Hypergraph-based

Parallel Computation of Passage Times in Large Semi-Markov Models

Jeremy T. Bradley Nicholas J. Dingle

William J. Knottenbelt Helen J. Wilson

Department of Computing
Imperial College London
United Kingdom

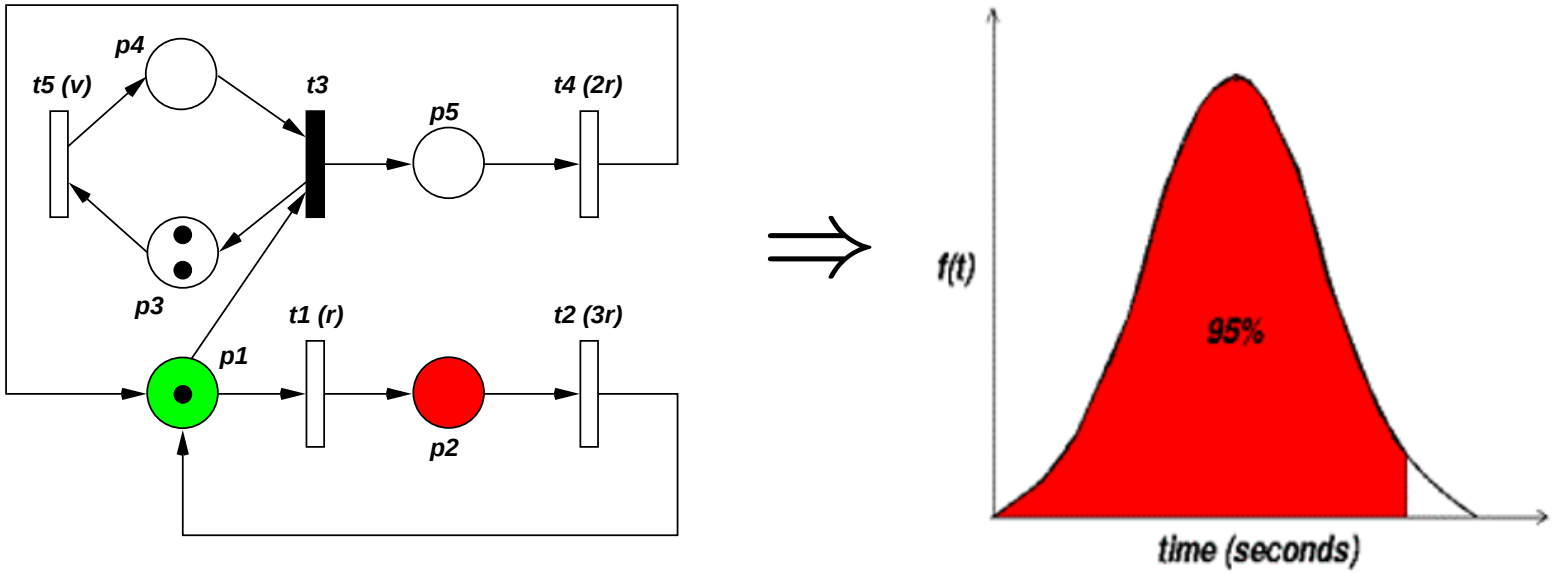
Department of Applied Mathematics
University of Leeds
United Kingdom

Email: {jb,njd200,wjk}@doc.ic.ac.uk, h.j.wilson@leeds.ac.uk

Motivation

- Complex concurrent systems are often modelled as Markov and semi-Markov chains
- Traditional steady-state analysis can compute performance measures such as:
  - system availability/throughput
- Require techniques to compute response-time measures for:
  - QoS metrics in benchmarks/Service Level Agreements

Aims



Semi-Markov Processes

- A more expressive generalisation of Markov processes
- An N-state SMP is characterised by:
  - an $N \times N$ one-step probability matrix P
  - an $N \times N$ matrix H where $h_{ij}(t)$ is the distribution function of the sojourn time of the process in state i, given that its going to state j
- We define the kernel $R(i, j, t)$ of an SMP as:

$$R(i, j, t) = p_{ij} H_{ij}(t)$$

First Passage Times

- First passage time from state i to target states $\vec{j}$ in an SMP:

$$P_{i\vec{j}} = \inf\{u > 0 : Z(u) \in \vec{j}, N(u) > 0, Z(0) = i\}$$

  - $Z(t)$ is the state of the SMP at time t
  - $N(u)$ is the number of state transitions by time u

- Laplace transform of $P_{i\vec{j}}$ density can be computed as:

$$L_{i\vec{j}}(s) = \sum_{k \notin \vec{j}} r^*_{ik}(s) L_{k\vec{j}}(s) + \sum_{k \in \vec{j}} r^*_{ik}(s)$$

  where $r^*_{ik}(s)$ is the LST of $R(i, k, t)$

Iterative Algorithm I

- Generates successively better approximations to the passage time quantity $P_{i\vec{j}}$
- $P^{(r)}_{i\vec{j}}$ is the conditional passage time for the system to reach any state in $\vec{j}$ in r transitions:

$$P^{(r)}_{i\vec{j}} = \inf\{u > 0 : Z(u) \in \vec{j}, 0 < N(u) \leq r, Z(0) = i\}$$
$$P^{(r)}_{i\vec{j}} \to P_{i\vec{j}} \text{ as } r \to \infty$$

- The second property ensures convergence of the algorithm

Iterative Algorithm II

- Calculate Laplace transform of rth iteration, $P_{i\vec{j}}^{(r)}$:

$$L_{i\vec{j}}^{(r)}(s) = \sum_{m=1}^{r} \text{contribution from } m\text{th transition matrix}$$

$$= \sum_{m=1}^{r} UU'^{m-1}\ \tilde{e}_{\vec{j}}$$

where matrix U has elements $u_{pq} = r_{pq}^{*}(s)$ and U' has states in $\vec{j}$ made absorbing

Iterative Algorithm III

- $L^{(r)}_{\vec{i}\vec{j}}(s)$ can be generalised to multiple source states, $\vec{i}$, and calculated in one go:

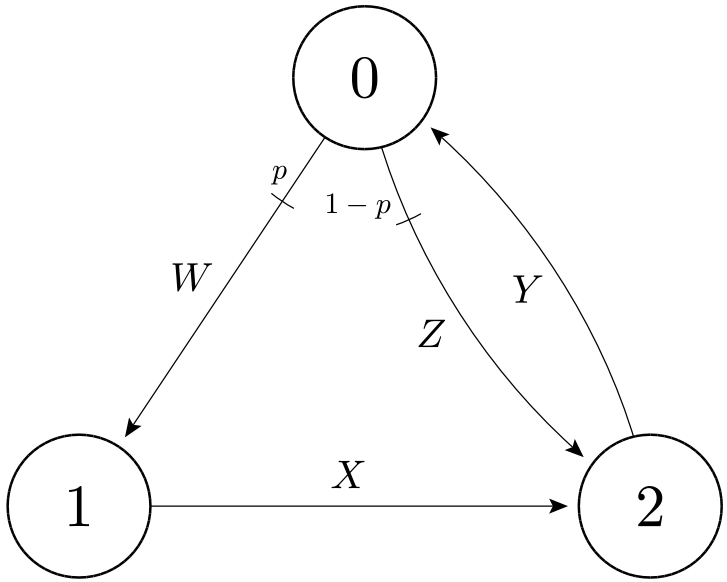
$$L^{(r)}_{\vec{i}\vec{j}}(s) = \sum_{k=0}^{r-1} \tilde{\alpha} U U'^{k} \; \tilde{e}_{\vec{j}}$$

  where $\tilde{\alpha}$ is a vector of weights for the source states in $\vec{i}$, e.g.:

$$\alpha_k = \begin{cases} \pi_k / \sum_{j \in \vec{i}} \pi_j & \text{if } k \in \vec{i} \\ 0 & \text{otherwise} \end{cases}$$

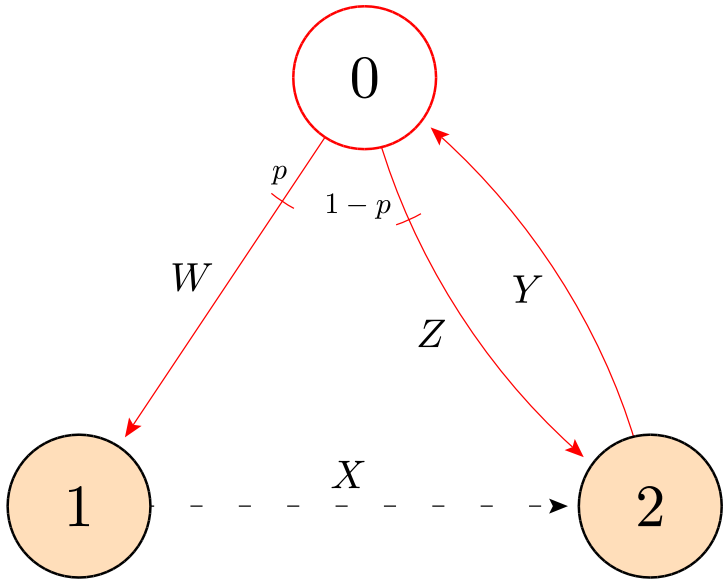
- Compute this efficiently using sparse matrix-vector multiplications and a vector accumulator

