A sub-linear method for computing columns of functions of sparse matrices

Kyle Kloster and David F. Gleich

Purdue University

March 3, 2014

Supported by NSF CAREER 1149756-CCF
Overview

1. $f(A)$: problem description and applications
2. Description of “sub-linear” results
3. The Algorithm for $f(A)b$
4. Intuition for proof
5. Experiments on real-world social networks
We can apply most functions, e.g. $f(x) = \cos(x)$, to any square matrices $A$ if $f$ is defined on the eigenvalues of $A$. One definition: Taylor series!

$$\cos(x) = \frac{1}{0!} + \frac{-x^2}{2!} + \frac{x^4}{4!} + \cdots$$

$$\cos(A) = \frac{1}{0!} + \frac{-A^2}{2!} + \frac{A^4}{4!} + \cdots$$

Then we can think of $f(A)b$ as the **action** of the operator $f(A)$ on $b$, or as a **diffusion** on a graph underlying the matrix $A$. 

Kyle Kloster (Purdue)
Functions of Matrices: applications

**Action:**
\[ f(x) = e^x: \quad \frac{dx}{dt} = Ax; \quad x(0) = x_0 \]
solution: \[ x(t) = \exp\{tA\}x_0 \]

\[ f(x) = x^{1/p}: \quad P(t) \text{ transition matrix for Markov process} \]
\[ P(1) \text{ describes process over a year; } P^{1/12} \text{ for a month} \]
Functions of Matrices: applications

**Action:**
\[ f(x) = e^x : \quad \frac{dx}{dt} = Ax; \quad x(0) = x_0 \]
solution: \[ x(t) = \exp\{ tA \} x_0 \]

\[ f(x) = x^{1/p} : \quad P(t) \text{ transition matrix for Markov process} \]
\[ P(1) \text{ describes process over a year; } P^{1/12} \text{ for a month} \]

**Diffusion:**
\[ f(x) = (1 - \alpha x)^{-1} : \quad \text{the resolvent yields the PageRank diffusion:} \]
\[ f(P)e_i \text{ interpreted as nodes' importance to node } i. \]

\[ f(x) = e^{tx} : \quad e^{tP}e_i, \text{ the heat kernel diffusion, offers} \]
\[ \text{an alternative ranking of nodes' importance} \]
Parameters of $f(A)b$

**A:**
- Original motivation: $A$ = a normalized version of an adjacency matrix from a social network; the Laplacian or random-walk matrix. Sparse, small diameter, stochastic, degree distribution follows power-law
- Generalized: any nonnegative $A$ with $\|A\|_1 \leq 1$.

**b:**
- Originally $b = e_i$, i.e. compute a column $f(A)e_i$
- Generalized: $b$ can be any sparse, stochastic vector

**$f(\cdot)$:**
- Originally $f(x) = e^x, (1 - \alpha x)^{-1}$
- Generalized: can be any function decaying “fast enough”
Columns of the Matrix Exponential

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

\[
\exp \{ A \} = \sum_{k=0}^{\infty} \frac{1}{k!} A^k
\]

- \((A^k)_{ij}\) gives the number of length-\(k\) walks from \(i\) to \(j\), so...
- Large entries of \(\exp \{ A \}\) denote "important" nodes / links
- Used for link-prediction, node ranking, clustering
The Problem

Columns of the Matrix Exponential

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

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

- \((\mathbf{A}^k)_{ij}\) gives the number of length-\(k\) walks from \(i\) to \(j\), so...
- Large entries of \(\exp\{\mathbf{A}\}\) denote “important” nodes / links
- Used for link-prediction, node ranking, clustering
- \(\exp\{\mathbf{A}\}\) is common, but other \(f(\mathbf{A})\) can be used:
- PageRank can be defined from the resolvent:

\[ (\mathbf{I} - \alpha \mathbf{A})^{-1} = \sum_{k=0}^{\infty} \alpha^k \mathbf{A}^k \]

\[ \rightarrow \text{replace} \frac{1}{k!} \text{ with other coefficients?} \]
The Problem

$f(A)$ as weighted sum of walks

For $f(A) = e^{tA}$ and $f(A) = (1 - \alpha A)^{-1}$, how are walks weighted?

$$f(A)b = (f_0 I + f_1 A + f_2 A^2 + f_3 A^3 + \cdots) b$$
Big Graphs from Social Networks

We’ve seen the computation ($f$); what does the domain of inputs look like?

