A Nearly Sublinear Approximation to $\exp\{\mathbf{P}\}\mathbf{e}_i$ for Large Sparse Matrices from Social Networks

Kyle Kloster and David F. Gleich

Purdue University

December 14, 2013

Supported by NSF CAREER 1149756-CCF

Columns of the Matrix Exponential

exp{ $\mathbf{A}$ } used for link-prediction, node centrality, and clustering. Why?

Columns of the Matrix Exponential

exp{ **A** } used for link-prediction, node centrality, and clustering. Why?

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

- $(\mathbf{A}^k)_{ij}$ gives the number of length-k walks from i to j, so...
- Large entries of $\exp\{\mathbf{A}\}$ denote "important" nodes / links

Columns of the Matrix Exponential

exp{ $\mathbf{A}$ } used for link-prediction, node centrality, and clustering. Why?

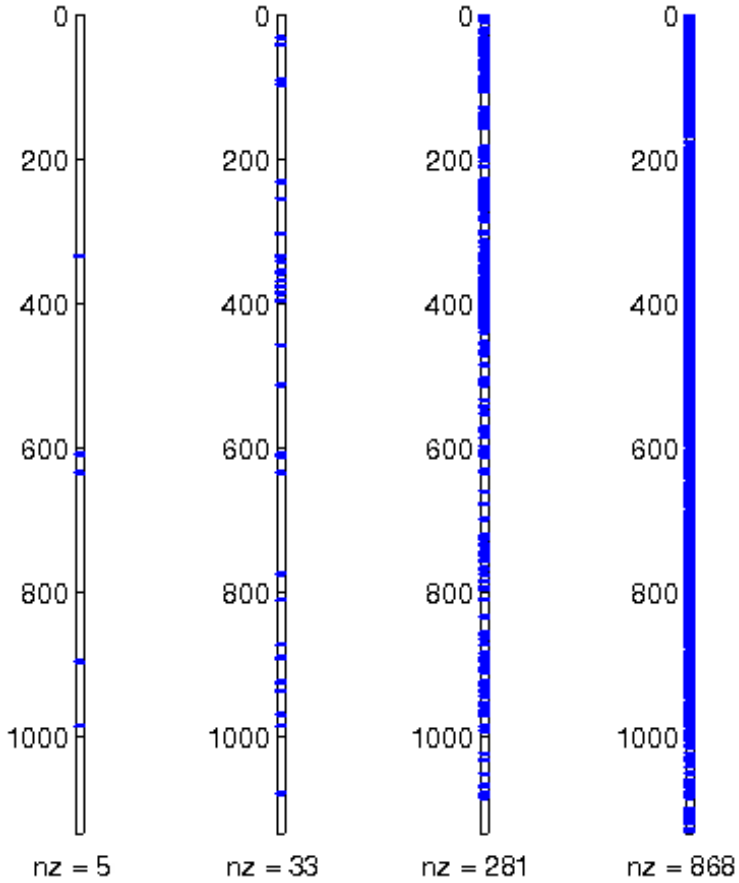
$$\exp\{\mathbf{A}\} = \sum_{k=0}^{\infty} \frac{1}{k!}\mathbf{A}^k$$

- $(\mathbf{A}^k)_{ij}$ gives the number of length-k walks from i to j, so...
- Large entries of $\exp\{\mathbf{A}\}$ denote "important" nodes / links
- $\exp\{\mathbf{A}\}$ is common, but other $f(\mathbf{A})$ can be used: pagerank and heatkernel PR
- Assume column stochastic, $\mathbf{P} = \mathbf{G}\mathbf{D}^{-1}$ (more on this later)

Difficulties with current methods: Sidje, TOMS 1998; Al-Mohy and Higham, SISC 2011

- Leading methods for $\exp\{\mathbf{A}\}\mathbf{b}$ use Krylov or Taylor methods: "basically" repeated mat-vecs
- "Small world" property: graph diameter $\leq 4 \Rightarrow$ repeated mat-vecs fill in rapidly (see picture)
- Not designed specifically for sparse networks.

