Asynchronous Parallel Solution of Markov Chains: Application to PageRank

Daniel B. Szyld
Temple University, Philadelphia

Joint work with:
Efstratios Gallopoulos and Giorgos Kollias
University of Patras

A A Markov Anniversary Conference
13 June 2006

Brief Outline

- review of block power method (setting up notation)
- review of parallel asynchronous methods
- asynchronous block power method
- some numerical results

Review - Notation - Power Method

Aim is to find the transition probability (row) vector π for the stochastic transition probability $n \times n$ matrix P, i.e., satisfying

$$\pi P = \pi, \qquad \pi \geq 0, \quad \sum \pi_i = 1.$$

Rewrite $\pi P = \pi$ as

$$Ax = (I - B)x = 0 \quad \text{or} \quad Bx = x$$

with $x = \pi^T$, $B = P^T$, $B \geq 0$, $B^T e = Pe = e$, $e^T = [1, \ldots, 1]$.
A is a (singular $n \times n$) M-matrix.

Review - Notation - Power Method

Classic power method:

For $k = 0, 1, \ldots$

$$y^{k+1} = Bx^k$$

$$x^{k+1} = y^{k+1}/e^T y^{k+1}$$

In the classic power method normalization is performed
so that the entries in the iterates do not grow unbounded
(other normalizations are also used).

Convergence rate is $|\lambda_2|/|\lambda_1|$.
In our case: $\lambda_1 = 1$, and let
$\gamma = \gamma(B) = \max\{|\lambda|,\ \lambda \in \sigma(B),\ \lambda \neq 1\}$.

This is the asymptotic convergence rate of the power iteration
(sometimes called "the second eigenvalue")

Review - Notation - Parallel Power Method

Let $R_i = [I_i|0]P_i$ $(n_i \times n)$,

P_i a permutation matrix,

i.e., the rows of R_i are rows of the $n \times n$ identity matrix I, e.g.,

$$R_i = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Let $RB_i = R_i \cdot B$ be an $n_i \times n$

block of rows of B, $i = 1, \ldots, p$. $\sum_{i=1}^{p} n_i = n.$

Partition x_k and y_k accordingly.

Classic parallel (synchronous) block power algorithm:

Let $x : x^0$

For $k = 0, 1, \ldots$

In each processor $i = 1, \ldots, p$:

Compute $y_i = RB_i x$

In master processor:

Wait for all p processes to complete. Then:

Assemble $y^T = [y_1^T, \ldots, y_p^T]$

$x^{k+1} = y/y^T e$

Send $x : x^{k+1}$ to all processors

Google matrix

- Q represents "random crawler" on the web.
- Normalize: get P so that $P^T e = e$.
- $G = \alpha P + (1-\alpha) v e^t > O, \quad v > 0, \quad v^T e = 1;$
- $G \geq O, \quad G^T e = e.$
- $\rho(G) = 1, \quad \gamma(G) = \alpha \quad (=.85$ in Google)
- $A = I - G^T$ primitive singular M-matrix
- Solution of $Gx = x$, $x^T e = 1$ is PageRank [Brin and Page, 1998]

Observation

If we do not normalize in the power method, we have standard iteration with splitting $A = I - G^T$, G convergent ($\lim G^k$ exists) .

Here no danger of overflow!!

Let $x > 0, \quad x^T e = 1$, then $\quad (Gx)^T e = x^T G^T e = x^T e = 1$.

Thus, in our experiments, we only normalize at the end, once we have achieved convergence.

Idea of Parallel Asynchronous Power Method

Iterate number has a different meaning: models with delays

[Chazan and Miranker, 1969], [Lubachevsky and Mitra, 1996],

[Bertsekas and Tsitsiklis, 1989], and many authors:

Bahi, Bahia, Bru, El Baz, Elsner, El Tarazi, Frommer, Kaszkurewicz,

Miellou, Migallón, Neumann, Penadés, Spiteri, ...

Let $x := x^0$ In each processor $i = 1, \ldots, p$:

Compute $y_i = RB_i x$,

Send y_i to all other processors,

Do not wait for all p processes to complete.

Assemble $x^T := [y_1^T, \ldots, y_p^T]$

with whatever information is available, and repeat.

