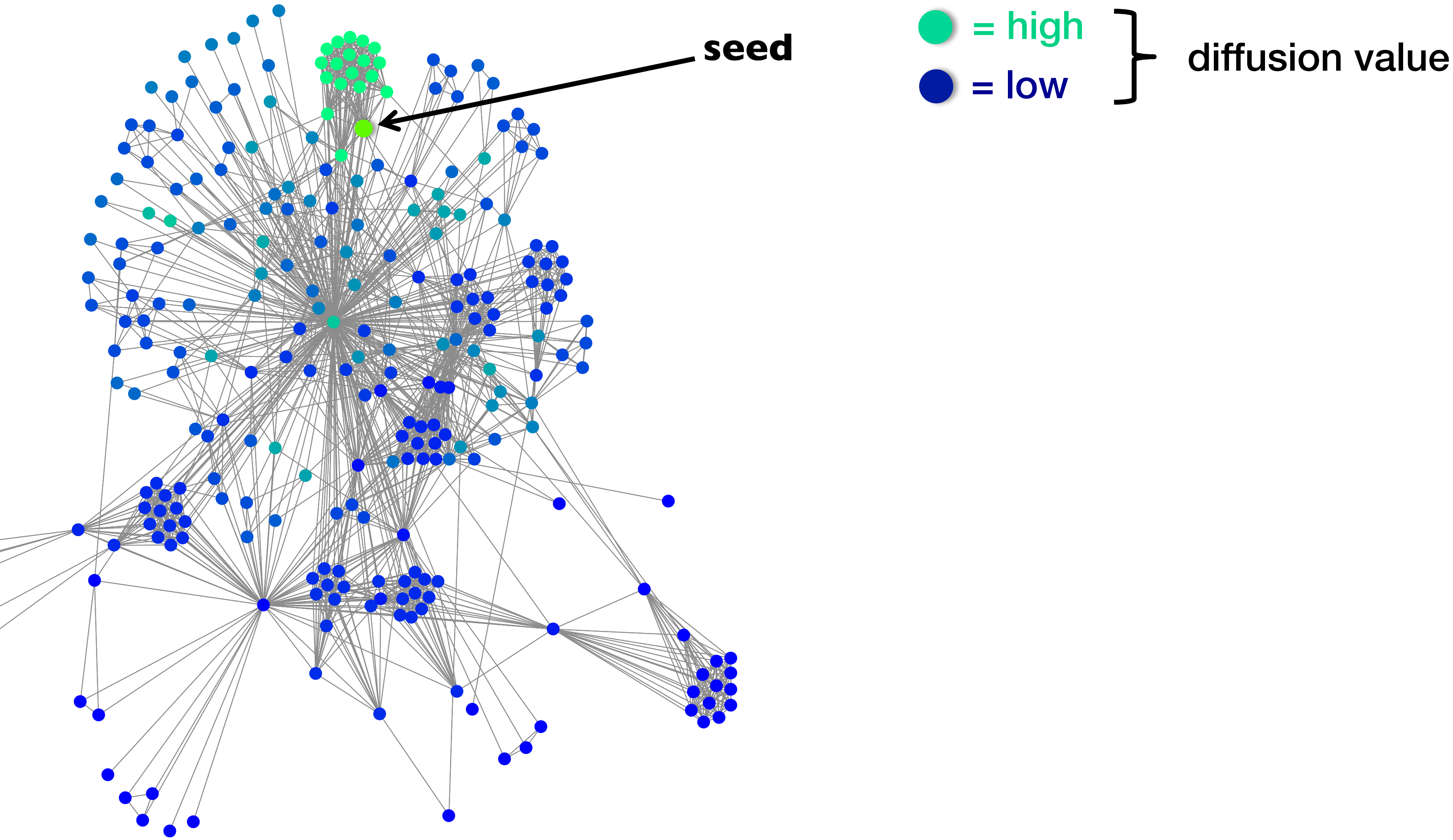
State of the art, 2016/4/22

	Weak localization	Strong localization
PR	✓	[Nassar, K., Gleich, 2015]
HK	[K. & Gleich 2014]	[Gleich & K., 2014]
Gen Diff	In preparation!	?

Weak localization in diffusions

General diffusions: intuition

A diffusion propagates “rank” from a seed across a graph.



Graph Matrices

Adjacency matrix, **A**

$$A_{ij} = \begin{cases} 1, \text{ if node i links to node j} \\ 0 \text{ otherwise} \end{cases}$$

Random-walk transition matrix, **P**

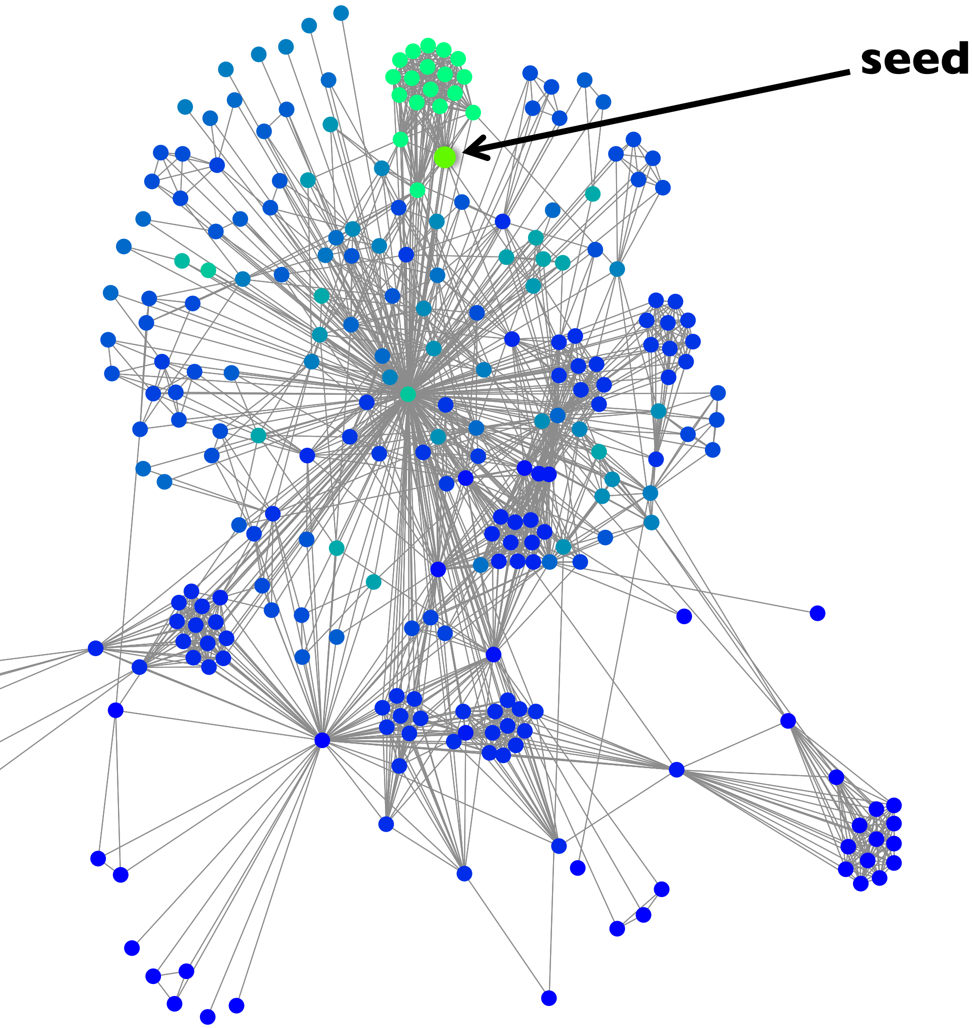
$$P_{ij} = A_{ji} / d_j$$ where d_j is the outdegree of node j.

$$P = A^T D^{-1}$$ where D is the diag degree matrix.

Column stochastic! i.e. column-sums = 1

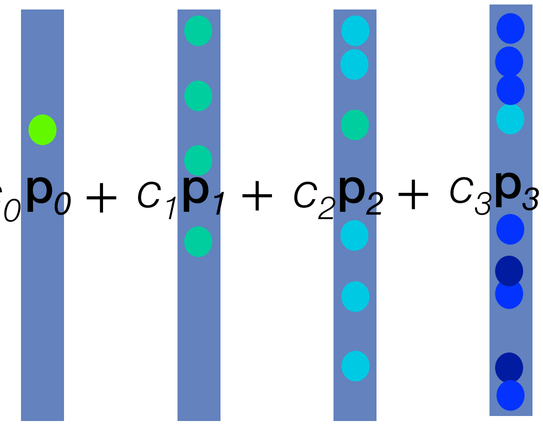
General diffusions: intuition

A diffusion propagates "rank" from a seed across a graph.