Fill-in from repeated matvecs



Vectors $\mathbf{P}^k\mathbf{e}_i$ for $k = 1, 2, 3, 4$. $n = 1133$

Local Method

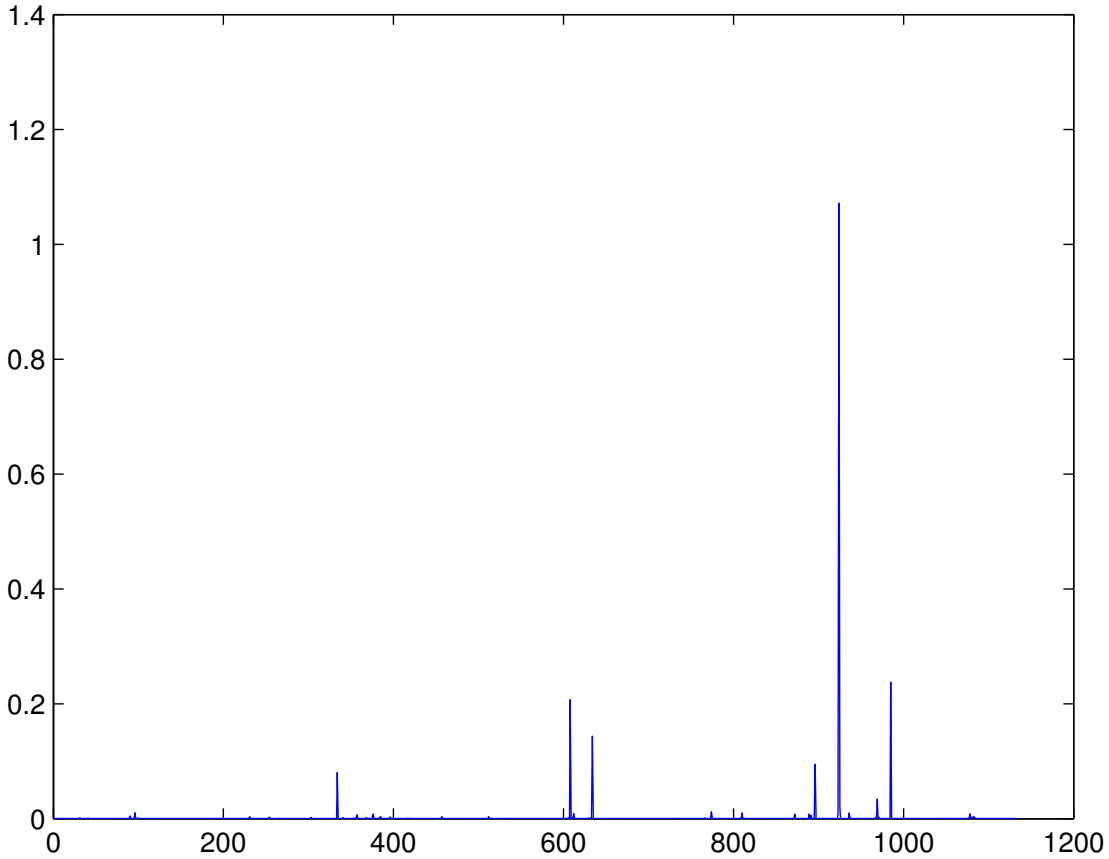
New method: avoid mat-vecs! $\rightarrow$ use a **local** method.

Local algorithms run in time proportional to size of output:

$$\text{sparse solution vector} = \text{small runtime}$$

Instead of matvecs, we do specially-selected vector adds using a relaxation method.

$\exp\{\mathbf{P}\}\mathbf{e}_i$ is a localized vector



x-axis: vector index, y-axis: magnitude of entry

the column of $\exp\{\mathbf{P}\}$ produced by previous slide's matvecs

Overview

Outline of Nexpokit method

1. Express $\exp\{\mathbf{A}\}\mathbf{e}_i$ via a Taylor polynomial
2. Form large linear system out of Taylor terms
3. Use sparse solver to approximate each terms' largest entries
4. Combine approximated terms into a solution

In terms of Taylor terms

Taylor polynomial:

$$\exp\{\mathbf{A}\}\mathbf{e}_i \approx \left(\mathbf{I} + \mathbf{A} + \tfrac{1}{2}\mathbf{A}^2 + \tfrac{1}{3!}\mathbf{A}^3 + \cdots + \tfrac{1}{N!}\mathbf{A}^N\right)\mathbf{e}_i$$

In terms of Taylor terms

Taylor polynomial:

$$\exp\{\mathbf{A}\}\mathbf{e}_i \approx \left(\mathbf{I} + \mathbf{A} + \tfrac{1}{2}\mathbf{A}^2 + \tfrac{1}{3!}\mathbf{A}^3 + \cdots + \tfrac{1}{N!}\mathbf{A}^N\right)\mathbf{e}_i$$

Compute terms recursively: $\mathbf{v}_k = \frac{1}{k!}\mathbf{A}^k\mathbf{e}_i = \frac{1}{k}\mathbf{A}\left(\frac{1}{(k-1)!}\mathbf{A}^{k-1}\right)\mathbf{e}_i$

$$\mathbf{v}_k = \tfrac{1}{k}\mathbf{A}\mathbf{v}_{k-1}$$

In terms of Taylor terms

Taylor polynomial:

$$\exp\{\mathbf{A}\}\mathbf{e}_i \approx \left(\mathbf{I} + \mathbf{A} + \tfrac{1}{2}\mathbf{A}^2 + \tfrac{1}{3!}\mathbf{A}^3 + \cdots + \tfrac{1}{N!}\mathbf{A}^N\right)\mathbf{e}_i$$