Mathematical model of Asynchronous Iterations

- Synchronous

  $x_i \leftarrow f_i(x),\ i = 1, \ldots, n$

  $x_i(t)$ = value of ith component at time t

- Asynchronous

  $T = \{0, 1, 2, \ldots\}$ = set of times at which some x_i is updated

  T^i = set of times at which x_i is updated

  $x_i(t+1) \leftarrow f_i(x_1(\tau_1^i(t)), \ldots, x_n(\tau_n^i(t)))$, if $t \in T^i$

  $x_i(t+1) \leftarrow x_i(t)$, otherwise

where $delay = t - \tau_j^i(t) \geq 0,\ \forall t \in T$

Convergence of Asynchronous Iterations

- **Basic result**

  Let $\{X(k)\} : \ \ldots \subset X(k+1) \subset X(k) \subset \ldots \subset X$ with properties:

  - synchronous convergence condition

    $\forall k,\, x \in X(k) : f(x) \in X(k+1)$

    $\{y^k\},\, y^k \in X(k)$: limit points of $\{y^k\}$ are fixed points of f

  - box condition

    $\forall k : \ X(k) = X_1(k) \times \ldots \times X_n(k)$

  then $x(0) \in X(0)$: limit points of $\{x(t)\}$ are fixed points of f

- **PageRank result**

  under non-restrictive conditions:

  for $t \to \infty,\ x(t) \to cx^*$

  where $G\, x^* = x^*$, c finite constant

  (Lubachevsky, Mitra (1986))

What is new here? Why now?

- Large size of problem, as never before
- Clusters of machines more common
- Internet potential vehicle for asynchronous computation
- Realization that power method (here) needs no normalization
- Thus: existing theory applies

Numerical experiment

Stanford-Berkeley matrix [Kamvar, Haveliwala, Manning, Golub, 2003]

$n = 281,903$. Homogeneous processor architecture.

Local convergence tolerance (in each processor): 10^{-6}

CPU times for parallel computations on Beowulf cluster of PCs

	Synchronous		**Asynchronous**		
pr	it	t **(sec)**	$[it_{min}, it_{max}]$	$[t_{min}, t_{max}]$ **(sec)**	$speedUp$
2	44	179.2	[68, 69]	[86.3, 94.5]	1.98
4	44	331.4	[82, 111]	[139.2, 153.1]	2.27
6	44	402.8	129, 148]	[141.7, 160.6]	2.66

Observations

- Computations are very fast
- Communication is bottleneck
- Main use of parallel machines here is to fit all (distributed) data (kept sparse).
- No scalability is expected.

Conclusions and comments

- Asynchronous Power method, OK with no normalization
- Good convergence theory
- Preliminary good timings
- One can include accelerations suggested by others, e.g.,
  [Kamvar, Haveliwala, Manning, Golub, 2003]
  [Langville, Meyer, 2005] [Kirkland, Ipsen, 2005]
  [several Talks here]

Current and planned work:

- Smaller blocks within same processor
- Stopping criteria following Bahi et al.
- How to choose blocks?

How to choose blocks?

- Currently, we just choose blocks by dividing variables by number of processors and taking blocks consecutively
- We can use graph-based permutation algorithms, e.g., TPABLO [Choi and S., 1996] designed for NCD Markov chains
- Google is not NCD, but ....