Passage Time Example



Distributions: $W \sim \Gamma(\alpha, \lambda)$, $Z \sim \text{Hyper}\left(\frac{2}{3}, \frac{1}{3}; \lambda, \mu\right)$, $X \sim \text{U}\left[\frac{1}{\lambda}, \frac{1}{\mu}\right]$, $Y \sim \delta\left[\frac{1}{\mu}\right]$

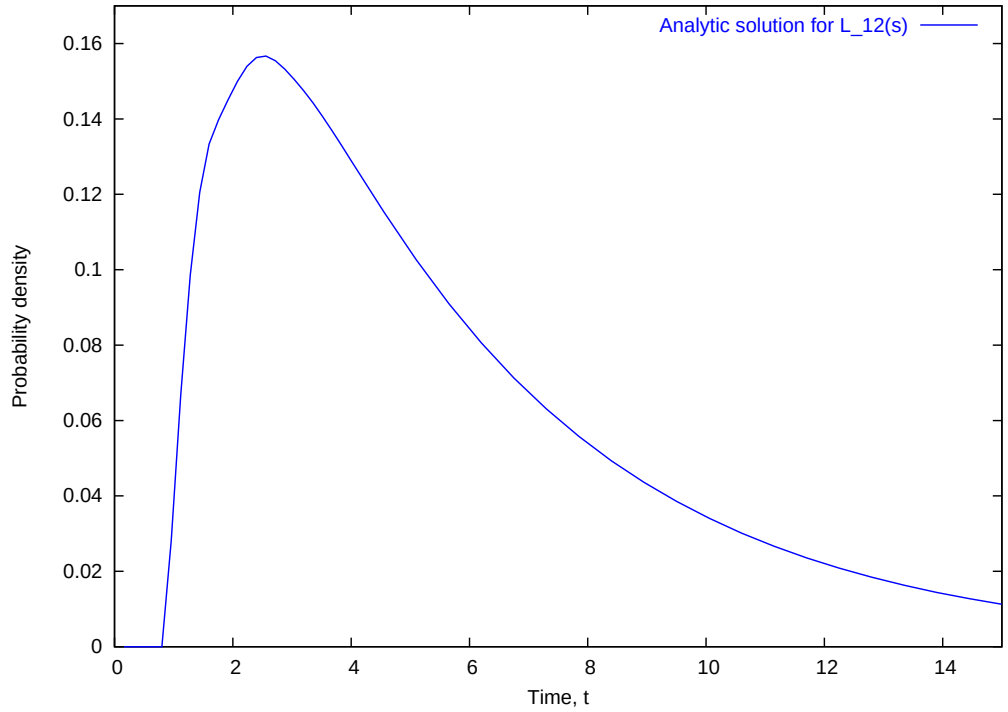
Passage Time: state 2 to 1



Distributions: $W \sim \Gamma(\alpha, \lambda)$, $Z \sim \text{Hyper}\left(\frac{2}{3}, \frac{1}{3}; \lambda, \mu\right)$, $X \sim \text{U}\left[\frac{1}{\lambda}, \frac{1}{\mu}\right]$, $Y \sim \delta\left[\frac{1}{\mu}\right]$