Compute terms recursively: $\mathbf{v}_k = \frac{1}{k!}\mathbf{A}^k\mathbf{e}_i = \frac{1}{k}\mathbf{A}\left(\frac{1}{(k-1)!}\mathbf{A}^{k-1}\right)\mathbf{e}_i$

$$\mathbf{v}_k = \tfrac{1}{k}\mathbf{A}\mathbf{v}_{k-1}$$

Then $\exp\{\mathbf{A}\}\mathbf{e}_i \approx \mathbf{v}_0 + \mathbf{v}_1 + \cdots + \mathbf{v}_{N-1} + \mathbf{v}_N$
(But we want to avoid computing $\mathbf{v}_j$ in full...)

Forming a linear system

So we convert the Taylor polynomial into a linear system:

Forming a linear system

So we convert the Taylor polynomial into a linear system:

$$\begin{bmatrix} \mathbf{I} & & & & \\ -\mathbf{A}/1 & \mathbf{I} & & & \\ & -\mathbf{A}/2 & \ddots & & \\ & & \ddots & \mathbf{I} & \\ & & & -\mathbf{A}/N & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{v}_0 \\ \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_N \end{bmatrix} = \begin{bmatrix} \mathbf{e}_i \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Forming a linear system

So we convert the Taylor polynomial into a linear system:

$$\begin{bmatrix} \mathbf{I} & & & & \\ -\mathbf{A}/1 & \mathbf{I} & & & \\ & -\mathbf{A}/2 & \ddots & & \\ & & \ddots & \mathbf{I} & \\ & & & -\mathbf{A}/N & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{v}_0 \\ \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_N \end{bmatrix} = \begin{bmatrix} \mathbf{e}_i \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

From
$\mathbf{v}_k = \frac{1}{k}\mathbf{A}\mathbf{v}_{k-1}$
$\exp\{\mathbf{A}\}\mathbf{e}_i \approx \mathbf{v}_0 + \mathbf{v}_1 + \cdots + \mathbf{v}_{N-1} + \mathbf{v}_N$
(never formed explicitly)

Sparse solver: Gauss Southwell

Basic idea of Gauss Southwell (GS): solving $\mathbf{Mx} = \mathbf{b}$ when $\mathbf{x}$ is "effectively sparse" (i.e. a localized vector)

Sparse solver: Gauss Southwell

Basic idea of Gauss Southwell (GS): solving $\mathbf{Mx} = \mathbf{b}$ when $\mathbf{x}$ is "effectively sparse" (i.e. a localized vector)

1. Set $\mathbf{x}^0 = 0$, $\mathbf{r}^0 = \mathbf{b}$, then iterate:

Sparse solver: Gauss Southwell

Basic idea of Gauss Southwell (GS): solving $\mathbf{Mx} = \mathbf{b}$ when $\mathbf{x}$ is "effectively sparse" (i.e. a localized vector)

1. Set $\mathbf{x}^0 = 0$, $\mathbf{r}^0 = \mathbf{b}$, then iterate:
2. At step k, relax largest entry of $\mathbf{r}^k$ (denoted r_i^k), add to $\mathbf{x}^k$;

$$\mathbf{x}^{k+1} = \mathbf{x}^k + r_i^k \cdot \mathbf{e}_i$$

Sparse solver: Gauss Southwell

Basic idea of Gauss Southwell (GS): solving $\mathbf{Mx} = \mathbf{b}$ when $\mathbf{x}$ is "effectively sparse" (i.e. a localized vector)

1. Set $\mathbf{x}^0 = 0$, $\mathbf{r}^0 = \mathbf{b}$, then iterate:
2. At step k, relax largest entry of $\mathbf{r}^k$ (denoted r_i^k), add to $\mathbf{x}^k$;

$$\mathbf{x}^{k+1} = \mathbf{x}^k + r_i^k \cdot \mathbf{e}_i$$

3. Add corresponding column of $\mathbf{M}$ to residual:

$$\mathbf{r}^{k+1} = (\mathbf{r}^k - r_i^k \cdot \mathbf{e}_i) + r_i^k \cdot \mathbf{M}(:, i)$$

NEXPOKIT

Apply GS to our linear system, $\mathbf{M}\bar{\mathbf{v}} = \bar{\mathbf{e}}_i$:

$$\begin{bmatrix} \mathbf{I} & & & & \\ -\mathbf{A}/1 & \mathbf{I} & & & \\ & -\mathbf{A}/2 & \ddots & & \\ & & \ddots & \mathbf{I} & \\ & & & -\mathbf{A}/N & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{v}_0 \\ \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_N \end{bmatrix} = \begin{bmatrix} \mathbf{e}_i \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Now residual has index and block section: $r(i,j)$. Iteration reduces to:
(1) adding $r(i,j)^k$ to a single entry of $\mathbf{x}$, our approximation;
(2) adding scaled column, $\frac{r(i,j)^k}{j}\mathbf{A}(:,i)$, to section j of the residual.