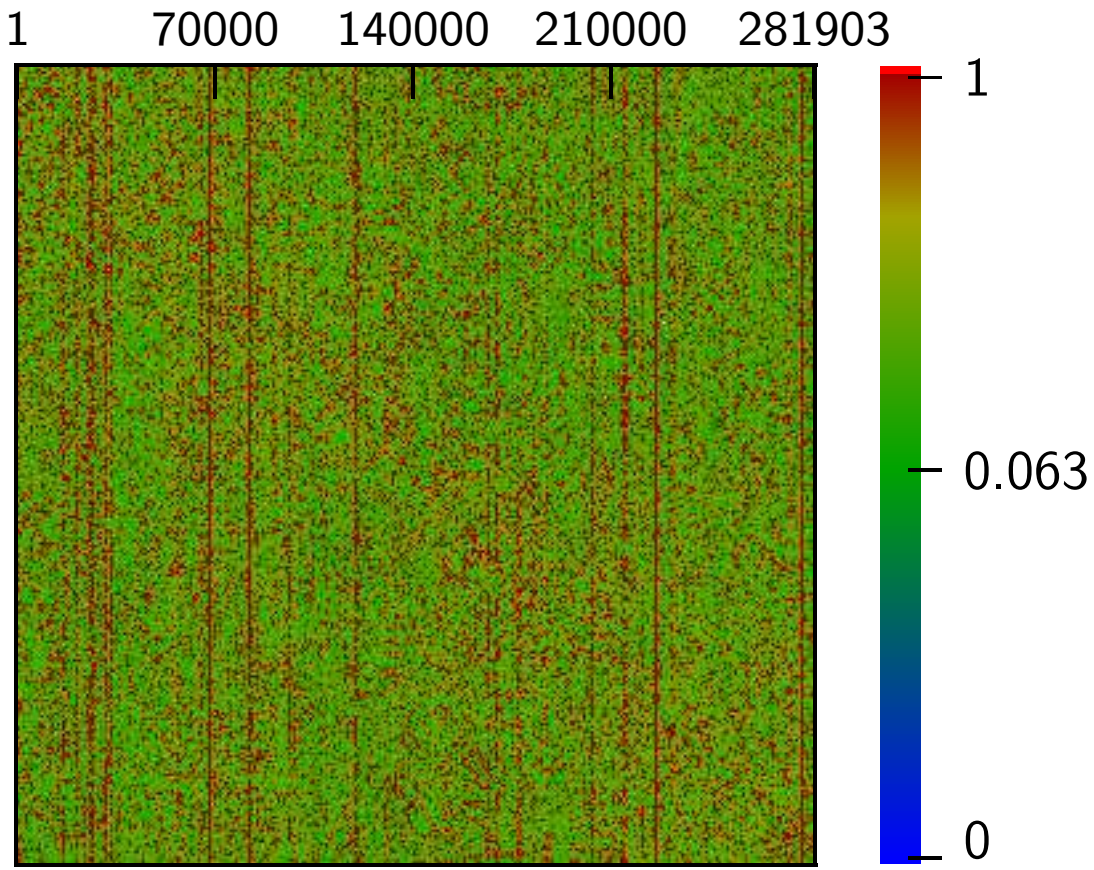

Figure 1: Spy plot of matrix STANFORD

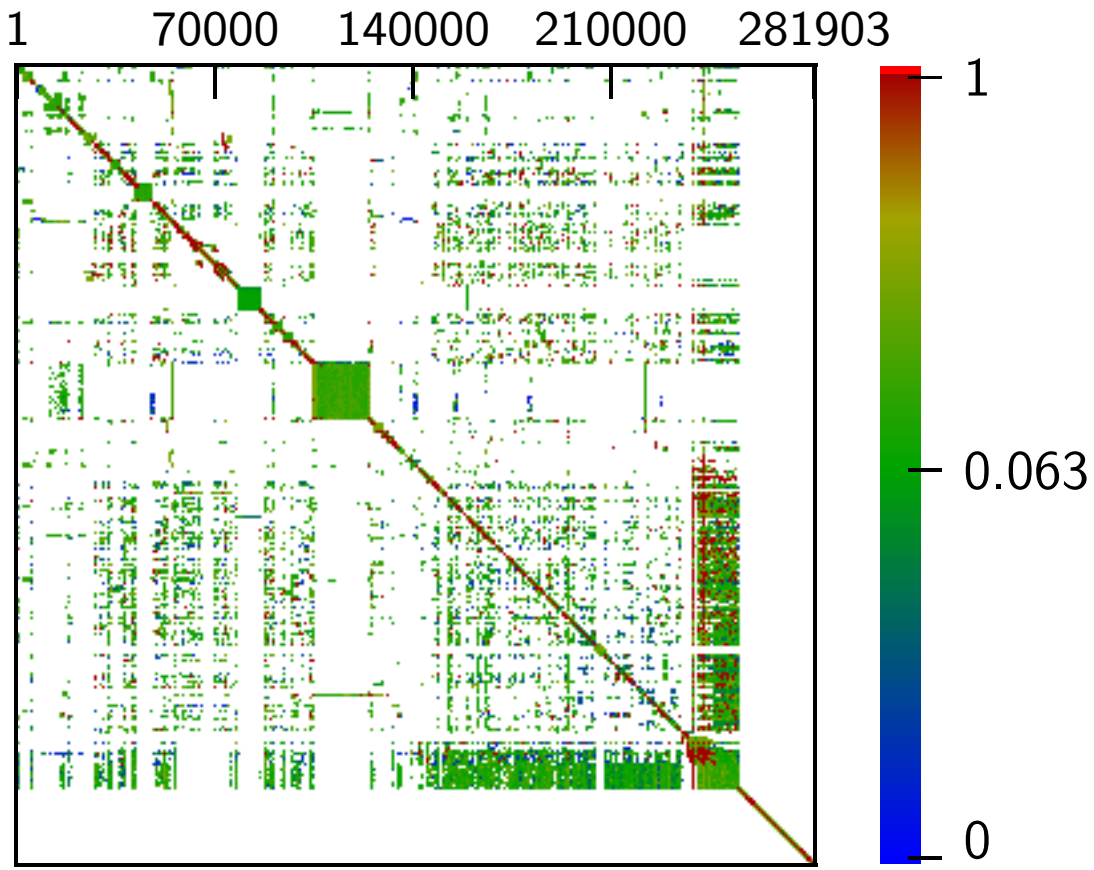

Figure 2: Spy plot of matrix STANFORD-P1000

Main idea of X/T/PABLO

- Graph based. Symmetric permutation.
- Goal: Dense block along the diagonal
  (and small entries in off-diagonal)
- Blocks generated one at a time.
  New node is added if it satisfies certain connectivity criteria.
- Linear complexity depending on n and nnz.