General diffusion vector

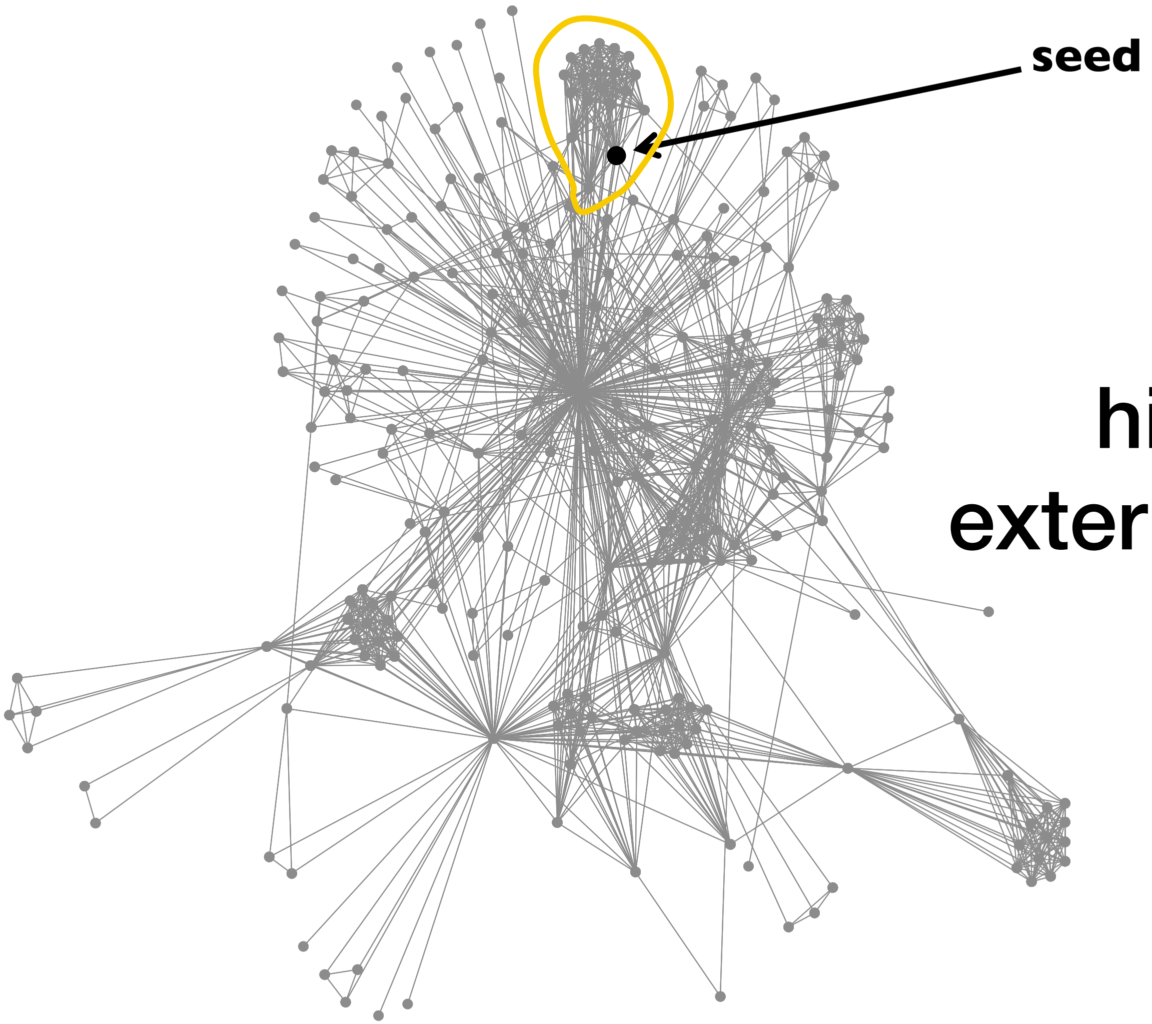
$$\mathbf{f} = \sum_{k=0} c_k \boldsymbol{P}^k \hat{\mathbf{s}} = f(\boldsymbol{P})\hat{\mathbf{s}}$$



$$\mathbf{f} = c_0\mathbf{p}_0 + c_1\mathbf{p}_1 + c_2\mathbf{p}_2 + c_3\mathbf{p}_3 + \ldots$$

Local Community Detection

Given seed(s) *S* in *G*, find a community that contains *S*.



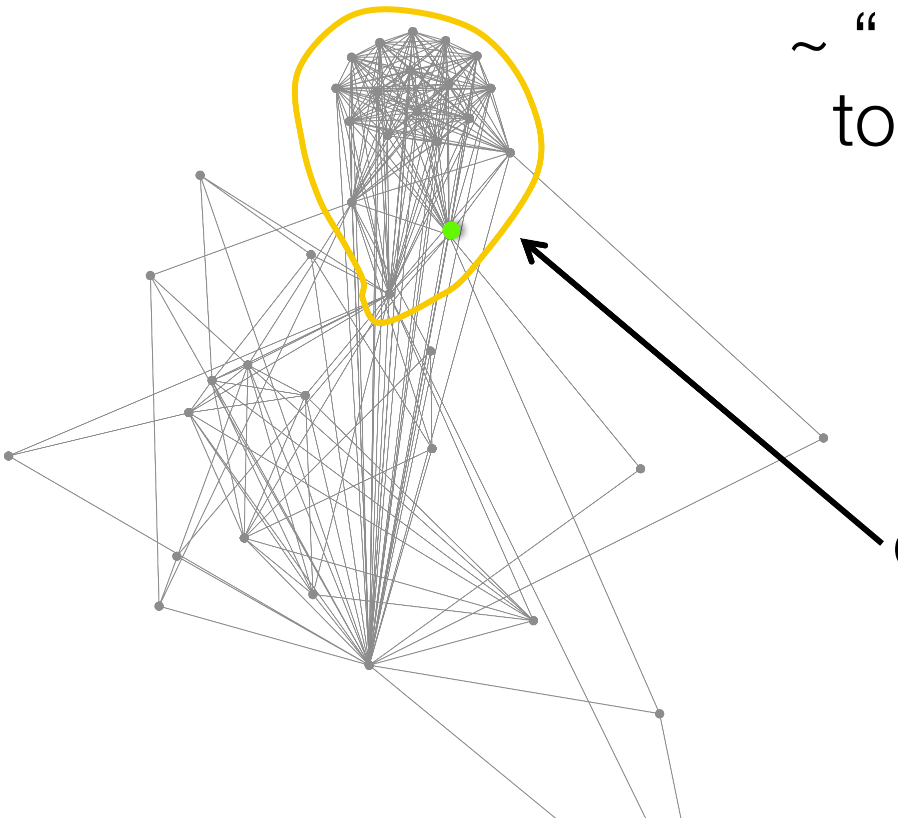
"Community" ?