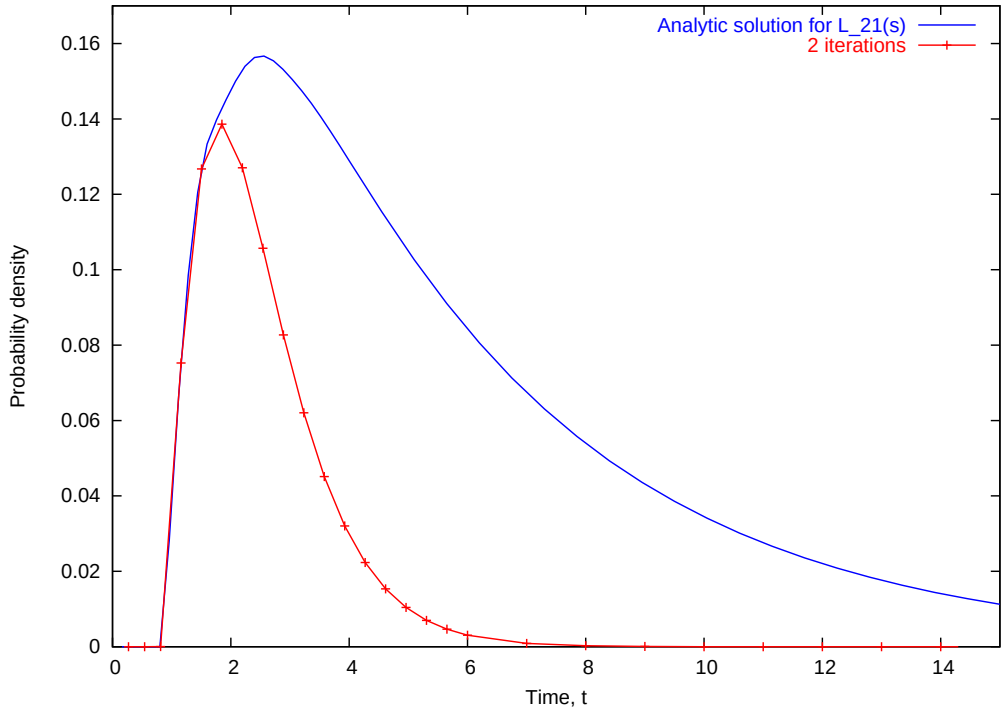
Passage Time: state 2 to 1

Analytic solution



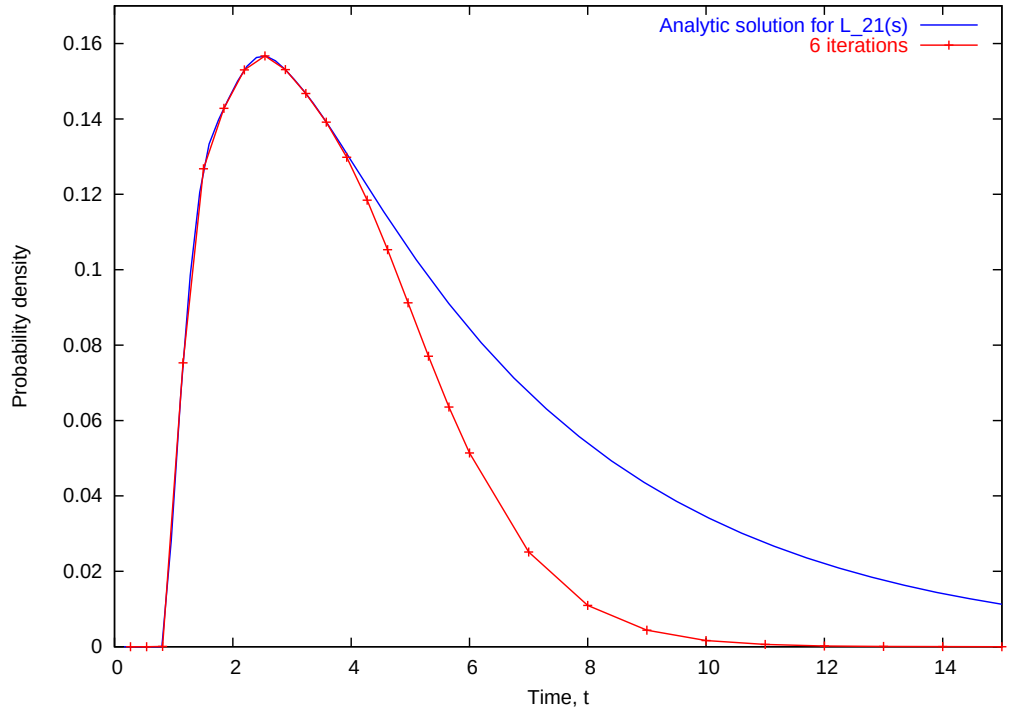
Passage Time: state 2 to 1

Iterations of algorithm: 2



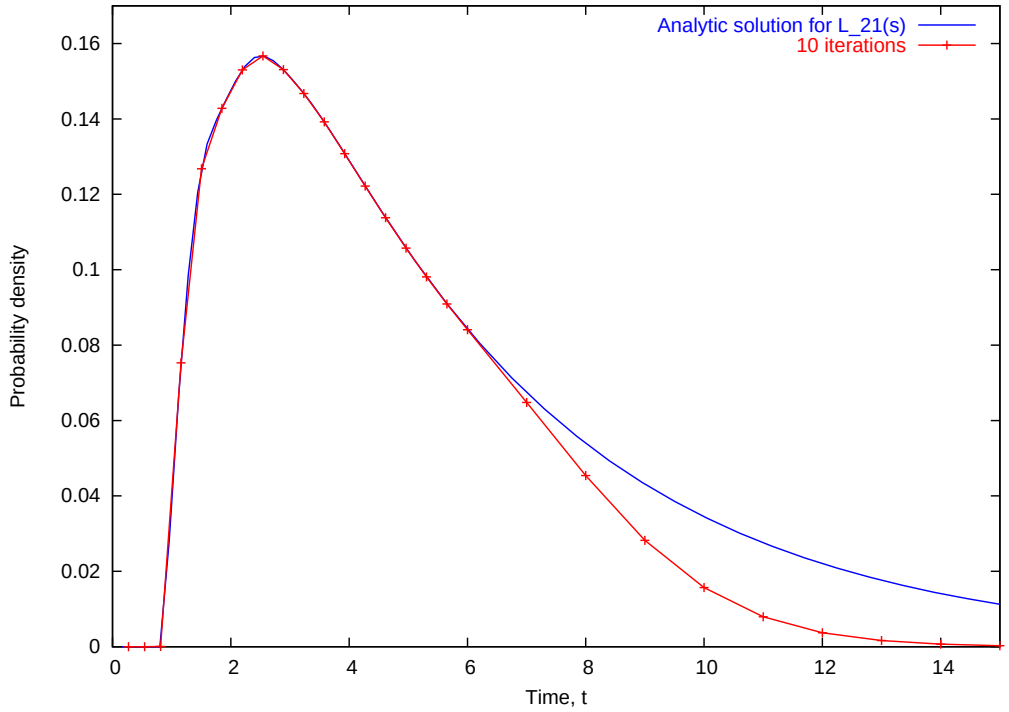
Passage Time: state 2 to 1

Iterations of algorithm: 6



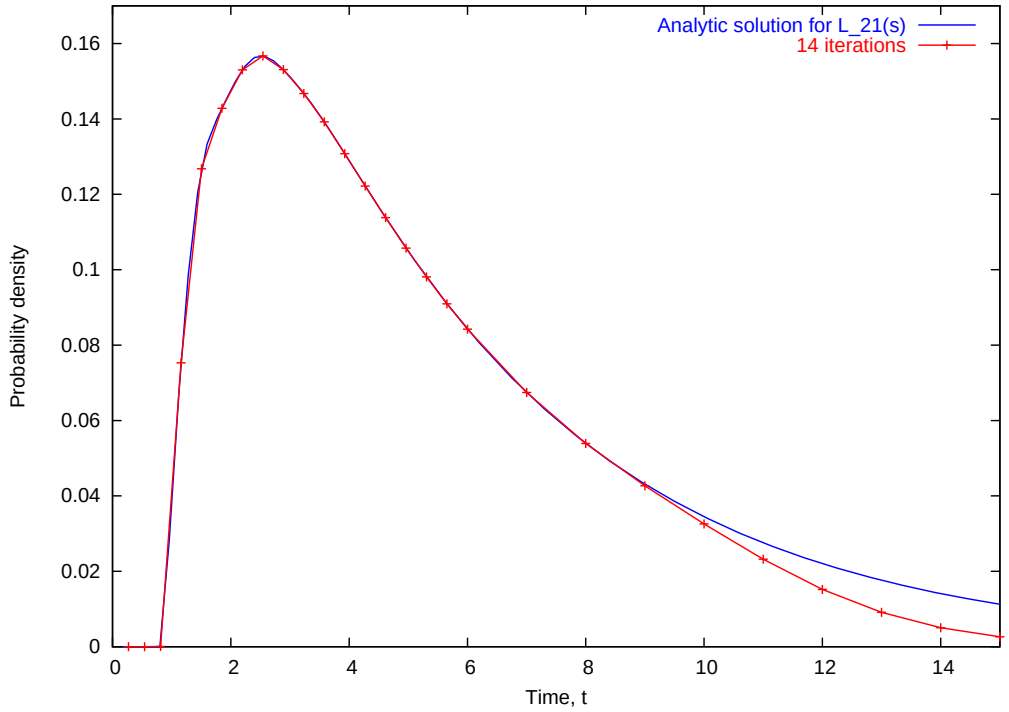
Passage Time: state 2 to 1

Iterations of algorithm: 10



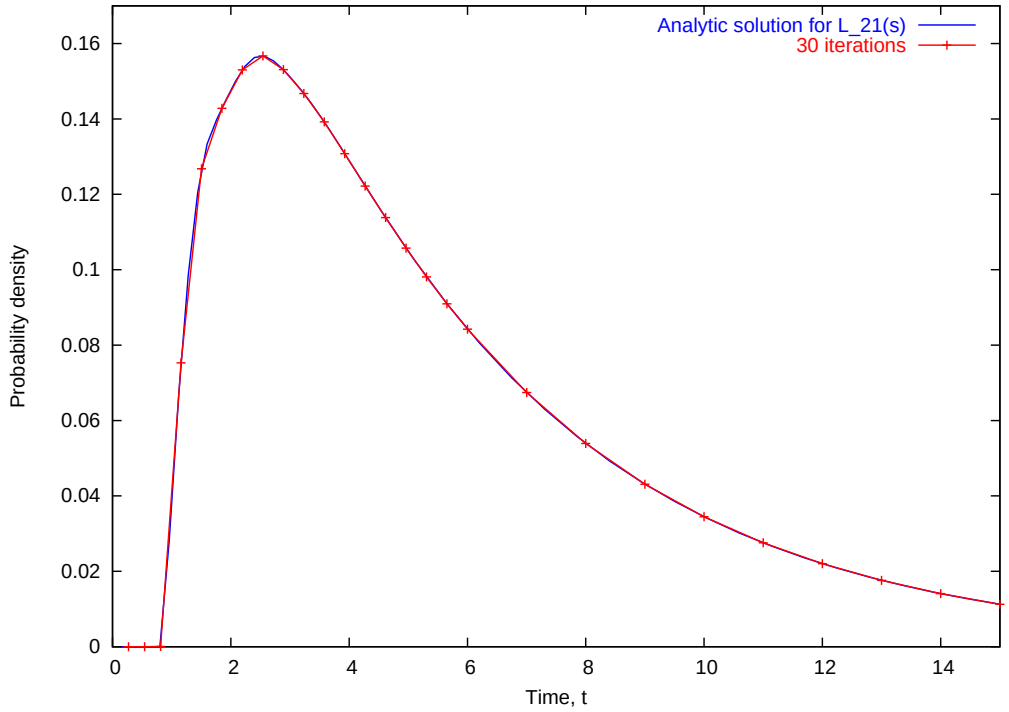
Passage Time: state 2 to 1

Iterations of algorithm: 14



Passage Time: state 2 to 1

Iterations of algorithm: 30



Truncation Error I

- The error for truncating the $L^{(r)}_{\overrightarrow{ij}}(s)$ sum at the rth term is

$$\sum_{i=r+1}^{\infty} \tilde{\alpha} U U'^{i} \simeq \frac{\lambda_1}{1-\lambda_1} \tilde{\alpha} U U'^{r}$$

  where λ_1 is dominant eigenvalue of U'

- Theoretical convergence condition is to find the minimum r such that:

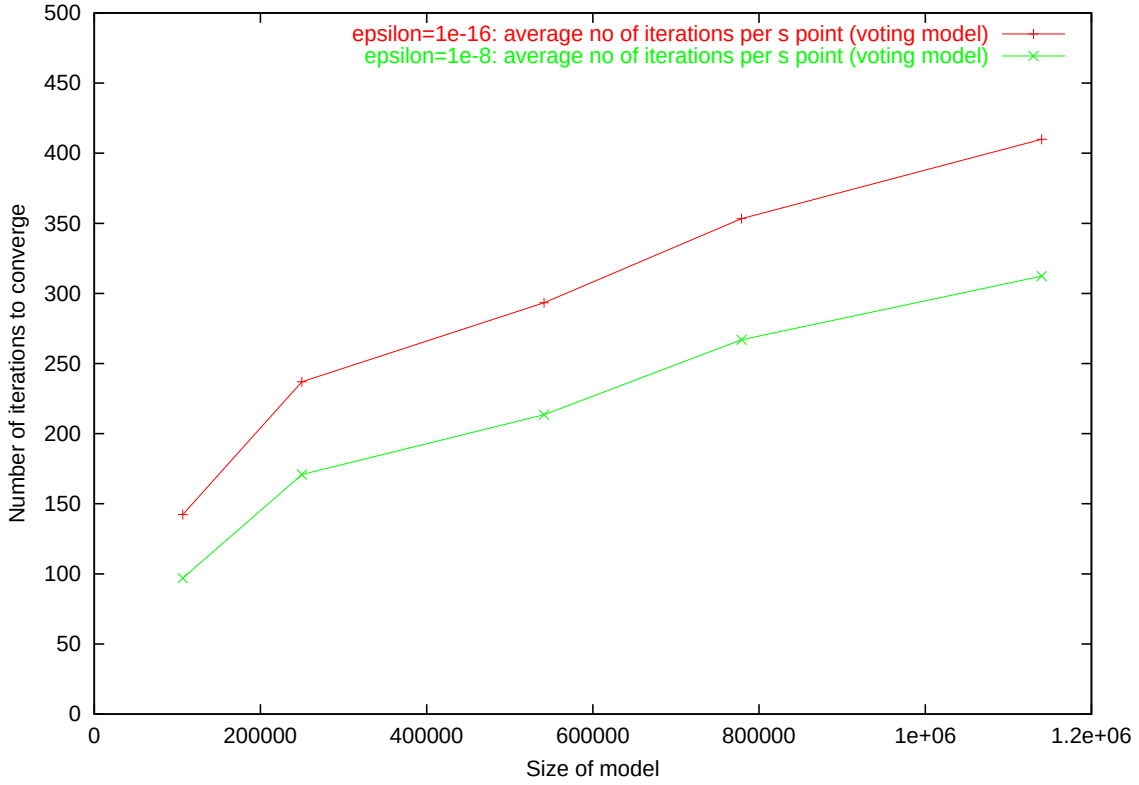
$$\left| \frac{\lambda_1}{1-\lambda_1} \tilde{\alpha} U U'^{r} \right| < \epsilon$$