- Social networks like Twitter, YouTube, Friendster, Livejournal
- Large: $n = 10^6, 10^7, 10^9 +$
- Sparse: $|E| = O(n)$, often $\leq 50n$
- Difficulty: “small world” property: diameter $\approx 4$ (!)
- Helpful: Power-law degree distribution (picture)
Power-law degree distribution

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

- Leading methods for $f(A)b$ 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.
The Problem

Fill-in from repeated matvecs

Vectors $\mathbf{P}^k \mathbf{e}_i$ for $k = 1, 2, 3, 4$. $n = 1133$
$f(P)e_i$ is a localized vector

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

the column of $\exp\{P\}$ produced by previous slide’s matvecs
New method: avoid mat-vecs! → 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.
Main Result 1

**Theorem 1:** [action of $f$ on $b$]

Given nonnegative $A$ satisfying $\|A\|_1 \leq 1$, with power-law degree distribution and max degree $d$; and sparse stochastic $b$; our method computes $x \approx f(A)b$ such that

$$\|f(A)b - x\|_1 < \varepsilon \text{ in work } (\varepsilon) = O \left( (1/\varepsilon)^{C_f} \log(1/\varepsilon) d^2 \log(d)^2 \right),$$

“work” “scales as” $d^2 \log(d)^2$ in the graph size

for any function $f$ that decays “fast enough”. The constant $C_f$ depends on how quickly the Taylor coefficients of $f$ decay.
Main Result 1

**Theorem 1:** [Action of \( f \) on \( b \)]

Given nonnegative \( A \) satisfying \( \|A\|_1 \leq 1 \), with power-law degree distribution and max degree \( d \); and sparse stochastic \( b \); our method computes \( x \approx f(A)b \) such that

\[
\|f(A)b - x\|_1 < \varepsilon \quad \text{in work} \quad (\varepsilon) = O\left(\left(\frac{1}{\varepsilon}\right)^{C_f} \log(1/\varepsilon) d^2 \log(d)^2\right),
\]

“work” “scales as” \( d^2 \log(d)^2 \) in the graph size for any function \( f \) that decays “fast enough”. The constant \( C_f \) depends on how quickly the Taylor coefficients of \( f \) decay.

For \( f(x) = (1 - \alpha x)^{-1} \), \( C_f = \frac{1}{1 - \alpha} \) (Note: \( \alpha \in (0, 1) \)).

For \( f(x) = e^x \), \( C_f = \frac{3}{2} \).

For \( f(x) = x^{1/p} \), \( C_f = \frac{3p}{5p-1} \) (Note: \( p \in (0, 1) \)).
Main Result 2

Theorem 2: [diffusion of \( f \) across a graph]
Given column stochastic \( A \) and \( b \), \( \tilde{x} \approx \tilde{f}(tA)b \) can be computed such that

\[
\| \tilde{f}(P)b - \tilde{x} \|_\infty < \varepsilon \text{ in work } (\varepsilon) = O \left( \frac{2f(t)}{\varepsilon} \right),
\]

(Remark: the ‘tilde’ denotes a degree-normalized version for the diffusion: \( D^{-1}\exp\{tP\}b \), for example. We normalize by degrees to adjust for the influence of the stationary distribution of \( P \).)

Corollary: \( f(A)b \) is a local vector.

Proof: Because sublinear work is done, \( f(A)b \) cannot have \( O(n) \) nonzeros.
Outline of Nexpokit method (our second method, hk-relax, is related)

1. Express $f(A)b$ via a Taylor polynomial
2. Form large linear system out of Taylor terms
3. Use sparse solver to approximate each term’s largest entries
4. Combine approximated terms into a solution
In terms of Taylor terms

Taylor polynomial:

\[ f(A)b \approx \left( f_0 I + f_1 A + f_2 A^2 + f_3 A^3 + \cdots + f_N A^N \right) b \]

Compute terms recursively:

\[ v_k = f_k A^k e_i = \frac{f_k}{f_{k-1}} A \left( f_{k-1} A^{k-1} \right) e_i \]

\[ v_k = \frac{f_k}{f_{k-1}} A v_{k-1} \]

Then \( f(A)b \approx v_0 + v_1 + \cdots + v_{N-1} + v_N \)