Convergence and Implementation

Scaling $\frac{r(i,j)^k}{j}\mathbf{A}(:,i)$ guarantees $\|\mathbf{r}^k\|_1$ decreases, for column stochastic $\mathbf{A}$:

Convergence and Implementation

Scaling $\frac{r(i,j)^k}{j}\mathbf{A}(:,i)$ guarantees $\|\mathbf{r}^k\|_1$ decreases, for column stochastic $\mathbf{A}$:

$$\|\mathbf{r}^{k+1}\|_1 = \|\mathbf{r}^k\|_1 - r(i,j)^k + \frac{r(i,j)^k}{j} = \|\mathbf{r}^k\|_1 - r(i,j)^k(1-\tfrac{1}{j})$$

Largest entry, $r(i,j)$ is bounded below by average, $r(i,j) > \|\mathbf{r}\|_1/(\#$ non zeros in $\mathbf{r})$.

Convergence and Implementation

Scaling $\frac{r(i,j)^k}{j}\mathbf{A}(:,i)$ guarantees $\|\mathbf{r}^k\|_1$ decreases, for column stochastic $\mathbf{A}$:

$$\|\mathbf{r}^{k+1}\|_1 = \|\mathbf{r}^k\|_1 - r(i,j)^k + \frac{r(i,j)^k}{j} = \|\mathbf{r}^k\|_1 - r(i,j)^k(1-\tfrac{1}{j})$$

Largest entry, $r(i,j)$ is bounded below by average, $r(i,j) > \|\mathbf{r}\|_1/(\#$ non zeros in $\mathbf{r})$.

No component of large linear system formed explicitly:
- residual vector stored in a heap (alternative: queue with threshold)
- matrix $\mathbf{M}$ not formed at all
- blocks $\mathbf{v}_j$ not stored separately

"A Nearly Sublinear Approximation ... "

- Converges for stochastic matrices
- "Nearly sublinear" – if $d_{\max} = O(\log\log n)$ (unrealistic)
- In practice, sublinear if NNZ $= O(n)$
- Less work than a single mat-vec

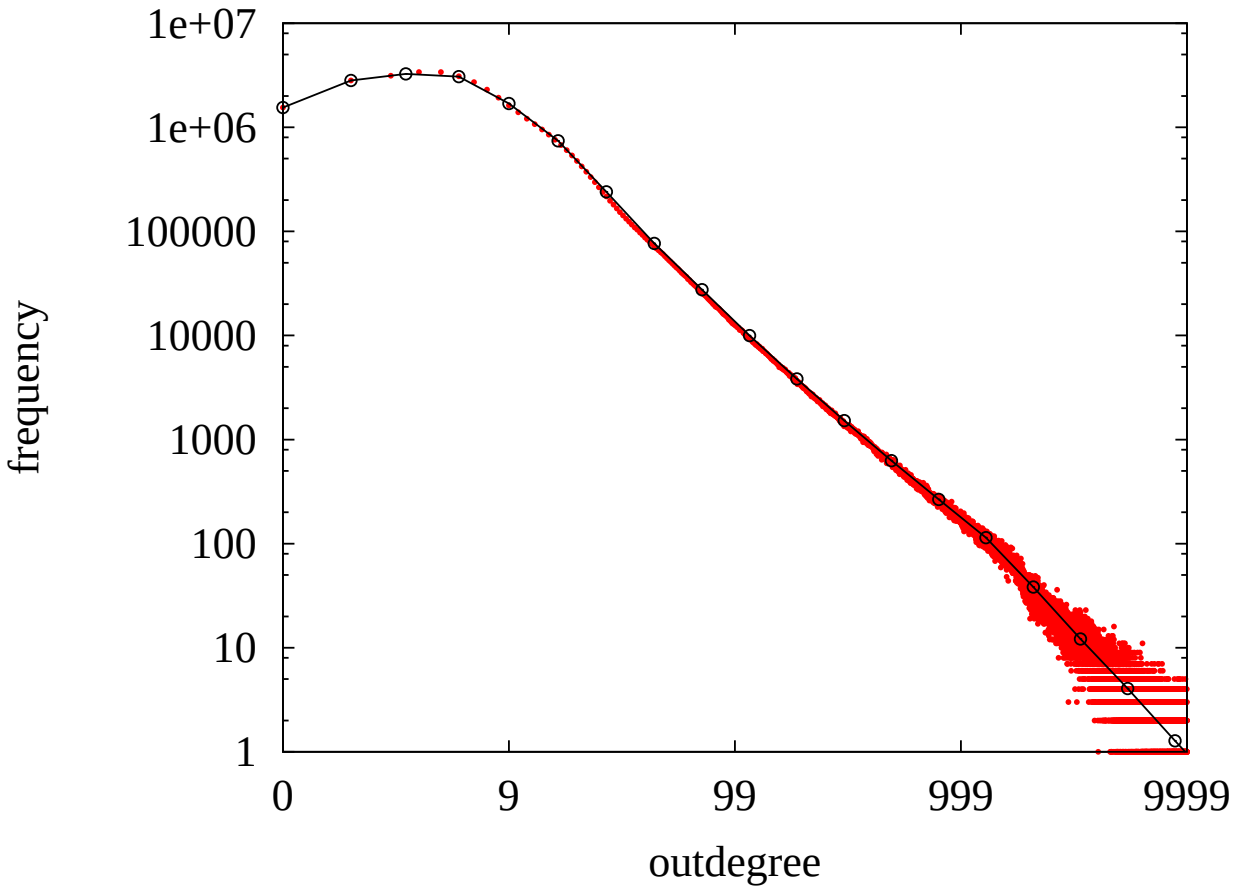
"A Nearly Sublinear Approximation ... "