Truncation Error II

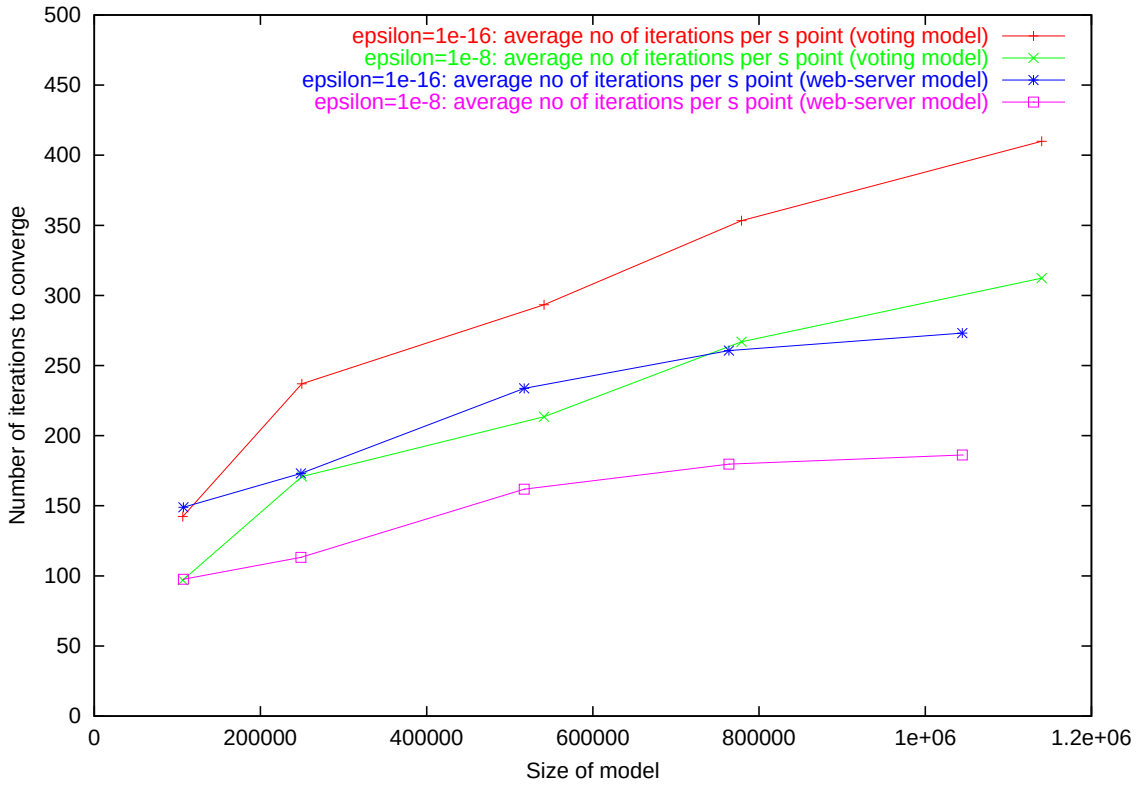
- Practical convergence test (if λ_1 is hard to find) is to compare the difference between successive iterations:

  $$|\mathrm{Re}(L_{\vec{i}\vec{j}}^{(r+1)}(s) - L_{\vec{i}\vec{j}}^{(r)}(s)| < \epsilon \quad \text{and} \quad |\mathrm{Im}(L_{\vec{i}\vec{j}}^{(r+1)}(s) - L_{\vec{i}\vec{j}}^{(r)}(s)| < \epsilon$$