high internal, low external connectivity

Low-conductance sets are communities

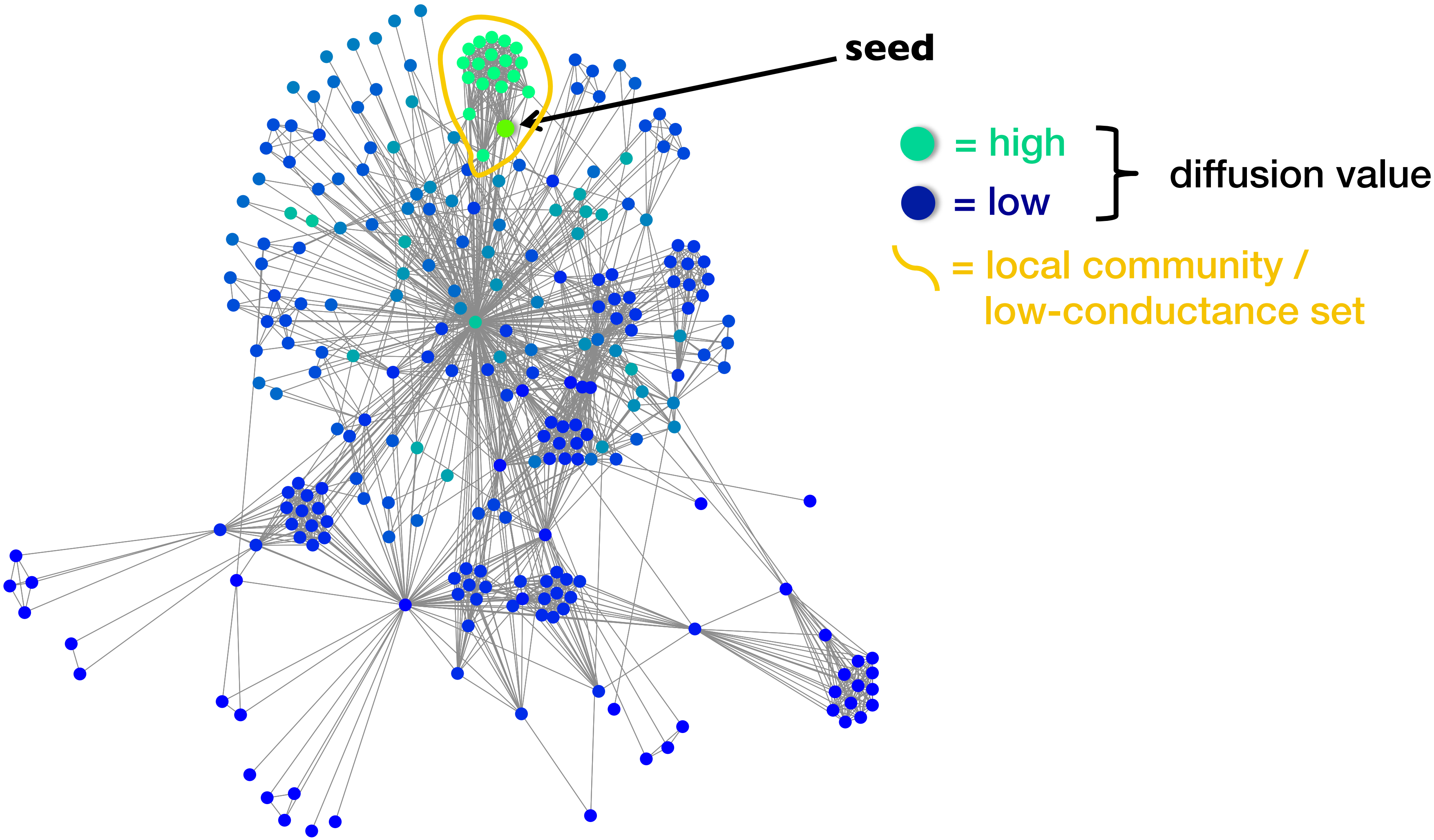
$$\text{conductance}(\ T\ ) = \frac{\text{cut}(T)}{\min(\ \text{vol}(T), \text{vol}(T^c)\ )}$$

~ " chance a random edge touching T also exits *T* "



conductance(comm) = 39/381 = .102

Graph diffusions find low-conductance sets



Use a diffusion for good conductance sets

1. Approximate **f** so $\|\boldsymbol{D}^{-1}(\mathbf{f} - \hat{\mathbf{f}})\|_\infty \le \varepsilon$, $\mathbf{f} \ge \hat{\mathbf{f}} \ge 0$
2. Then "sweep" for best conductance set.

Sweep:

1. Sort diffusion vector so $f_1/d(1) \ge f_2/d(2) \ge \cdots$
2. Consider the sweep sets S(j) = {1,2,…,j}
3. Return the set S(j) with the best conductance.

Weak localization in diffusions

1. Approximate **f** so $\|\boldsymbol{D}^{-1}(\mathbf{f} - \hat{\mathbf{f}})\|_\infty \le \varepsilon$, $\mathbf{f} \ge \hat{\mathbf{f}} \ge 0$

Weak localization:

When an approximation of **f** satisfies $\|\boldsymbol{D}^{-1}(\mathbf{f} - \hat{\mathbf{f}})\|_\infty \le \varepsilon$ and is sparse, the diffusion is weakly localized.

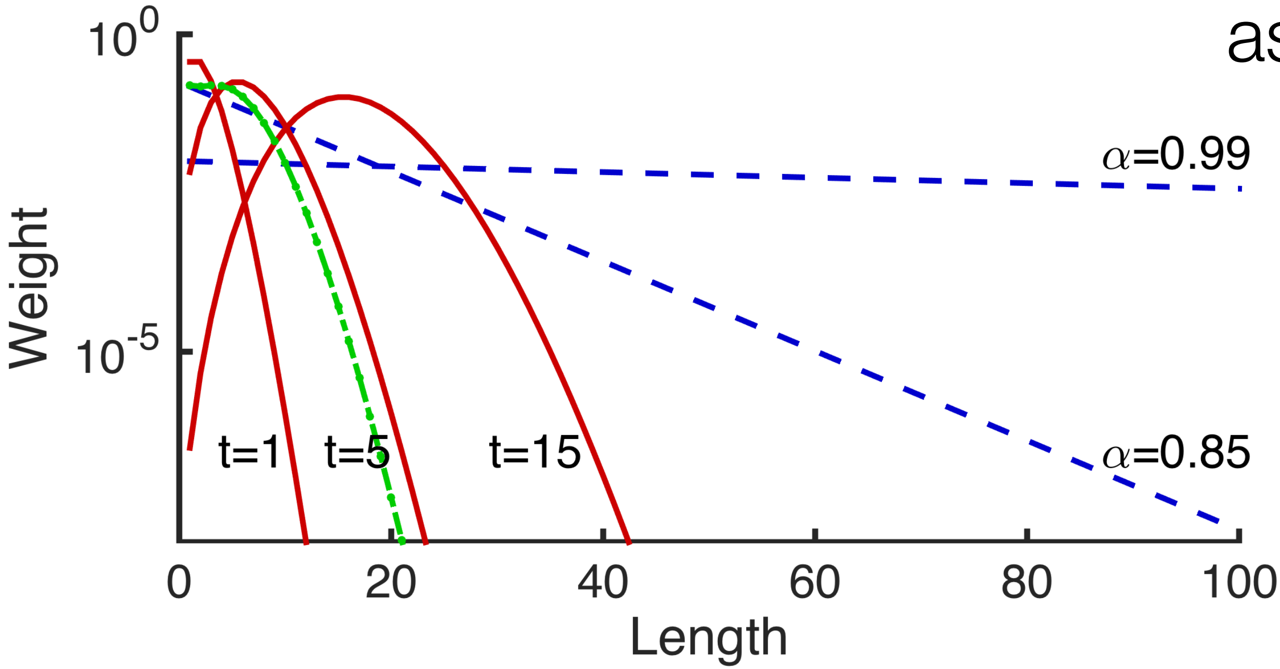
Basically: “get just the biggest entries sort of correct”

Diffusions used for conductance

Personalized PageRank (PPR) $$\mathbf{f} = \sum_{k=0}^{\infty} \alpha^k \boldsymbol{P}^k \tilde{\mathbf{s}}$$

Heat Kernel (HK) $$\mathbf{f} = \sum_{k=0}^{\infty} \frac{t^k}{k!} \boldsymbol{P}^k \tilde{\mathbf{s}}$$

Time-dependent PageRank (TDPR)



Various diffusions explore different aspects of graphs.

Diffusions: conductance & algorithms

	good conductance	fast algorithm
PR	Local Cheeger Inequality [Andersen,Chung,Lang 06]	[Andersen Chung Lang 06] “PPR-push” is $O(1/(\varepsilon(1-\alpha)))$
HK	Local Cheeger Inequality [Chung ’07]	[K., Gleich ’14] “HK-push” is $O(e^{t}C/\varepsilon)$
TDPR	Open question	[Avron, Horesh ’15] Constant-time heuristically
Gen Diff	[Ghosh et al. ’14] on L; open question for general f	In preparation with Gleich and Simpson

Our algorithms for $\hat{\mathbf{f}} \approx f(\boldsymbol{P})\hat{\mathbf{s}}$

- *constant time* on any graph,
    - heat kernel: $\tilde{O}(\frac{e^1}{\varepsilon})$
    - general: $O(\frac{N^2}{\varepsilon})$

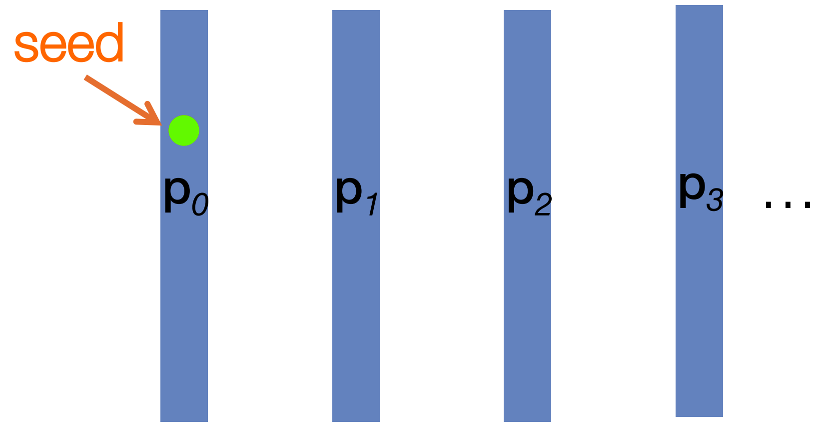
- accuracy: $\|\boldsymbol{D}^{-1}(\mathbf{f} - \hat{\mathbf{f}})\|_\infty \leq \epsilon$
- our experiments show heat kernel outperforms PageRank on real-world communities