(But we want to avoid computing \( v_j \) in full...)
Forming a linear system

So we convert the Taylor polynomial into a linear system. For simplicity’s sake, we use the example of $\exp\{A\}e_i$ here.
Our method: Nexpokit

Forming a linear system

So we convert the Taylor polynomial into a linear system. For simplicity’s sake, we use the example of $\exp\{A\}e_i$ here.

\[
\begin{bmatrix}
    I & & & & \\
    -A/1 & I & & & \\
    & -A/2 & \ddots & & \\
    & & \ddots & I & \\
    & & & -A/N & I
\end{bmatrix}
\begin{bmatrix}
    v_0 \\
    v_1 \\
    v_2 \\
    \vdots \\
    v_N
\end{bmatrix}
= 
\begin{bmatrix}
    e_i \\
    0 \\
    0 \\
    \vdots \\
    0
\end{bmatrix}
\]

where we use the identity $v_k = \frac{1}{k}Av_{k-1}$ (which comes from $v_k = \frac{f_k}{f_{k-1}}Av_{k-1}$, since $f_k = \frac{1}{k!}$, so $f_k/f_{k-1} = \frac{(k-1)!}{k!} = \frac{1}{k}$).

Then $\exp\{A\}e_i \approx v_0 + v_1 + \cdots + v_{N-1} + v_N$
Sparse solver: Gauss Southwell

Basic idea of Gauss Southwell (GS): solving $Mx = b$ when $x$ is “effectively sparse” (i.e. a localized vector)

1. Set $x^{(0)} = 0$, $r^{(0)} = b$, then iterate:
2. At step $k$, relax maximal entry of $r^{(k)}$ (denoted $m^{(k)}$), add to $x^{(k)}$;
   \[
   x^{(k+1)} = x^{(k)} + m^{(k)} \cdot e_i
   \]
3. Add corresponding column of $M$ to residual:
   \[
   r^{(k+1)} = r^{(k)} - m^{(k)} \cdot M(:, i)
   \]
Apply GS to our linear system, $M\tilde{v} = \bar{e}_i$:

$$
\begin{bmatrix}
    r_0 \\
    r_1 \\
    r_2 \\
    \vdots \\
    r_N
\end{bmatrix}
= 
\begin{bmatrix}
    e_i \\
    0 \\
    0 \\
    \vdots \\
    0
\end{bmatrix}
- 
\begin{bmatrix}
    I & & & \\
    -A/1 & I & & \\
    & -A/2 & \ddots & \\
    & & \ddots & I \\
    & & & -A/N & I
\end{bmatrix}
\begin{bmatrix}
    v_0 \\
    v_1 \\
    v_2 \\
    \vdots \\
    v_N
\end{bmatrix}
$$

The update can be simplified to a block-wise update:

$$
\textbf{r}^{(k+1)} = (\textbf{r}^{(k)} - \textbf{m}^{(k)} \otimes \textbf{e}_j) + \frac{\textbf{m}^{(k)}}{j+1} \cdot \textbf{A}(\cdot, i)
$$

No component of large linear system formed explicitly:
- residual vector stored in a heap (alternative: queue with threshold)
- matrix $M$ not formed at all
- blocks $\textbf{v}_j$ not stored separately, stored as one solution vector $\textbf{x} = \sum \textbf{v}_j$. 
Outline of proof

Initial residual is $r = e_i$, has $\|r^{(0)}\|_1 = 1$, and it decreases at each step. We show that

1. decay of $\|r^{(k)}\|_1$ depends on its max value $m^{(k)}$
2. max value $m^{(k)}$ is bounded below by average value of $r$
3. average value of $r$ depends on $\#$ nonzeros in $r$
4. growth of $\#\text{nnz}(r)$ depends on degree distribution
5. Power-law degree distribution implies $\#\text{nnz}(r)$ grows slowly, so
6. $\|r\|_1 \to 0$ at a certain minimum speed!
Proof

Decay of $\|r\|_1$

Residual $r = [r_0; r_1; \cdots; r_N]$ has index and block section: $r(i,j)$. For our special linear system, the GS residual reduces to: during step $k$, do

1. delete $r(i,j)^{(k)}$ in $r$ and add it to $x_i$, our approximation;
2. add scaled column, $\frac{m^{(k)}}{j}A(:, i)$, to section $j$ of the residual.