- Converges for stochastic matrices
- "Nearly sublinear" – if $d_{\max} = O(\log\log n)$ (unrealistic)
- In practice, sublinear if NNZ $= O(n)$
- Less work than a single mat-vec
- New: for power-law degree distributed networks, the runtime for an error of ε is

$$\log(1/\varepsilon)\,(1/\varepsilon)^{3/2}\,d_{\max}\log(d_{\max})^2$$

- Social networks tend to have $d_{\max} = O(n^r)$ for $r < 1$, so this is sublinear in n.

Power-law degree distribution



[Laboratory for Web Algorithms, http://law.di.unimi.it/index.php]

Intuition for Proof

Our sublinear runtime proof depends on the degree distribution:

- decrease in $\|\mathbf{r}\|$ depends on largest value in $\mathbf{r}$, r_i
- lowerbound r_i using the average value of $\mathbf{r}$
- average value $= \|\mathbf{r}\|/(\#$ of nonzeros in $\mathbf{r})$

$(\#$ of nonzeros in $\mathbf{r})$ upper bounded by $d_{\max} * (\#iterations)$

Intuition for Proof

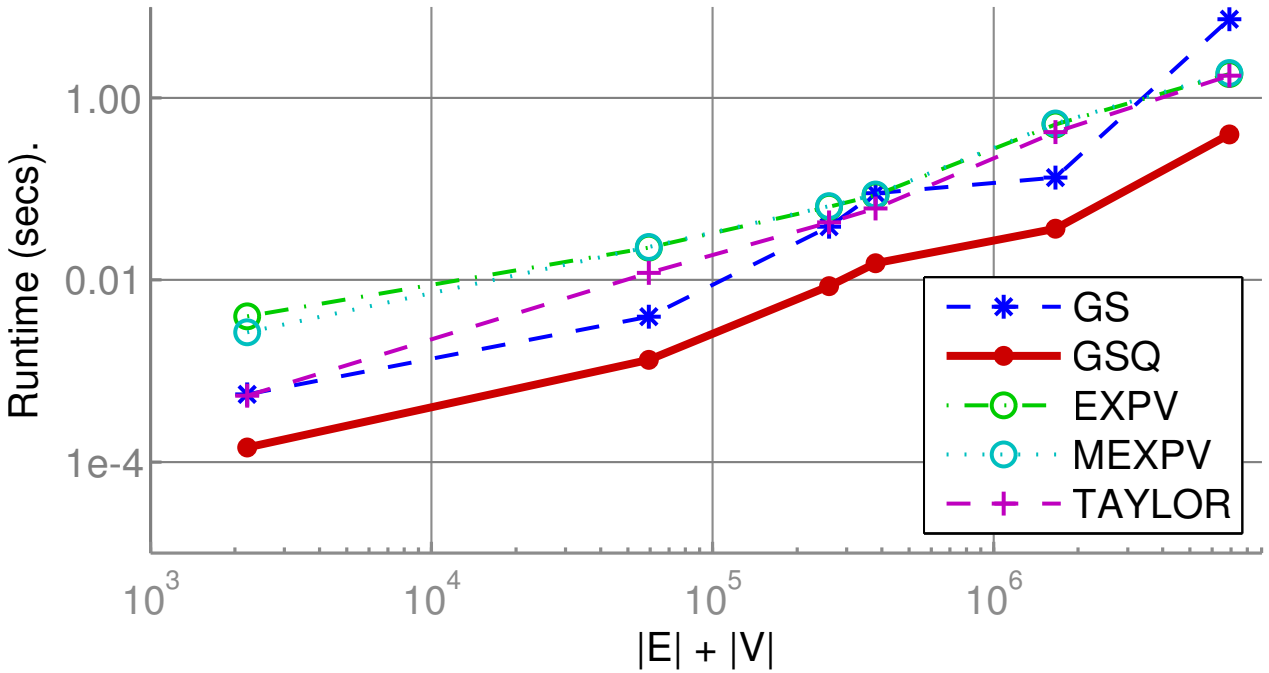
Our sublinear runtime proof depends on the degree distribution:

- decrease in $\|\mathbf{r}\|$ depends on largest value in $\mathbf{r}$, r_i
- lowerbound r_i using the average value of $\mathbf{r}$
- average value $= \|\mathbf{r}\|/(\#$ of nonzeros in $\mathbf{r})$

$(\#$ of nonzeros in $\mathbf{r})$ upper bounded by $d_{\max} * (\#iterations)$

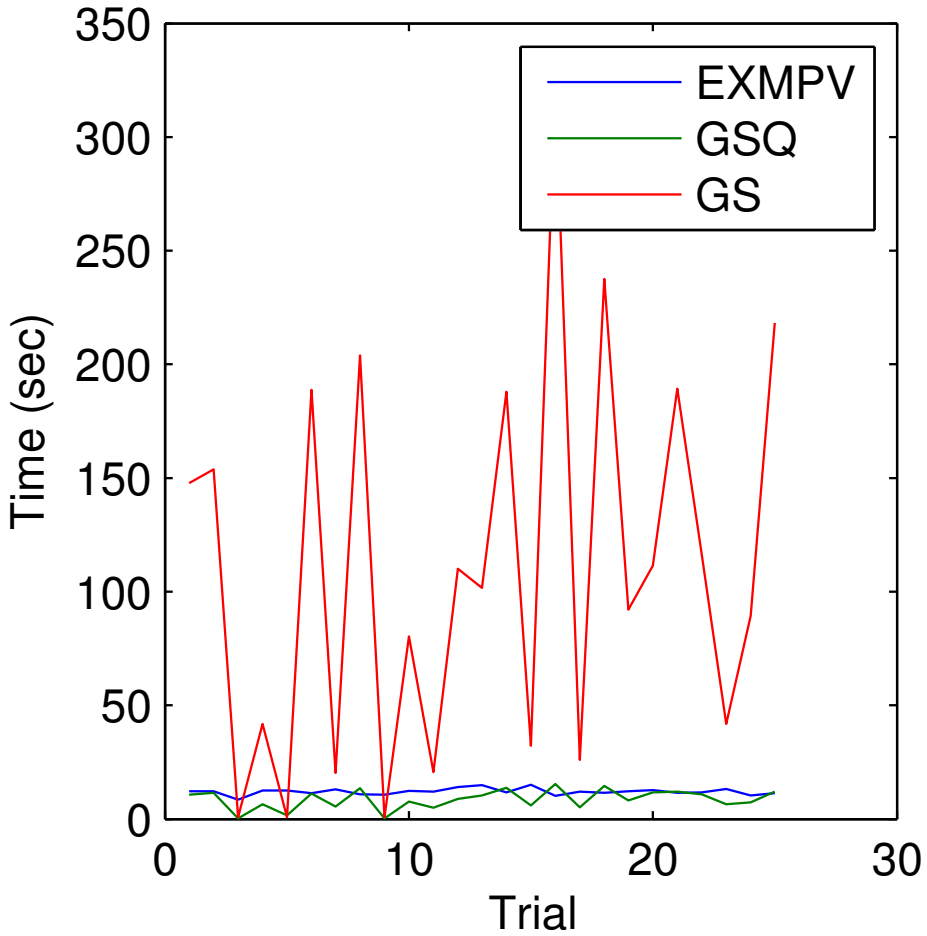
Power-law network: $\#$ of nonzeros in $\mathbf{r}$ after t iterations grows like $O(t)$ instead of $d_{\max} * t$. $\Rightarrow$ average value can't decay too fast. Hence, $\|\mathbf{r}\|$ is guaranteed to decrease "fast enough": $\|\mathbf{r}^t\| < O(t^{-2/3})$