General diffusion: Algorithm Intuition

From parameters c_k, ε, seed **s** …

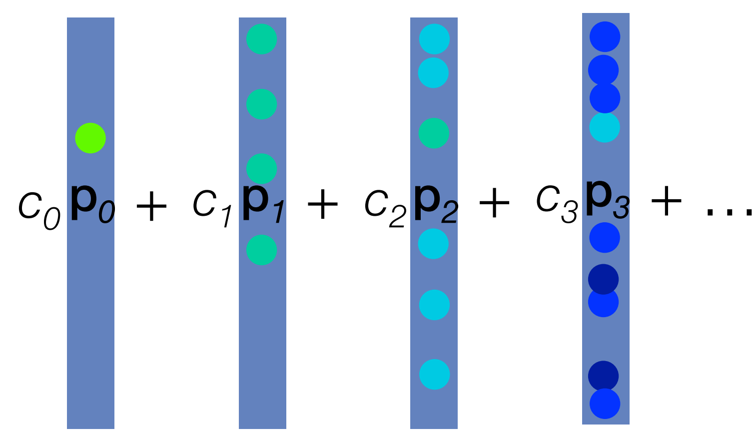
Starting from here…

“residual staging area”:



How to end up here?

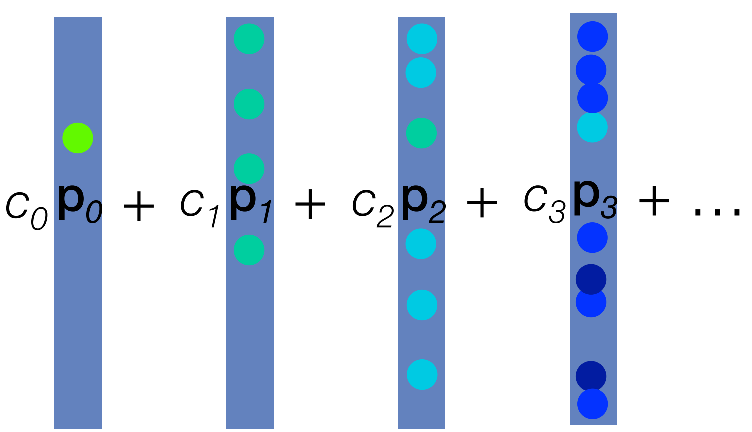
$$\mathbf{f} = \sum_{k=0} c_k \boldsymbol{P}^k \hat{\mathbf{s}} = f(\boldsymbol{P})\hat{\mathbf{s}}$$



General diffusion: Algorithm Intuition

$$\begin{bmatrix} \mathbf{I} & & & & \\ -\mathbf{P} & \mathbf{I} & & & \\ & -\mathbf{P} & \mathbf{I} & & \\ & & -\mathbf{P} & \mathbf{I} & \\ & & & \ddots & \ddots \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \vdots \\ \vdots \end{bmatrix} = \begin{bmatrix} \mathbf{s} \\ 0 \\ 0 \\ \vdots \\ \vdots \end{bmatrix}$$

$$\mathbf{f} = \sum_{k=0} c_k \boldsymbol{P}^k \hat{\mathbf{s}} = f(\boldsymbol{P})\hat{\mathbf{s}}$$

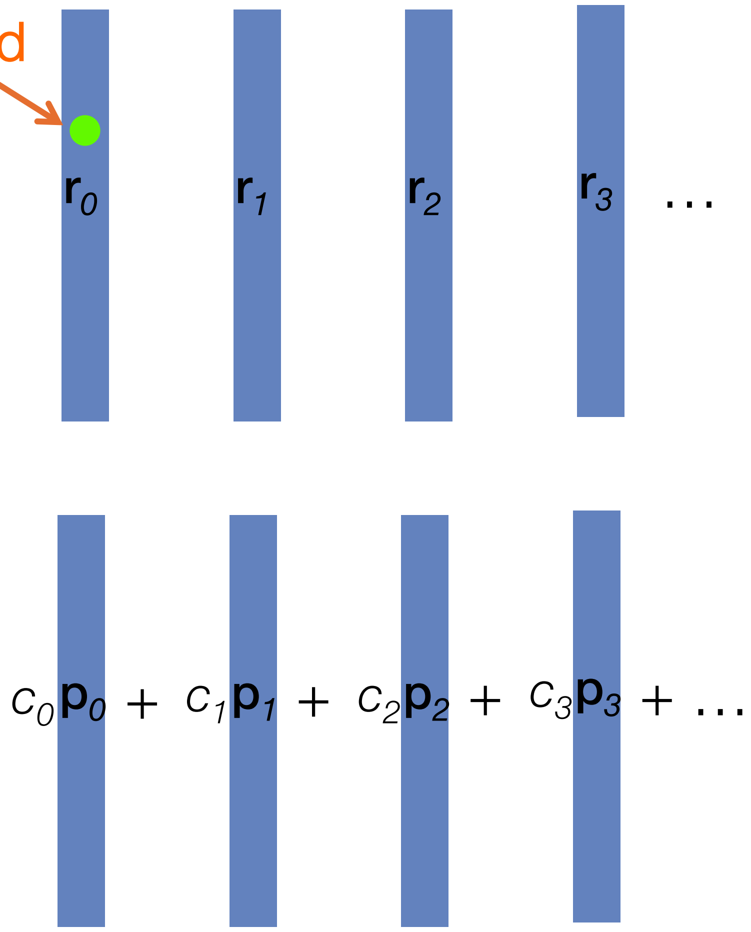


Algorithm Intuition

Begin with mass at seed(s) in a "residual" staging area, $\mathbf{r}_0$

The residuals $\mathbf{r}_k$ hold mass that is unprocessed – it's like *error*

Idea: "push" any entry $r_k(j)/ d_j >$ (some threshold)



Thresholds

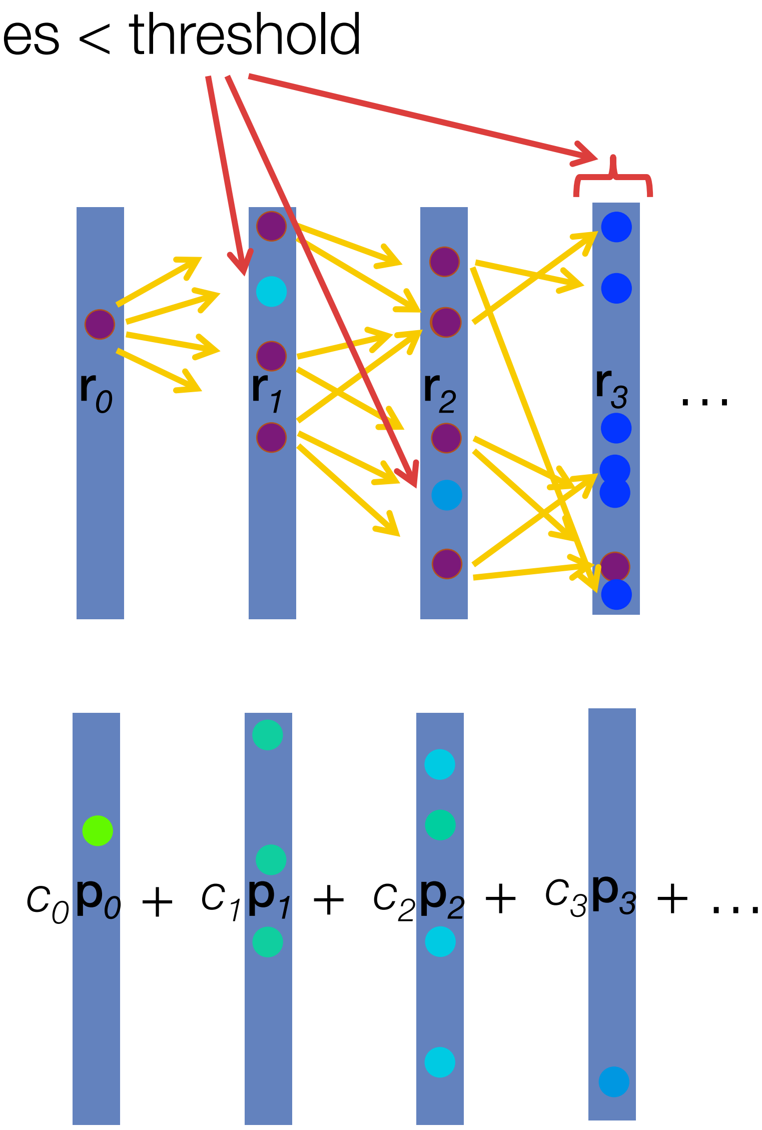
ERROR equals weighted sum of entries left in the vectors $\mathbf{r}_k$