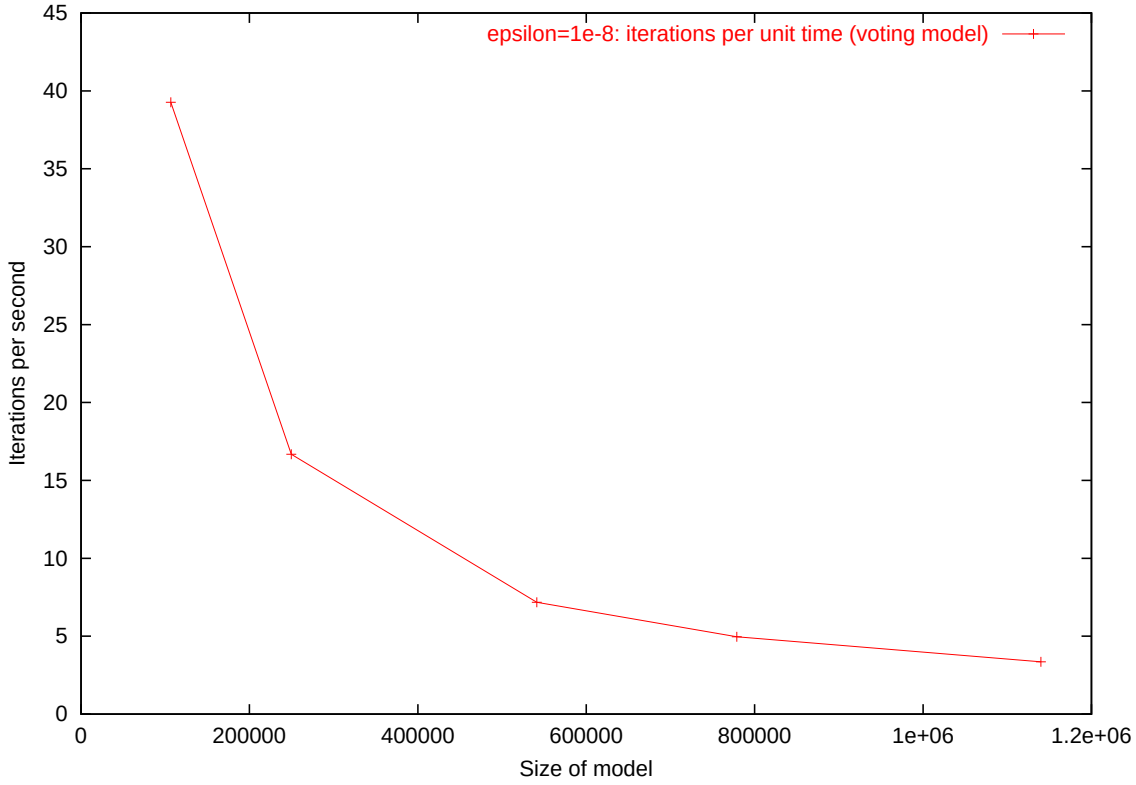
Convergence Results I



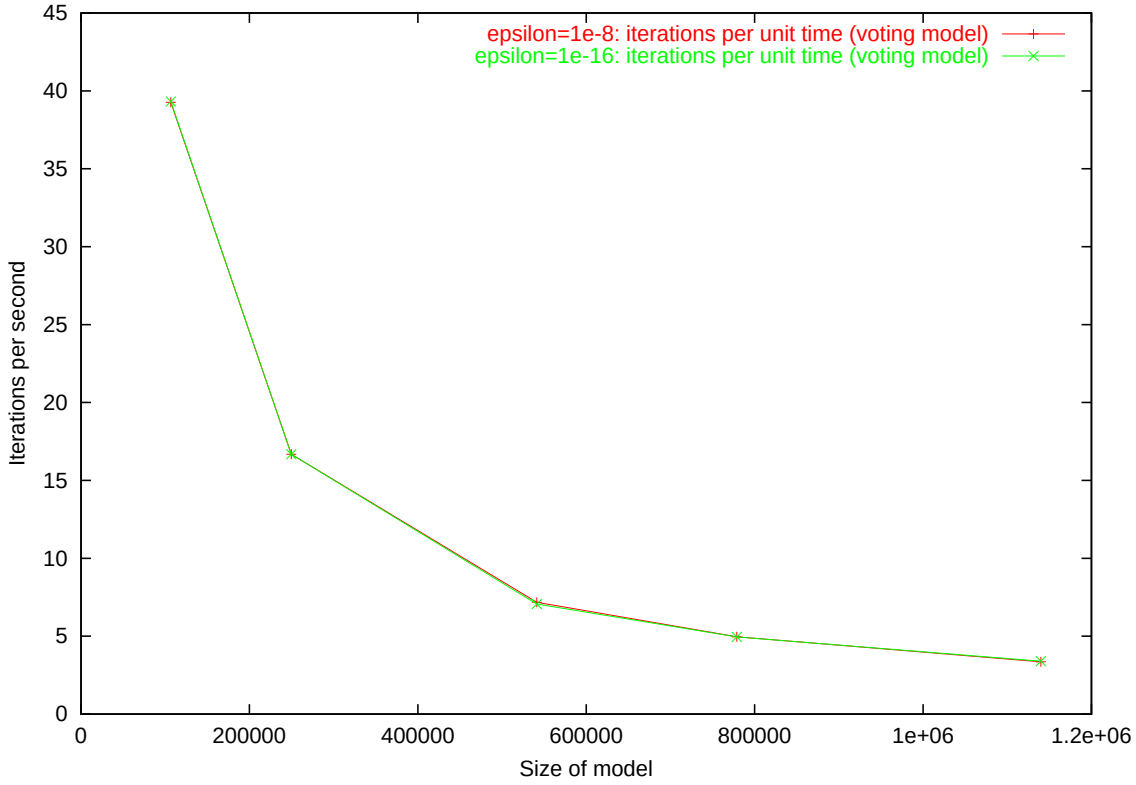
Convergence Results I



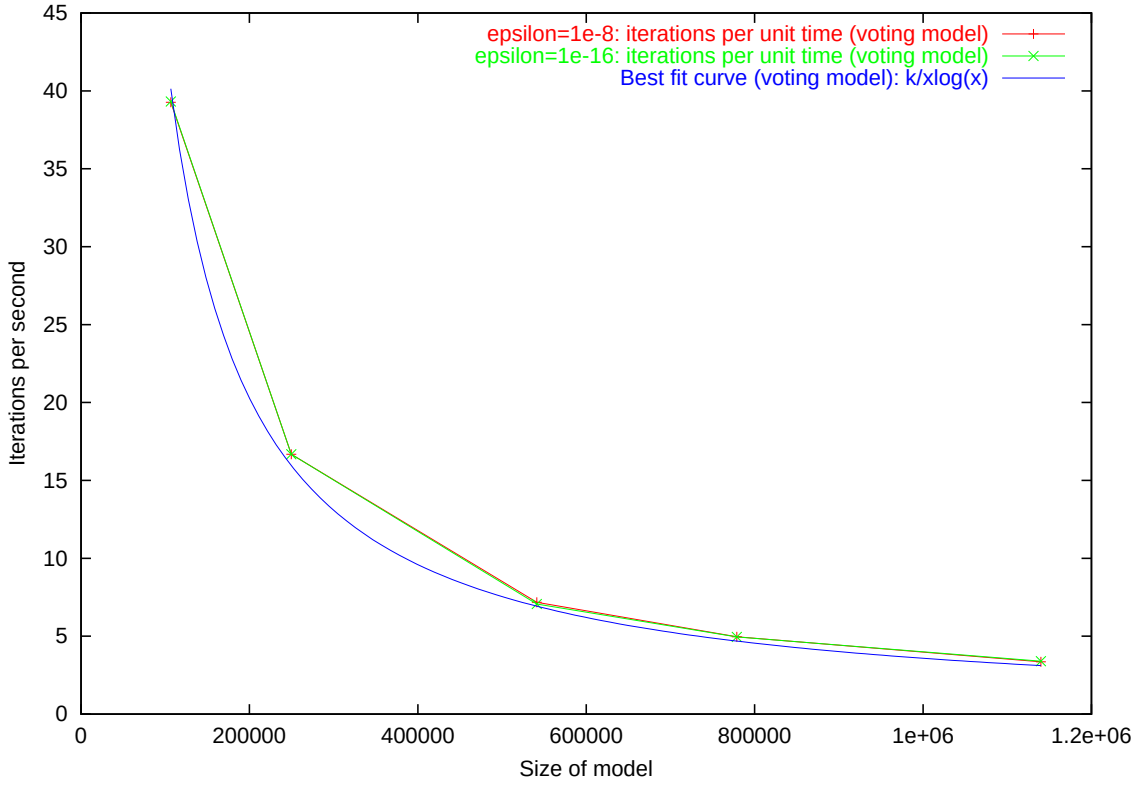
Convergence Results II



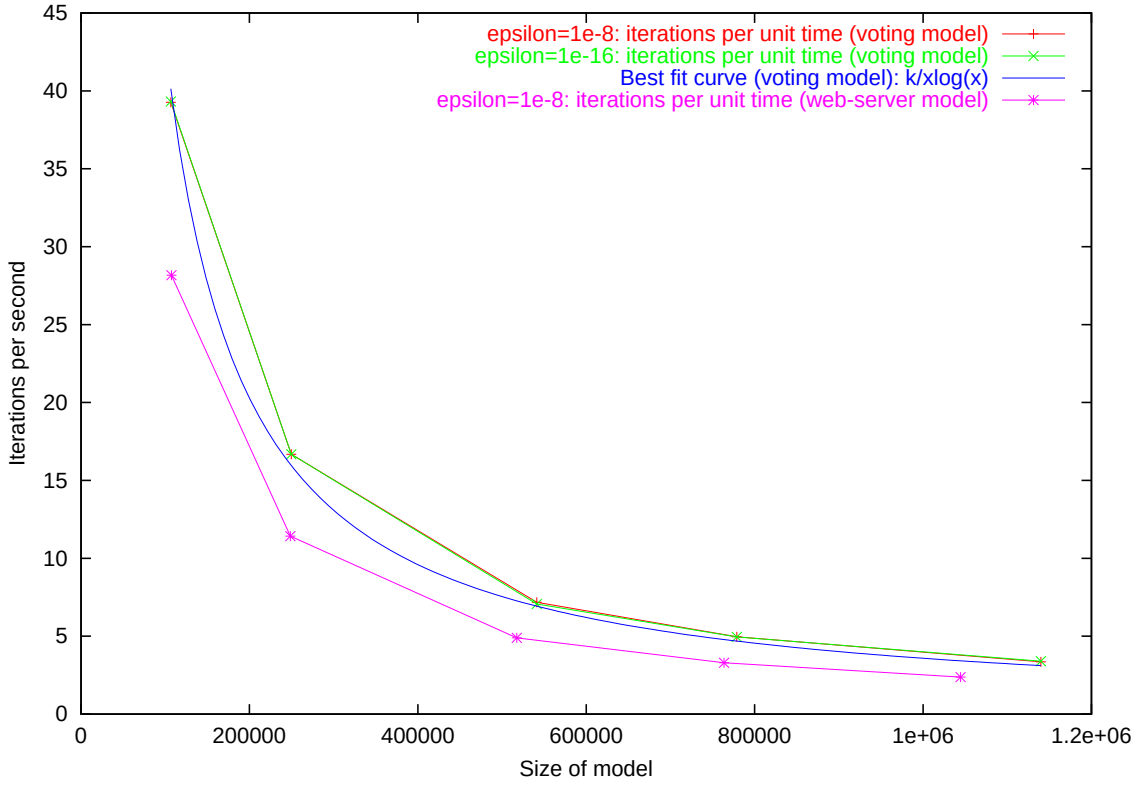
Convergence Results II



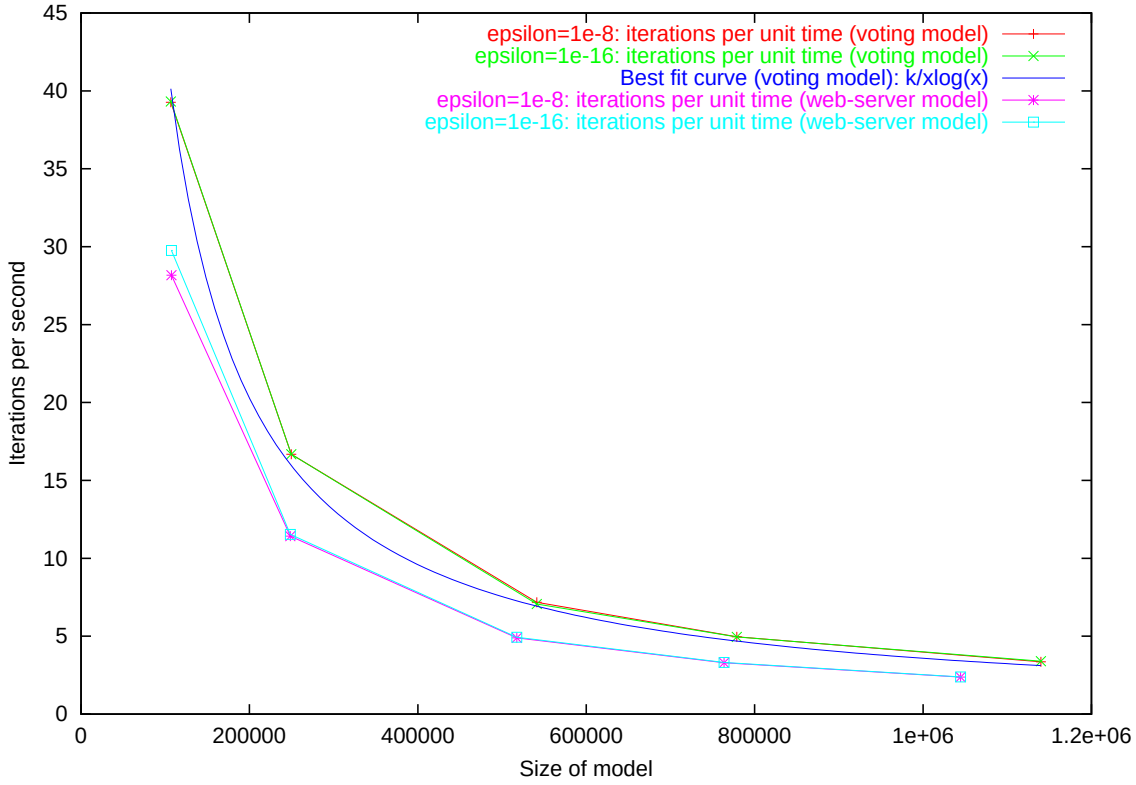
Convergence Results II



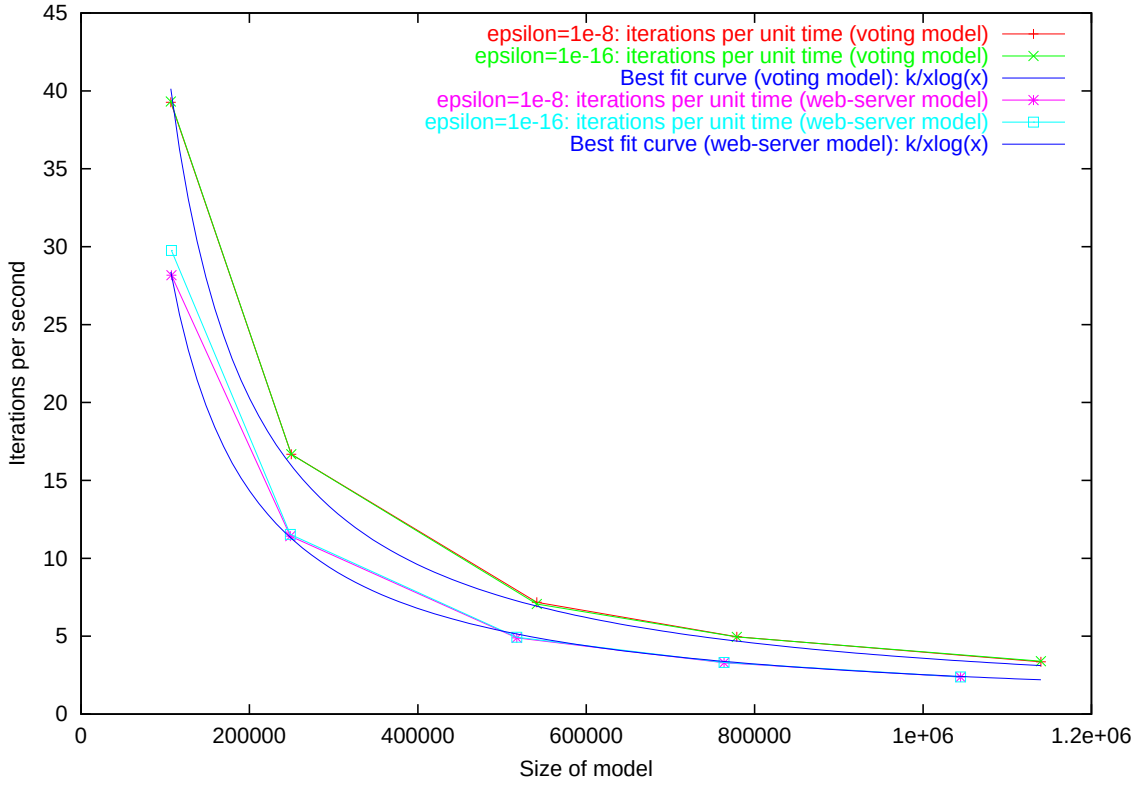
Convergence Results II



Convergence Results II



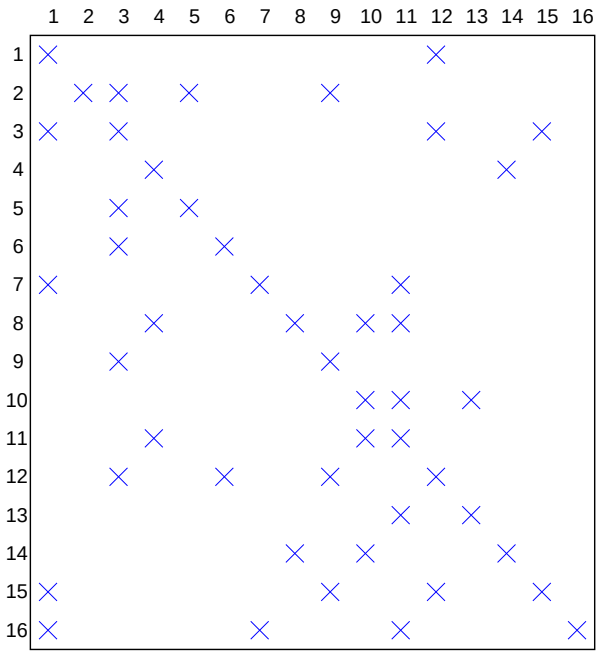
Convergence Results II

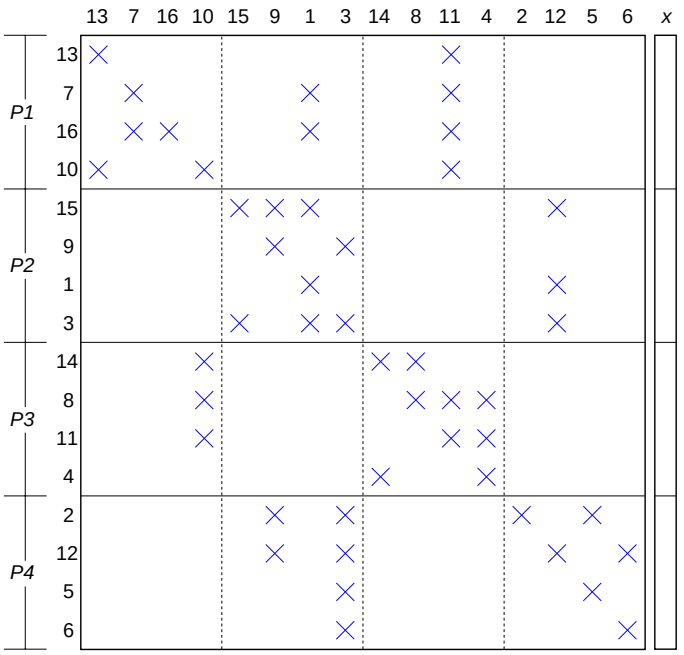


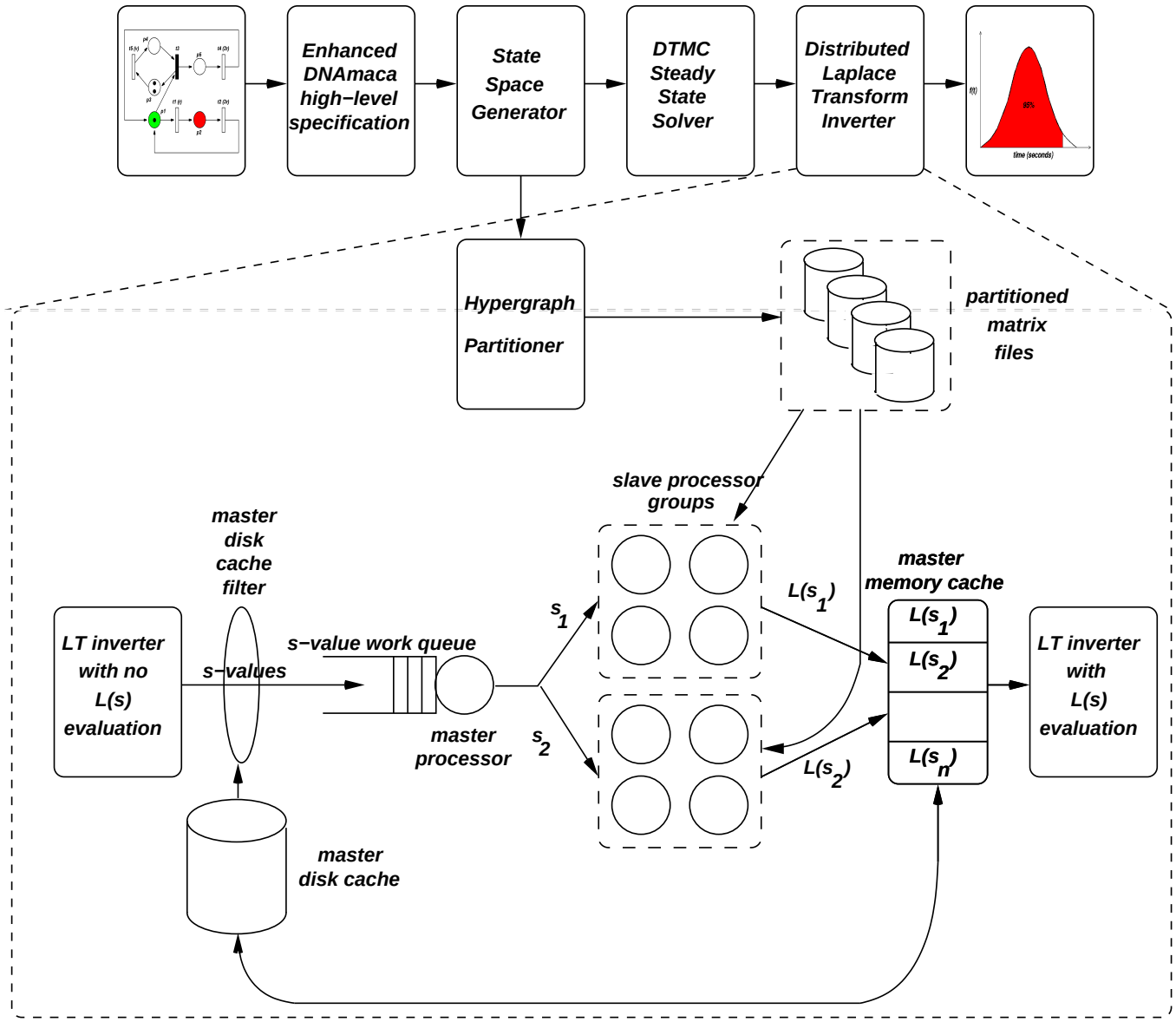
Data Partitioning Techniques

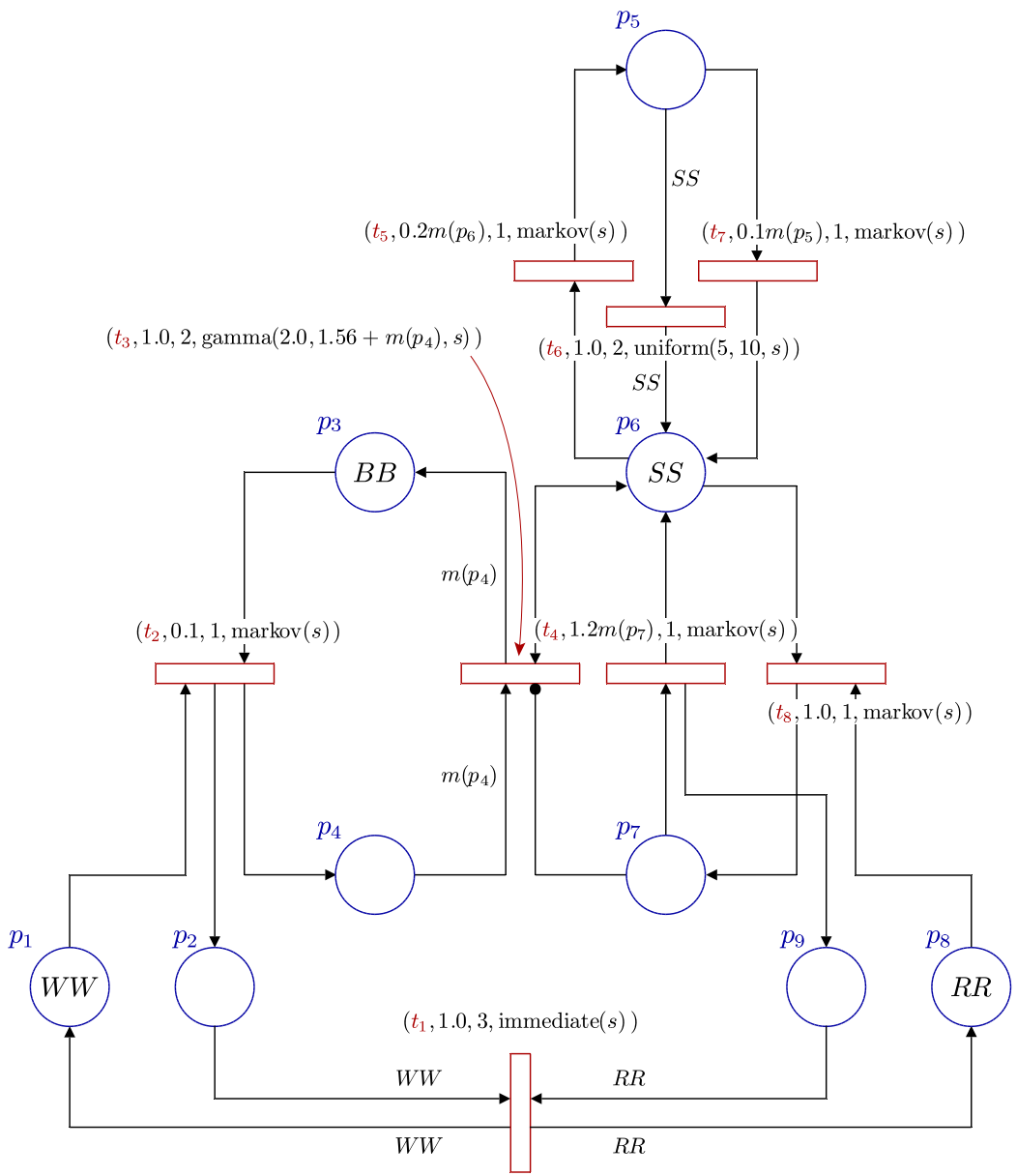