Taking the 1-norm of (1) shows

$$\|r^{(k+1)}\|_1 \leq \|r^{(k)}\|_1 - m^{(k)}(1 - \frac{1}{j})$$

Note the $(1 - \frac{1}{j})$ factor appears because we’re looking specifically at $e^x$. For the resolvent, $f(x) = (1 - \alpha x)^{-1}$, this factor would be $(1 - \alpha)$ instead.
Proof

Number of nonzeros

Largest entry, \( m^{(k)} = r(i,j) \) is bounded below by average value of the residual,

\[
m^{(k)} = r(i,j) > \|r\|_1/(\# \text{ non zeros in } r)
\]

But we can bound \( \text{nnz}(r) := (\# \text{ of nonzeros in } r) \) based on the degree of the column of \( A \) that we add to the residual each step.
Number of nonzeros

Largest entry, $m^{(k)} = r(i,j)$ is bounded below by average value of the residual,

$$m^{(k)} = r(i,j) > \|r\|_1/(\text{# non zeros in } r)$$

But we can bound $\text{nnz}(r):= (\text{# of nonzeros in } r)$ based on the degree of the column of $A$ that we add to the residual each step.

Each iteration we can add no more nonzeros to $r$ than the largest degree among all unvisited nodes.

Usually the best we can say is that this is upper bounded by $d := d_{\text{max}} \times (\text{# iterations})$, because it’s possible every node has max degree.

But with the power-law assumption ...
Number of nonzeros

Largest entry, $m^{(k)} = r(i,j)$ is bounded below by average value of the residual,

$$m^{(k)} = r(i,j) > \|r\|_1/(\# \text{ non zeros in } r)$$

But we can bound $\text{nnz}(r) := (\# \text{ of nonzeros in } r)$ based on the degree of the column of $A$ that we add to the residual each step.

Each iteration we can add no more nonzeros to $r$ than the largest degree among all unvisited nodes.

Usually the best we can say is that this is upper bounded by $d := d_{\text{max}} \ast (\#\text{iterations})$, because it’s possible every node has max degree.

But with the power-law assumption ...
Proof

Power-law degree distribution

With power-law assumption, we know that the $t^{th}$ largest degree, $d_t$, is bounded by $d_t \leq Cd \cdot t^{-\beta}$ for some $\beta$ near 1 and some constant $C$.

After $k$ iterations, $\text{nnz}(r)$ is bounded by the sum of the degrees of the new vertices visited in those $k$ iterations. By step $k$, this is at most $\text{nnz}(r) \leq \sum_{t=1}^{k} d_t$, so

$$\text{nnz}(r) \leq \sum_{t=1}^{k} d_t \leq \sum_{t=1}^{k} Cd \cdot t^{-1}$$

In fact, after the first $d$ iterations, $d_t$ is just a small constant, $c$. Then this sum grows no faster than $\sum_{t=1}^{k} d \cdot t^{-1} \leq d \log(d) + c \cdot t$. So $\text{nnz}(r)$ grows like $t \cdot c$ for $c \approx 1$ instead of $t \cdot d$ (!).
Convergence

We had

$$\|r^{(k+1)}\|_1 \leq \|r^{(k)}\|_1 - m^{(k)}(1 - \frac{1}{j})$$

The power-law assumption allows the bound

$$-m^{(k)} \leq -\frac{\|r^{(k)}\|_1}{C_2 + c \cdot k}.$$

$$\|r^{(k+1)}\|_1 \leq \|r^{(k)}\|_1 \left(1 - \frac{2/3}{C_2 + c \cdot k}\right)$$

$$\leq \|r^{(k)}\|_1 \exp\left\{-\frac{2}{3} \frac{1}{C_2 + c \cdot k}\right\}$$

$$\leq \|r^{(0)}\|_1 \exp\left\{-\frac{2}{3} \sum_{t=0}^{k} \frac{1}{C_2 + c \cdot t}\right\}$$

$$\leq \exp\left\{-\frac{2}{3} \log(k + C)\right\}$$

$$\|r^{(k+1)}\|_1 \leq (k + C)^{-2/3}$$

(See the paper cited at the end for a precise completion of the proof).
“GSQ” is a version of our Gauss-Southwell method that stores the residual vector in a queue instead of a heap.
Experimental Results

Runtime on larger networks

For webbase-2001, $n = 118, 142, 155$, ave degree $= 8.6$. 
Code available at

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

For details and references, see our paper at