→ Set threshold so "leftovers" sum to $< \varepsilon$

Threshold for stage $\mathbf{r}_k$ is

$$\varepsilon / \left(\sum_{j=k+1}^{\infty} c_j \right)$$

Then $\| \boldsymbol{D}^{-1} (\mathbf{f} - \hat{\mathbf{f}}) \|_\infty \leq \varepsilon$



General diffusions: conclusion

THM: For diffusion coefficients $c_k >= 0$ satisfying

$$\sum_{k=0}^{\infty} c_k = 1 \quad \text{and} \quad \sum_{k=0}^{N} c_k \leq \epsilon/2$$

"rate of decay"

Our algorithm approximates the diffusion **f**

on an undirected graph so that $\|\boldsymbol{D}^{-1}(\mathbf{f} - \hat{\mathbf{f}})\|_\infty \leq \epsilon$

in work bounded by $O(2N^2/\epsilon)$

Constant for any inputs!
(If diffusion decays fast)

Proof sketch

1. Stop pushing after N terms. $\sum_{k=0}^{N} c_k \leq \epsilon/2$

2. Push residual entries in first N terms if $r_k(j) \geq d(j)\epsilon/(2N)$

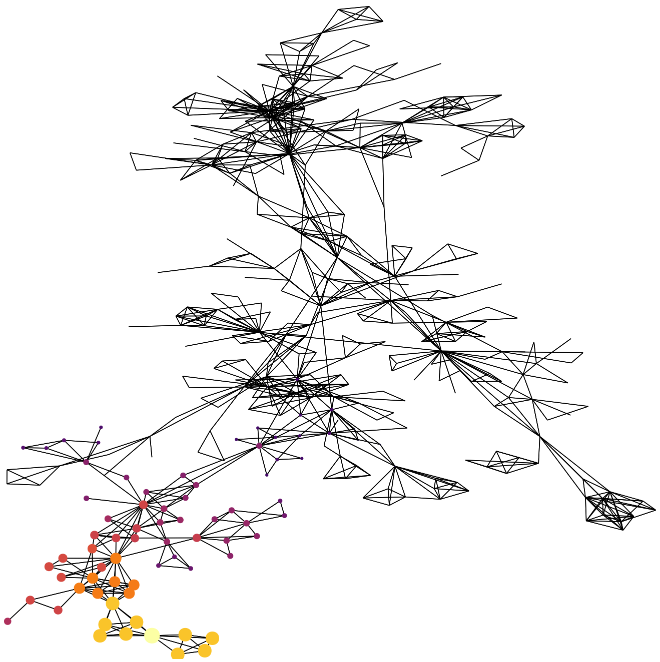
3. Total work is # pushes: $\sum_{k=0}^{N-1} \sum_{t=1}^{m_k} d(j_t) \leq \sum_{k=0}^{N-1} \sum_{t=1}^{m_k} r_k(j_t)(2N)/\epsilon$

4. Each r_k sums to <= 1
(each push is added to **f**, which sums to 1) $\sum_{t=1}^{m_k} r_k(j_t) \leq 1$

$$O(2N^2/\epsilon)$$

Strong localization in seeded PageRank

Strong localization in seeded PageRank



Given a seed and a graph

$\mathbf{e}_s$ $\quad\quad$ $\boldsymbol{P} = \boldsymbol{A}^T \boldsymbol{D}^{-1}$

Seeded PageRank is defined as the solution to

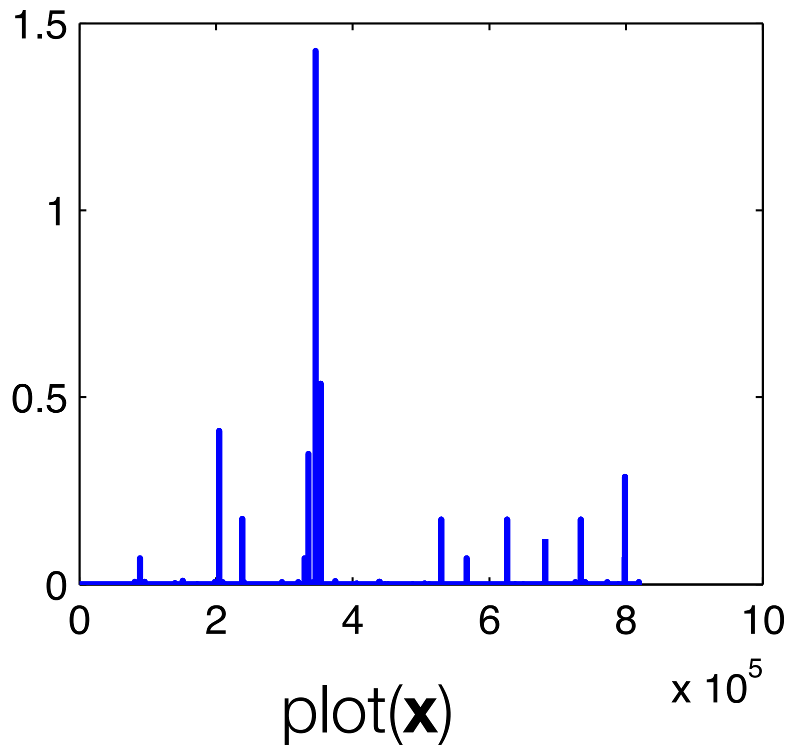
$$(\boldsymbol{I} - \alpha \boldsymbol{P})\mathbf{x} = (1 - \alpha)\mathbf{e}_s$$

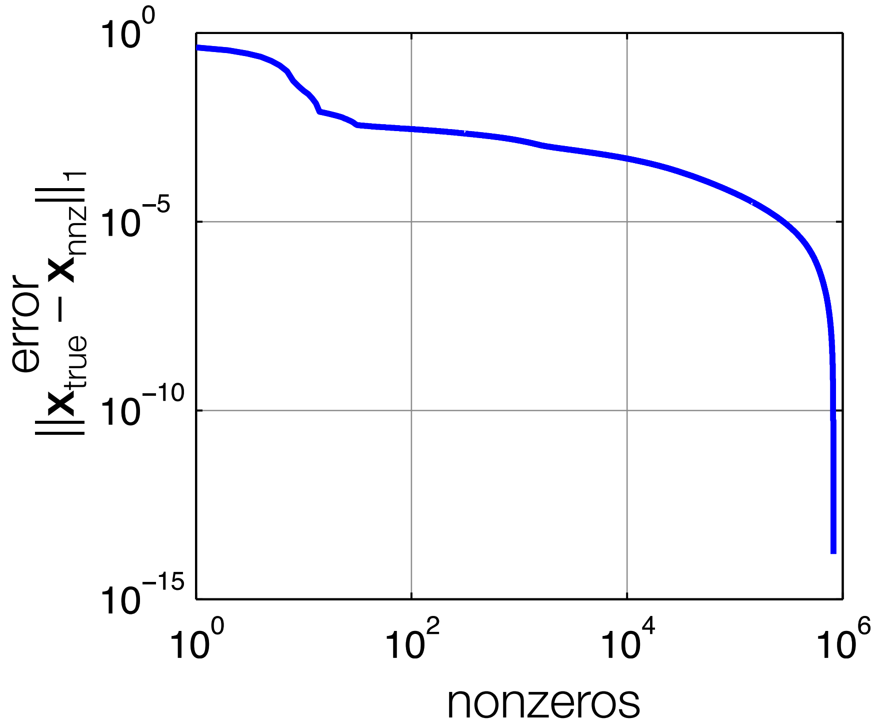
where α is the "teleportation parameter" in (0,1).

Strong localization: if we can approximate **x** so that $\|\mathbf{x} - \hat{\mathbf{x}}\|_1 \leq \varepsilon$ and the approximation is sparse, **x** is strongly localized.

An example on a bigger graph

Crawl of flickr from 2006: ~800K nodes, 6M edges, seeded PageRank with $\alpha = 0.5$





X-axis: node index

Y-axis: value at that index in true PageRank vector

Conditions for localization?

When is localization in diffusions possible?