Runtime v. Graph Size



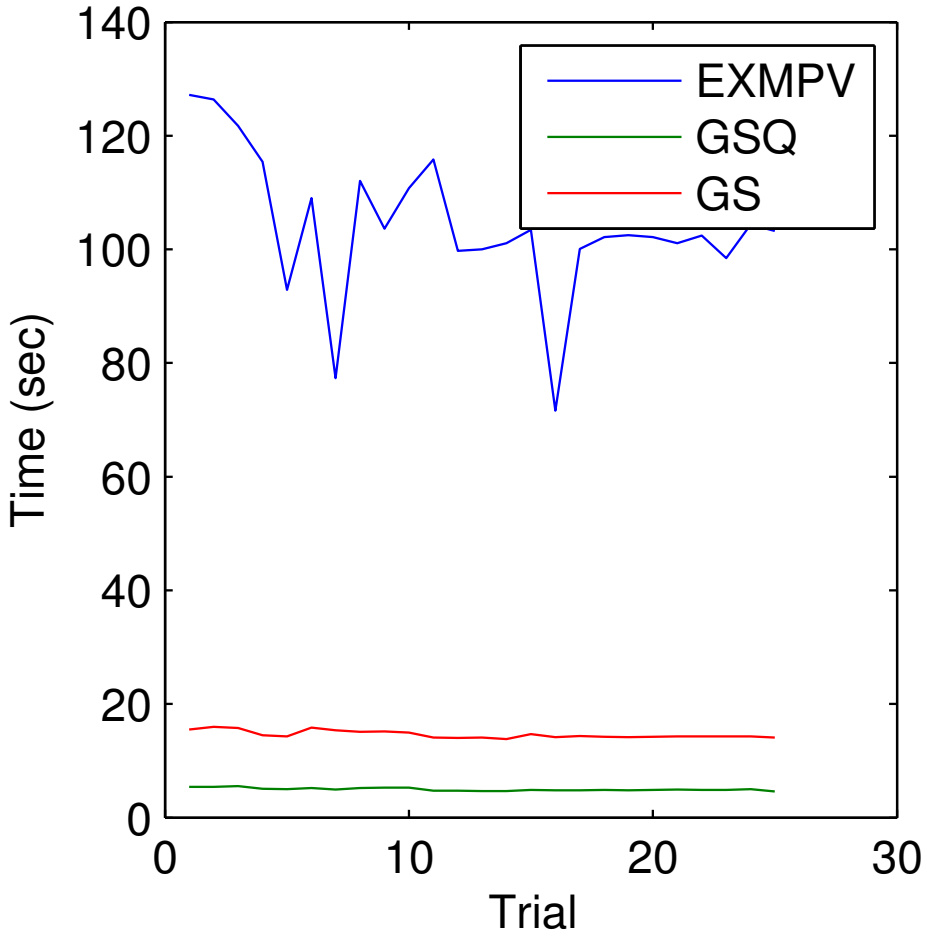
"GSQ" is a version of our Gauss-Southwell method that stores the residual vector in a queue instead of a heap.

Runtime on larger networks



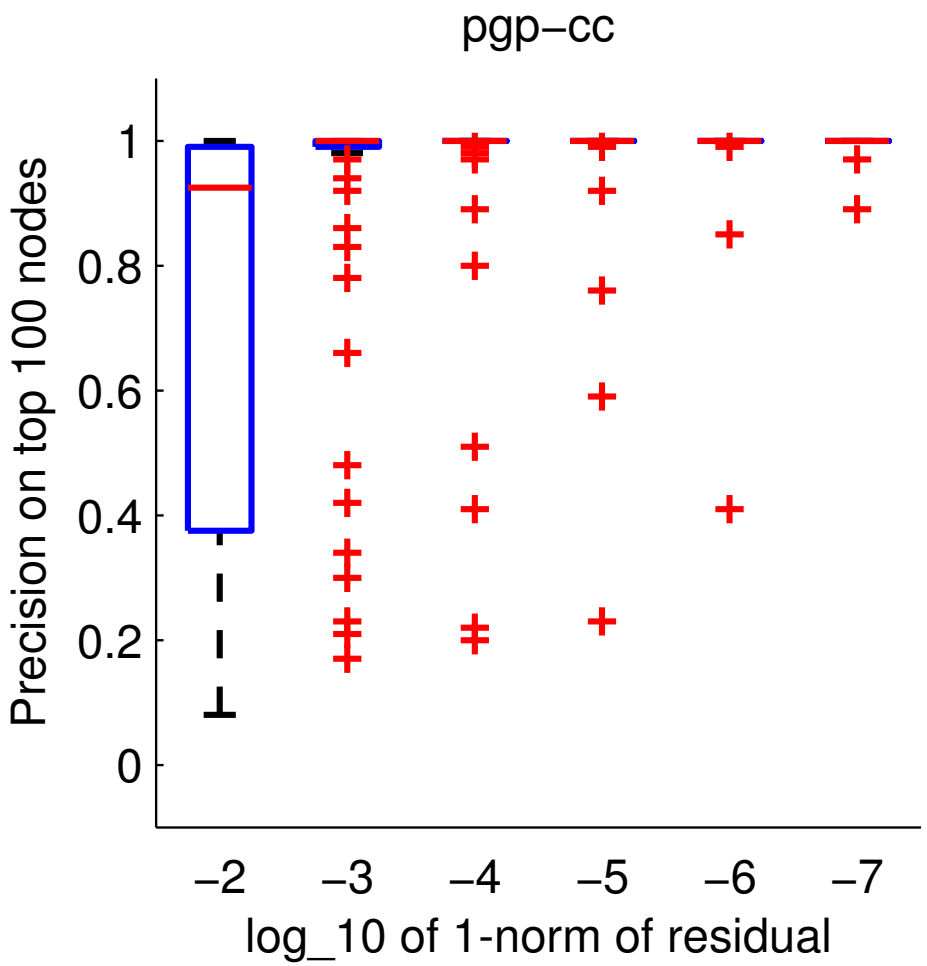
For `ljournal-2008`, $n = 5,363,260$, ave degree $= 14.7$.