- For large models, we need to take advantage of the combined compute power and memory capacity of a network of workstations
- Main opportunity for parallelisation is in sparse matrix-vector multiplications
- Need to minimise the communication between processors while maintaining load balance

Hypergraph Partitioning

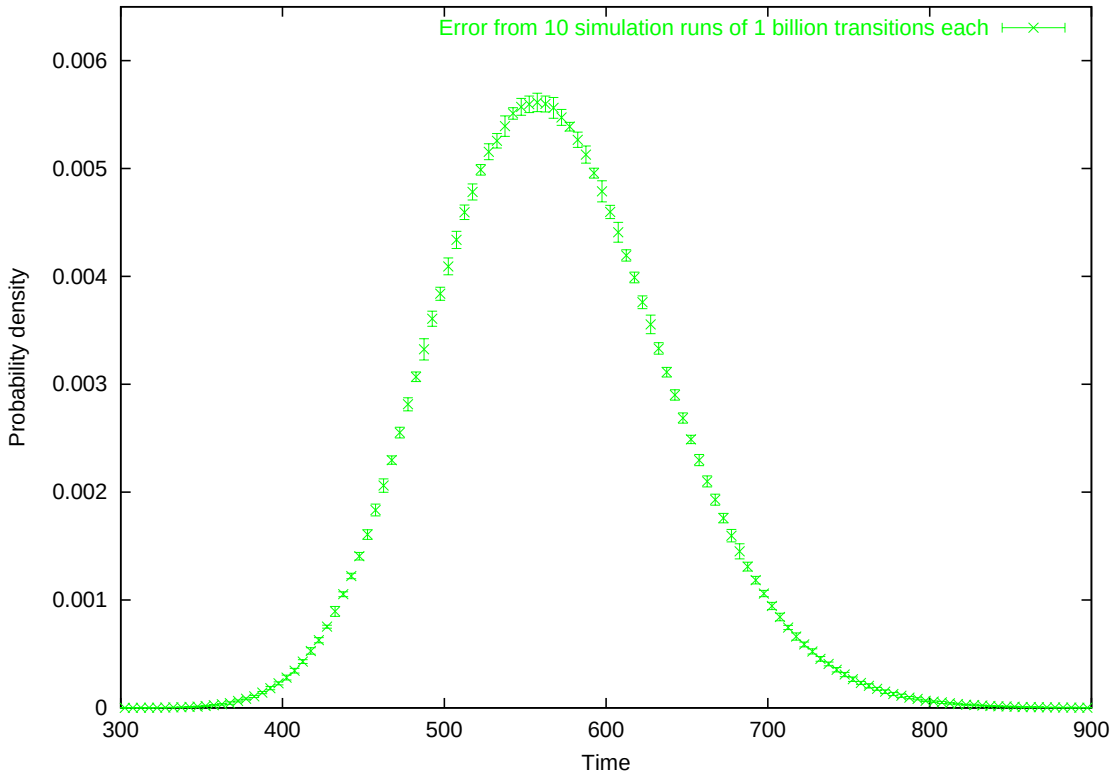




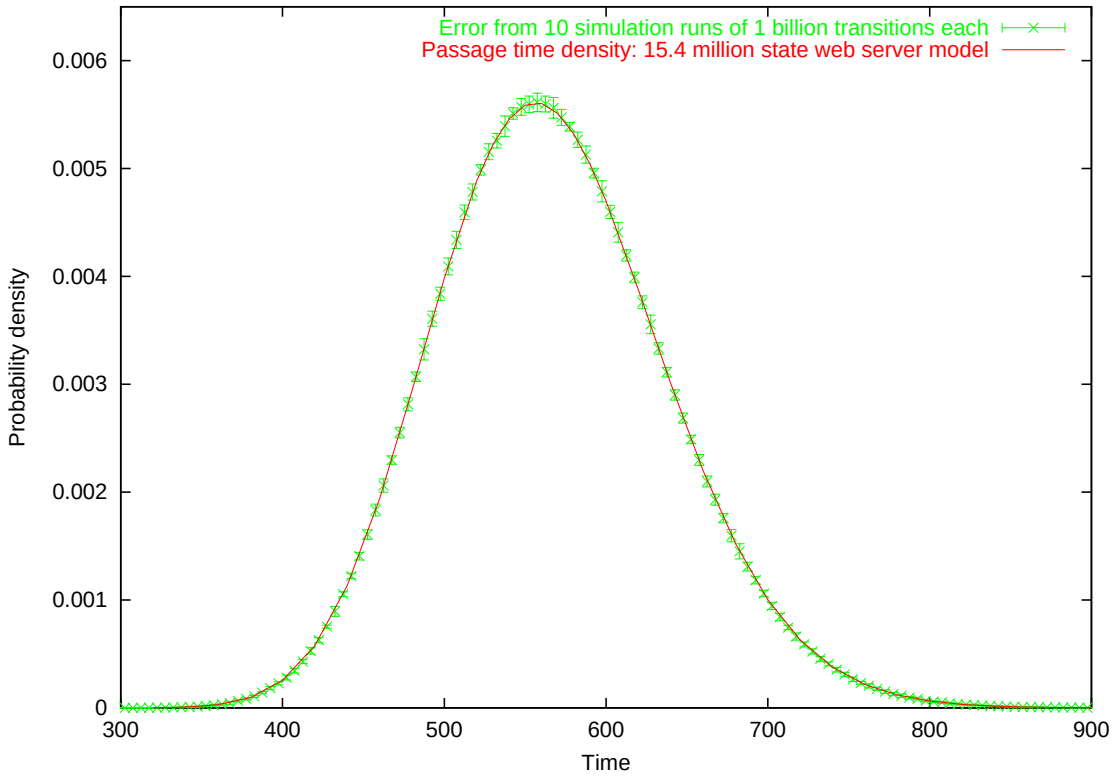

Enhanced DNAmaca high−level specification
State Space Generator
DTMC Steady State Solver
Distributed Laplace Transform Inverter
f(t)
95%
time (seconds)
Hypergraph Partitioner
partitioned matrix files
slave processor groups
master disk cache filter
master memory cache
LT inverter with no L(s) evaluation
s−values
s−value work queue
master processor
s_1
s_2
$L(s_1)$
$L(s_2)$
$L(s_n)$
LT inverter with L(s) evaluation
master disk cache


p_5
SS
$(t_5, 0.2m(p_6), 1, \text{markov}(s))$
$(t_7, 0.1m(p_5), 1, \text{markov}(s))$
$(t_3, 1.0, 2, \text{gamma}(2.0, 1.56 + m(p_4), s))$
$(t_6, 1.0, 2, \text{uniform}(5, 10, s))$
SS
p_3
p_6
BB
SS
$m(p_4)$
$(t_2, 0.1, 1, \text{markov}(s))$
$(t_4, 1.2m(p_7), 1, \text{markov}(s))$
$(t_8, 1.0, 1, \text{markov}(s))$
$m(p_4)$
p_4
p_7
p_1
p_2
p_9
p_8
WW
RR
$(t_1, 1.0, 3, \text{immediate}(s))$
WW
RR
WW
RR