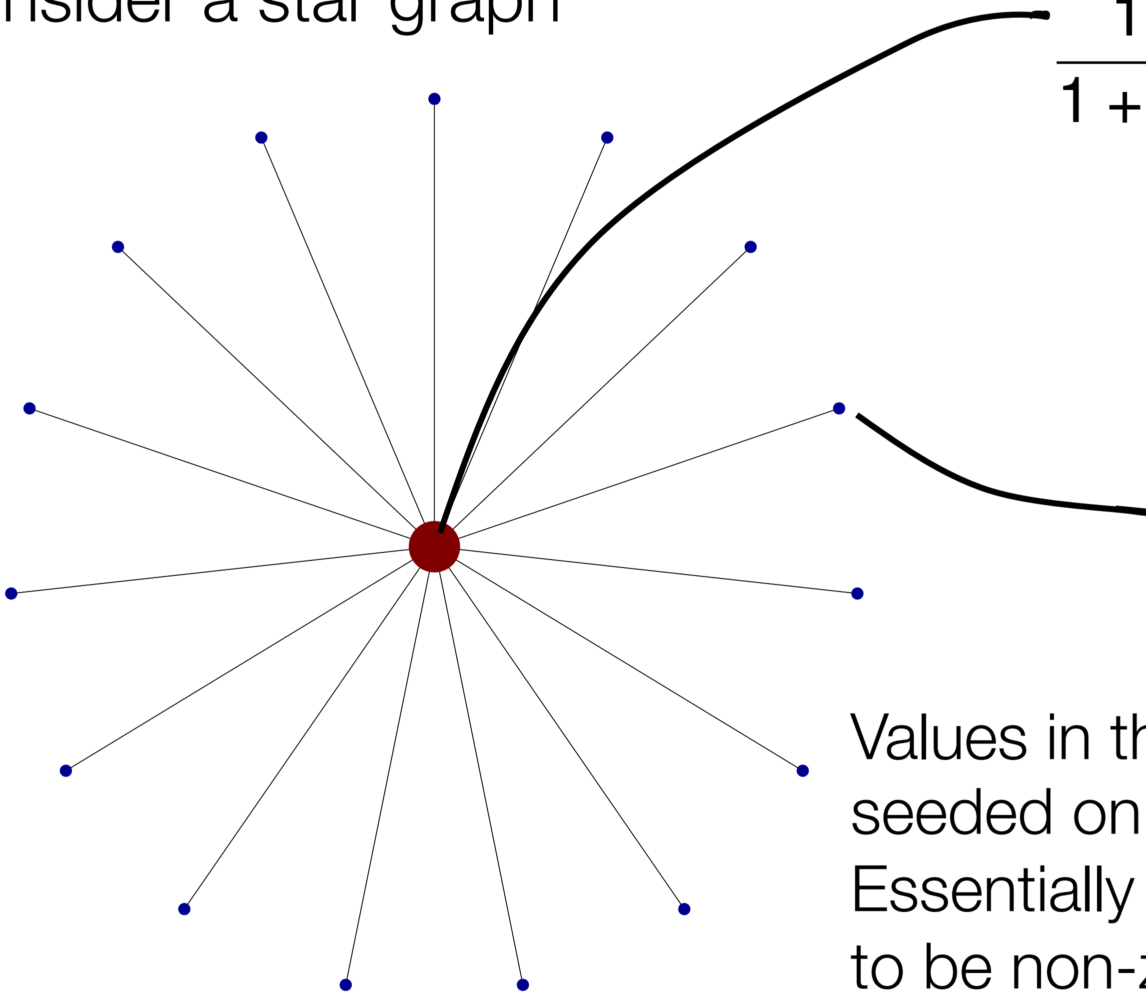
We’ve observed localization in real world graphs.
Does it always occur?

Are there graphs in which no localization occurs?

If localization occurs *everywhere* then our result is less meaningful...

Strong localization can be impossible

Consider a star graph



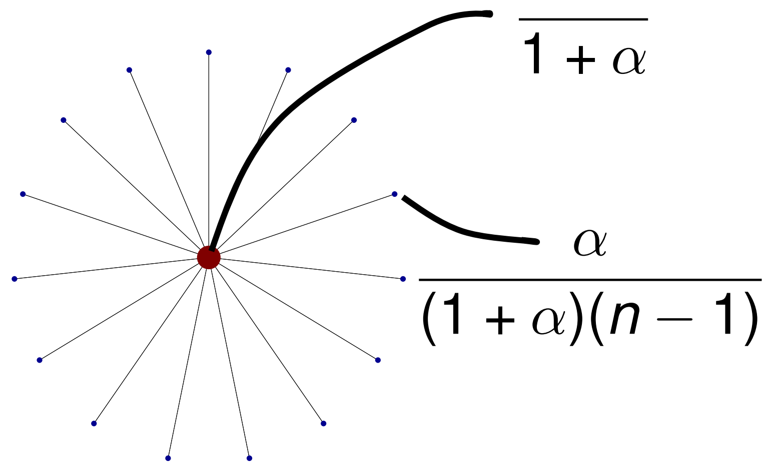
$$\frac{1}{1+\alpha}$$

$$\frac{\alpha}{(1+\alpha)(n-1)}$$

Values in the PageRank vector seeded on the center node. Essentially everything is needed to be non-zero to get a global error bound.

Strong localization can be impossible

Consider a star graph



How many entries in x can we round to zero before its error is too large?

This: $\|\mathbf{x} - \mathbf{x}^*\|_1 \leq \varepsilon$

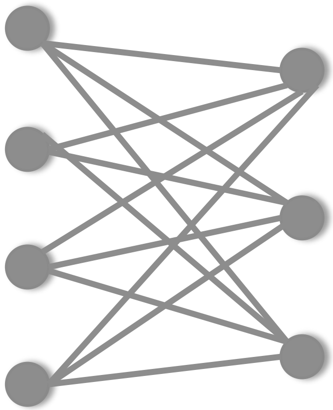
requires

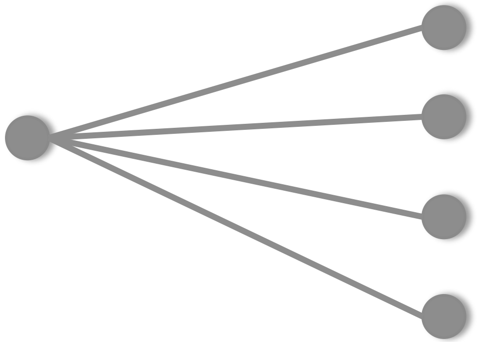
$$1 + n\left(1 - \frac{\varepsilon(1+\alpha)}{\alpha}\right) \leq \text{nnz}(\mathbf{x}^*)$$

Values in the PageRank vector seeded on the center node. Essentially everything is needed to be non-zero to get a global error bound.

Strong localization can be impossible

THM: (Nassar, K., Gleich) Seeded PageRank is non-local on any complete bipartite graphs (generalizing star graphs).





Strong localization can be impossible

THM: (Nassar, K., Gleich) Seeded PageRank is non-local on any complete bipartite graphs (generalizing star graphs).

Why?

Fact: $\boldsymbol{P}$ is complete-bipartite iff eigenvalues = {-1,0,1}.

PageRank is really a matrix function, $f(x) = (1 - \alpha x)^{-1}$.

Fact: a matrix function is equiv to interpolating polynomial

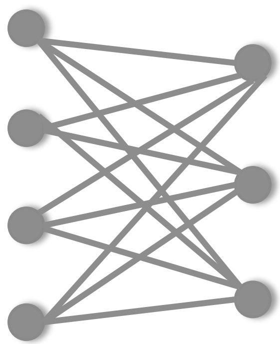
$$p(\lambda_i) = f(\lambda_i) \to p(\boldsymbol{P}) = f(\boldsymbol{P})$$

Only 3 eigenvalues ⟶ p(x) is degree 2 (!)

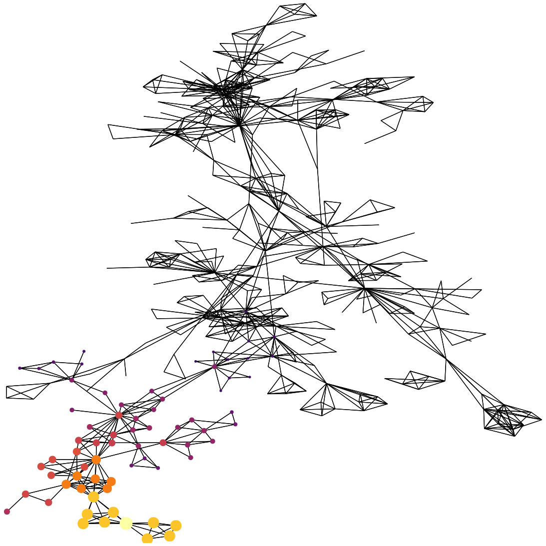
$$(\boldsymbol{I} - \alpha \boldsymbol{P})^{-1}\mathbf{e}_j \quad = \quad f(\boldsymbol{P})\mathbf{e}_j \quad = \quad (c_0\boldsymbol{I} + c_1\boldsymbol{P} + c_2\boldsymbol{P}^2)\mathbf{e}_j$$

When is localization possible?