Runtime on larger networks



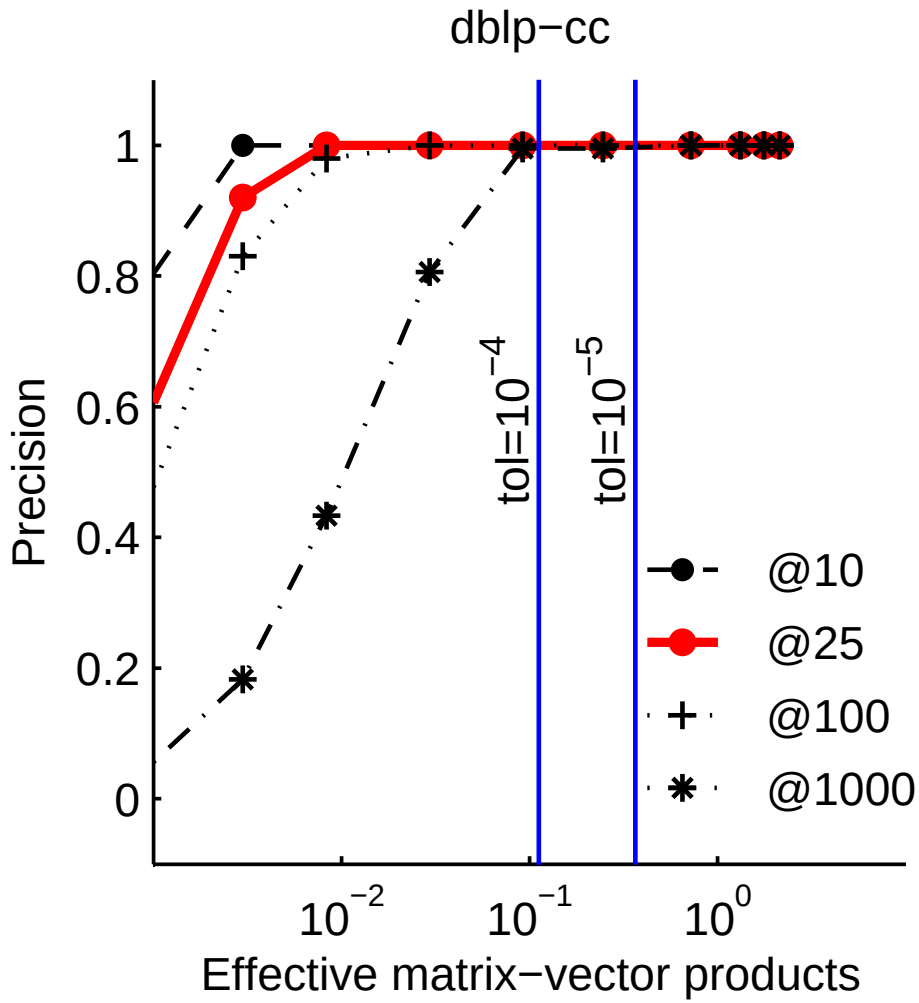
For `webbase-2001`, $n = 118,142,155$, ave degree $= 8.6$.

Accuracy of algorithm



For pgp-cc $n = 10,680$, but this is representative of dataset.

Number of operations performed



For `dblp-cc`, $n = 226,413$. Again, this is representative.

Future Work

- Adapt the method to other functions: $\cosh(x), x^{\frac{1}{p}}, \log(x)$.
- Allow for scaling, $f(t\mathbf{A})\mathbf{e}_i$.
- Allow for $f(\mathbf{A})$ times a vector $\mathbf{v}$ (other than $\mathbf{e}_i$).
- Improve domain of convergence, $\rho(\mathbf{A}) \in (0, 1]$.

Code and Further Details

Code available at

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

For details and references, see our paper at

`http://arxiv.org/abs/1310.3423`