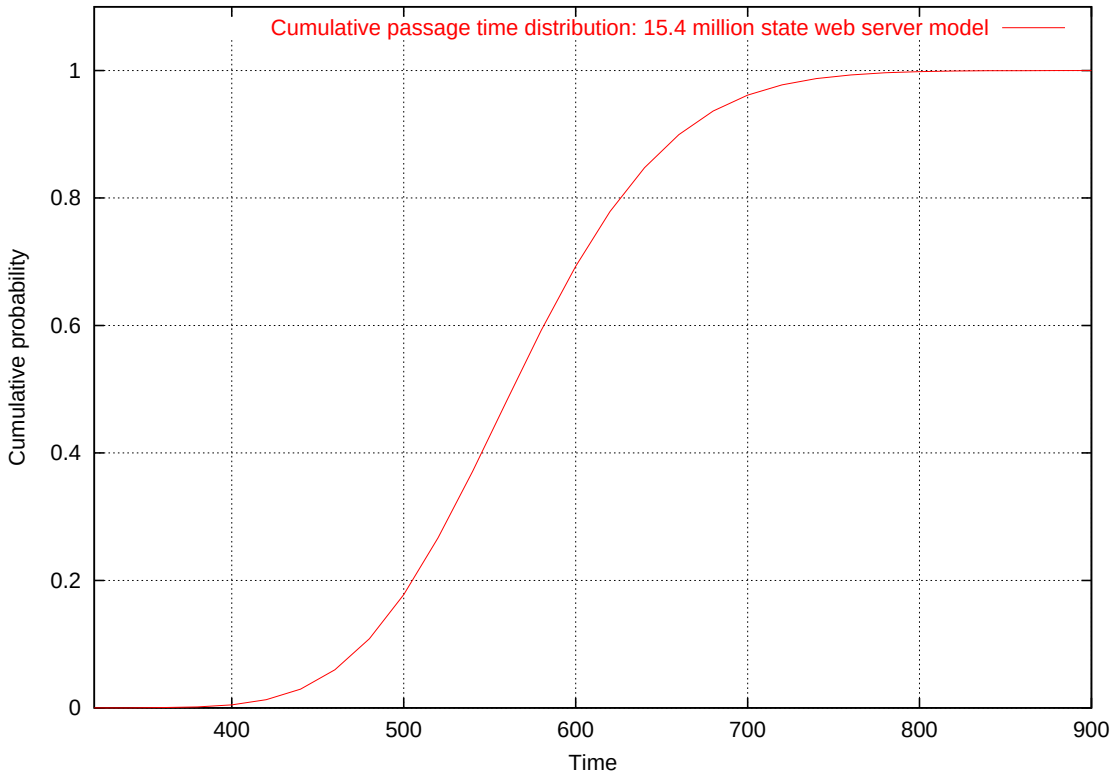
Numerical Results I



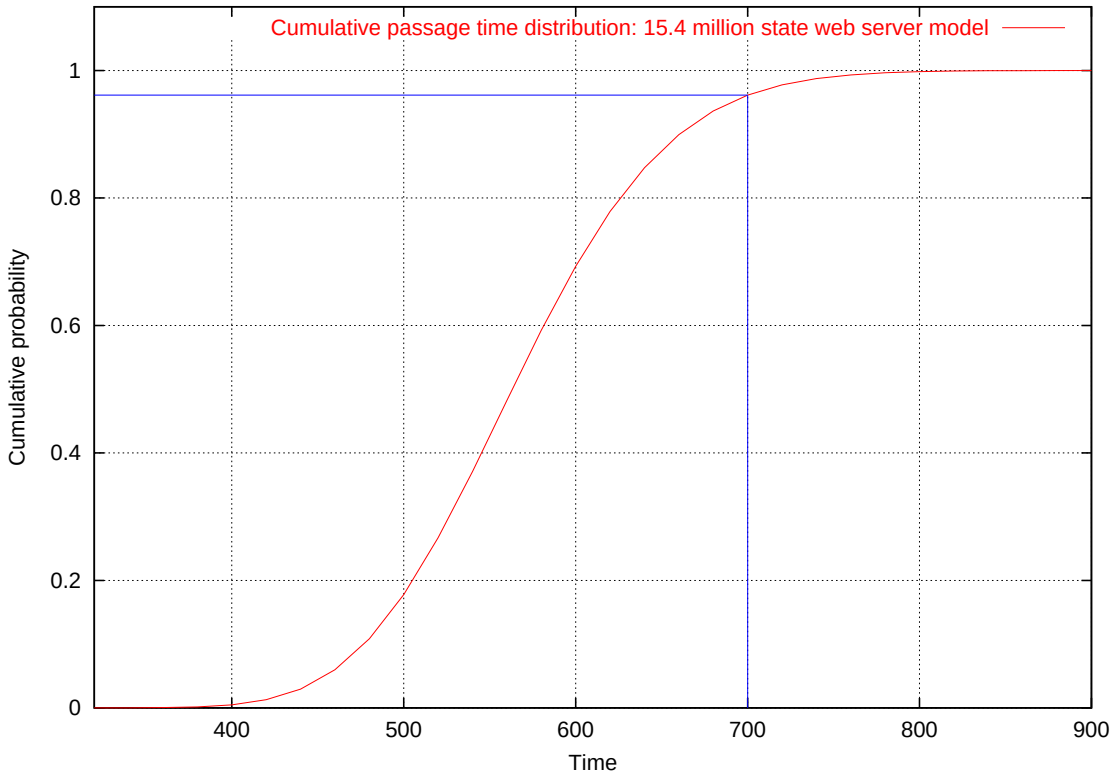
Numerical Results I



Numerical Results II



Numerical Results II



Scalability Results

p	Time (s)	Speedup	Efficiency
1	3968.07	1.00	1.000
2	2199.98	1.80	0.902
4	1122.97	3.53	0.883
8	594.07	6.68	0.835
16	320.19	12.39	0.775
32	188.14	21.09	0.659

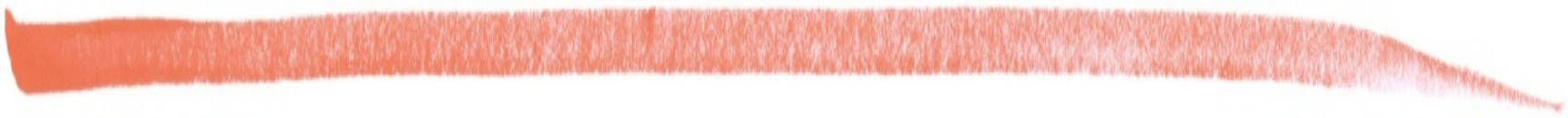
Scalability Results

p	Time (s)	Speedup	Efficiency
1	3968.07	1.00	1.000
2	2199.98	1.80	0.902
4	1122.97	3.53	0.883
8	594.07	6.68	0.835
16	320.19	12.39	0.775
32	188.14	21.09	0.659

p	Time (s)	Speedup	Efficiency
32×1	150.13	26.43	0.830
16×2	159.55	24.87	0.777
8×4	162.13	24.47	0.765
4×8	165.24	24.01	0.750
16×2	173.76	22.84	0.714
1×32	188.14	21.09	0.659

Conclusions

- We have developed a parallel iterative algorithm for computing passage time densities in large semi-Markov models
- We have examined the truncation error of our algorithm and observed a practical complexity of better than $O(N^2 \log N)$
- Key to its scalability is the use of hypergraph partitioning
- The method has been validated against simulation and found to be extremely accurate



Convergence of Algorithm

- Iterative algorithm approximates $L_{\vec{ij}}(s)$ with a geometric sum:

$$\lim_{r\to\infty} \sum_{i=0}^{r} \tilde{\alpha} U U'^{i} = \tilde{\alpha} U (I - U')^{-1}$$

- Converges iff $|\lambda_1| < 1$, where λ_1 is dominant eigenvalue of U'

- We know this is true for U' since:

$$P_{\vec{ij}}^{(r)} \to P_{\vec{ij}} \text{ as } r \to \infty$$