Graphs exist where seeded PageRank has **no** localized behavior (complete bipartite)



& graphs exist with localized behavior **everywhere** (degree <= constant, or log log(n))



So what properties can

determine localization

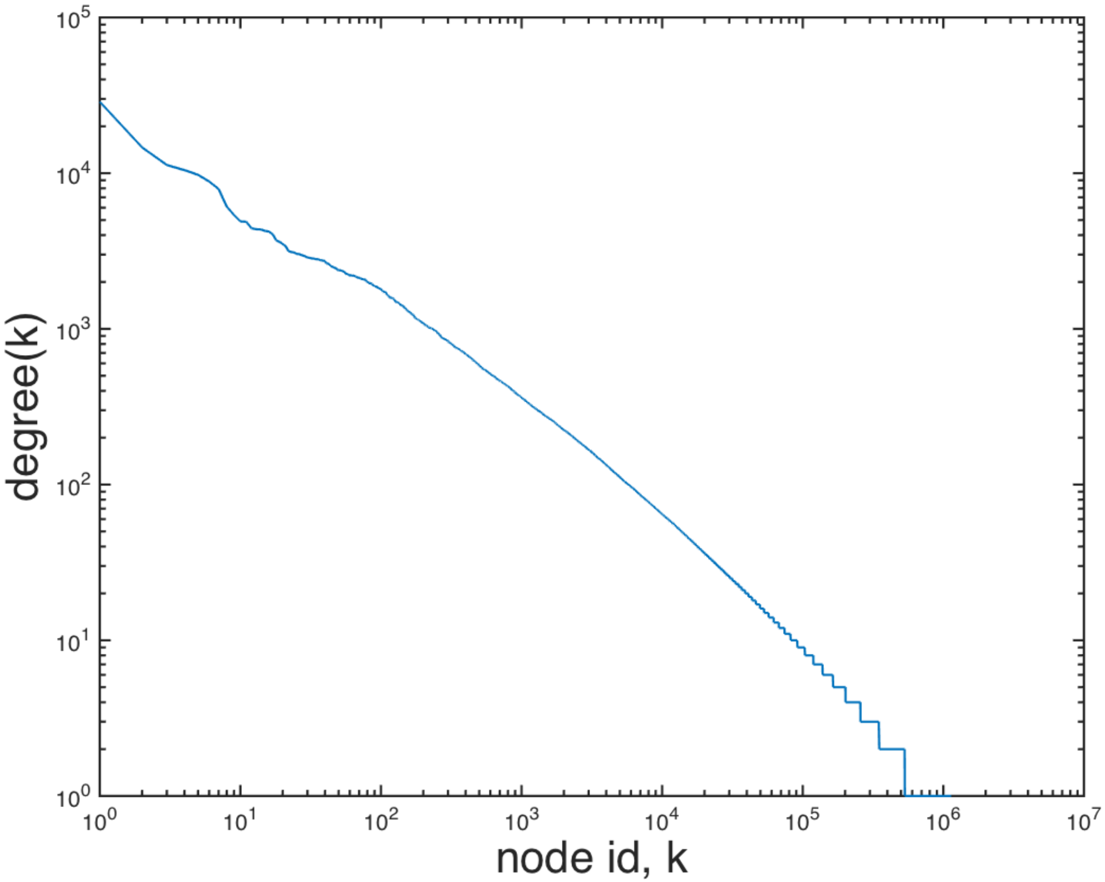
in seeded PageRank?

Skewed degree sequences

Graphs where the k-th largest degree $d(k) \leq \max(dk^{-p}, \delta)$

(δ is min degree,
d is max degree
p is decay exponent)

Youtube degree nodes



[Yang and Leskovec, ICDM 2015]

Log-log scale

1.1M nodes
3M edges
p ~ 0.71

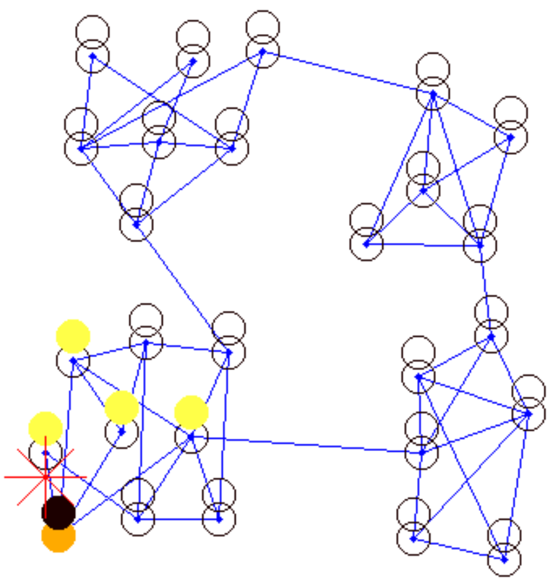
Strong localization in personalized PageRank Vectors

Theorem (Nassar, K., Gleich):
Let a graph have max-degree d, min-degree δ, n nodes, and let p be the decay exponent. Then Gauss Southwell computes $\mathbf{x}_\varepsilon$ with accuracy $\|\mathbf{x} - \mathbf{x}_\varepsilon\|_1 \leq \varepsilon$, and the number of non-zeros in $\mathbf{x}_\varepsilon$ is no greater than: $\min\left\{ n\,,\ \frac{1}{\delta} C_p (1/\varepsilon)^{\frac{\delta}{1-\alpha}} \right\}$

where $C_p = \begin{cases} d(1 + \log d) & p = 1 \\ d\left(1 + \frac{1}{1-p}(d^{(1/p)-1} - 1)\right) & \text{otherwise} \end{cases}$

Due to the maximum degree d, this does not say anything about traditional power-law graphs (e.g. the Pareto case)

Strong localization in personalized PageRank Vectors (sketch)



We study the behavior of the *Gauss-Southwell or push algorithm* for computing PageRank

- residual = remaining rank/dye to assign
- solution = assigned rank/dye

Algorithm

1. pick node with most residual dye
2. assign dye to node
3. update residual dye on neighbors,
4. then repeat.

Coordinate relaxation for PageRank

Approximating

$$(\boldsymbol{I} - \alpha \boldsymbol{P})\mathbf{x} = \tilde{\mathbf{s}}$$

Initial solution and residual: $\mathbf{x}^{(0)} = 0, \mathbf{r}^{(0)} = (1-\alpha)\mathbf{s}$

Iterative updates: first pick entry of residual, j

- update solution: $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + r_j \cdot \mathbf{e}_j$
- update residual: $\mathbf{r}^{(k+1)} = \tilde{\mathbf{s}} - (\boldsymbol{I} - \alpha \boldsymbol{P})\mathbf{x}^{(k+1)}$

$$= \mathbf{r}^{(k)} - r_j \mathbf{e}_j + r_j \alpha \boldsymbol{P} \mathbf{e}_j$$

KEY: non-zeros in solution
bounded by number of iterations, k+1

PageRank Convergence: error & residual

Approximating a solution to

$$(\boldsymbol{I} - \alpha \boldsymbol{P})\mathbf{x} = \tilde{\mathbf{s}}$$

residual and error satisfy

$$\begin{aligned}
\mathbf{r}^{(k)} &= \tilde{\mathbf{s}} - (\boldsymbol{I} - \alpha \boldsymbol{P})\mathbf{x}^{(k)} \\
&= (\boldsymbol{I} - \alpha \boldsymbol{P})\mathbf{x} - (\boldsymbol{I} - \alpha \boldsymbol{P})\mathbf{x}^{(k)} \\
(\boldsymbol{I} - \alpha \boldsymbol{P})^{-1}\mathbf{r}^{(k)} &= (\mathbf{x} - \mathbf{x}^{(k)}) \\
\|\mathbf{x} - \mathbf{x}^{(k)}\| &\leq \|(\boldsymbol{I} - \alpha \boldsymbol{P})^{-1}\| \|\mathbf{r}^{(k)}\|
\end{aligned}$$

for any sub-multiplicative matrix norm $\| \; \|$.

PageRank Convergence: residual bound

Approximating a solution to $(\boldsymbol{I} - \alpha \boldsymbol{P})\mathbf{x} = \tilde{\mathbf{s}}$

Error satisfies $\|\mathbf{x} - \mathbf{x}^{(k)}\| \leq \|(\boldsymbol{I} - \alpha \boldsymbol{P})^{-1}\| \|\mathbf{r}^{(k)}\|$

Initial solution and residual: $\mathbf{x}^{(0)} = 0, \mathbf{r}^{(0)} = (1 - \alpha)\mathbf{s}$

Update residual: $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - r_j \mathbf{e}_j + r_j \alpha \boldsymbol{P} \mathbf{e}_j$

$$
\begin{aligned}
\|\mathbf{r}^{(k+1)}\|_1 &\leq \|\mathbf{r}^{(k)} - r_j \mathbf{e}_j\|_1 + \|r_j \alpha \boldsymbol{P} \mathbf{e}_j\|_1 && \text{Triangle inequality} \\
&\leq \|\mathbf{r}^{(k)}\|_1 - r_j + |r_j \alpha| \|\boldsymbol{P} \mathbf{e}_j\|_1 && \text{Residual nonnegative} \\
&\leq \|\mathbf{r}^{(k)}\|_1 - r_j + |r_j \alpha| && \mathbf{P} \text{ is column-stochastic} \\
&\leq \|\mathbf{r}^{(k)}\|_1 - r_j(1 - \alpha) && \text{Residual nonnegative}
\end{aligned}
$$

PageRank Convergence: residual bound

Approximating a solution to $(\boldsymbol{I} - \alpha \boldsymbol{P})\mathbf{x} = \tilde{\mathbf{s}}$

Error satisfies $\|\mathbf{x} - \mathbf{x}^{(k)}\| \le \|(\boldsymbol{I} - \alpha \boldsymbol{P})^{-1}\| \|\mathbf{r}^{(k)}\|$

Initial solution and residual: $\mathbf{x}^{(0)} = 0, \mathbf{r}^{(0)} = (1 - \alpha)\mathbf{s}$

Residual norm: $\|\mathbf{r}^{(k+1)}\|_1 \le \|\mathbf{r}^{(k)}\|_1 - r_j(1 - \alpha)$

Assume we chose r_j to be at least as big as the average magnitude of the residual entries. Then

$$r_j \ge \|\mathbf{r}^{(k)}\|_1 / \mathrm{nnz}(\mathbf{r}^{(k)}) \quad \text{(definition of average)}$$

Bounding $\mathrm{nnz}(\mathbf{r}^{(k)})$ -- use the skewed degree seq!

PageRank Convergence: the weeds, brief

Degree sequence assumption: $d(t) \leq d \cdot t^{-p}$

enables us to prove: $\mathrm{nnz}(\mathbf{r}^{(k)}) \leq C_p + \delta k$

[.... skipping the thickest weeds ....]

which enables a bound on residual decay!

$$\|\mathbf{r}^{(k+1)}\|_1 \leq (1-\alpha)\left((\delta(k+1) + C_p)/C_p\right)^{-(1-\alpha)/\delta}$$

(recall that $C_p \approx d \log d$ and δ is min degree)

Strong localization in personalized PageRank Vectors (repeated)

Theorem (Nassar, K., Gleich):
Let a graph have max-degree d, min-degree δ, n nodes, and let p be the decay exponent. Then Gauss Southwell computes $\mathbf{x}_\varepsilon$ with accuracy $\|\mathbf{x} - \mathbf{x}_\varepsilon\|_1 \leq \varepsilon$, and the number of non-zeros in $\mathbf{x}_\varepsilon$ is no greater than: $\min \left\{ n \,,\ \frac{1}{\delta} C_p (1/\varepsilon)^{\frac{\delta}{1-\alpha}} \right\}$

where $C_p = \begin{cases} d(1 + \log d) & p = 1 \\ d\left(1 + \frac{1}{1-p}(d^{(1/p)-1} - 1)\right) & \text{otherwise} \end{cases}$

Only C_p depends on n, the rest are constants!

State of the art, 2016/4/22

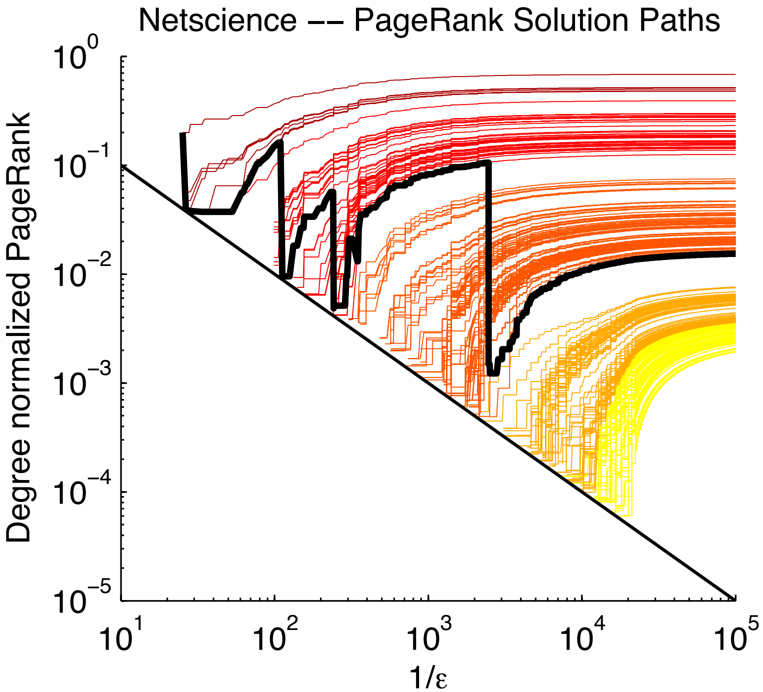
	Weak localization	Strong localization
PR	✓	[Nassar, K., Gleich, 2015]
HK	[Gleich & K. 2014]	[Gleich & K., 2014]
Gen Diff	In preparation!	?

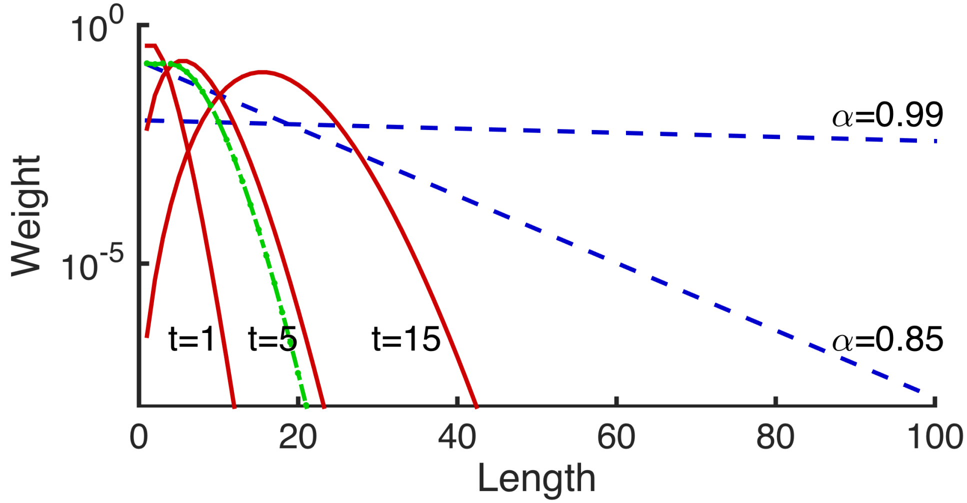
Diffusion paths

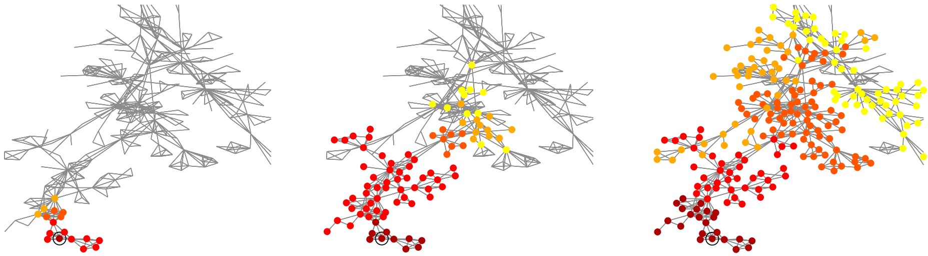
(K. & Gleich)

AptRank: Adaptive diffusions

(Jiang, K., Gleich, Gribskov)







$$\mathbf{f} = \sum_{k=0} c_k \boldsymbol{P}^k \hat{\mathbf{s}} = f(\boldsymbol{P})\hat{\mathbf{s}}$$

Thank you

David F. Gleich (Purdue CS professor)
advisor and collaborator on all projects

Huda Nassar (Purdue CS grad student)
collaborator on PageRank localization and de-localization

Olivia Simpson (UCSD CS grad student)
collaborator on generalized diffusion work

The whole research group, for many proofreads in slides and papers alike: Nichole Eikmeier, Yangyang Hou, Huda Nassar, Bryan Rainey, Yanfei Ren, Ayan Sinha, Varun Vasudevan, Nate Veldt, Tau Wu