Reducing the Run-time of MCMC Programs by Multithreading on SMP Architectures

Jonathan M. R. Byrd Stephen A. Jarvis
Abhir H. Bhalerao

Department of Computer Science
University of Warwick

MTAAP IPDPS 2008

Outline

THE UNIVERSITY OF WARWICK

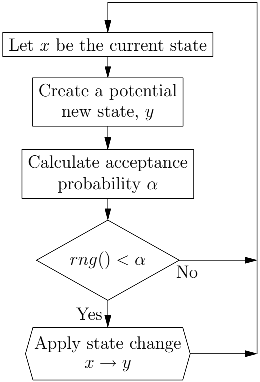
What is Markov Chain Monte Carlo?

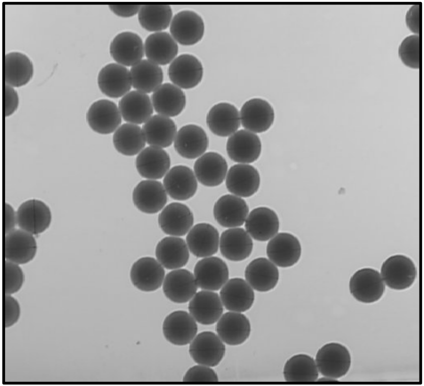
- MCMC is a computationally expensive iterative technique for sampling from a probability distribution.
- Basic idea:
  - Construct a Markov Chain such that its stationary distribution is equal to the distribution we wish to sample.
  - After sufficient burn-in time, sampling from the chain is equivalent to sampling from the distribution.

What uses Markov Chain Monte Carlo?

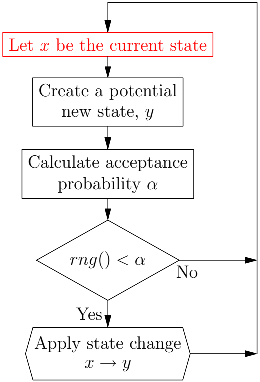
- MCMC is widely used in
  - Bayesian statistics
  - Computational physics
  - Computational biology
- Specific applications include:
  - Phylogenetic analysis
  - Spectral modelling of X-ray data from the Chandra X-ray satellite
  - Calculating financial econometrics
  - Mapping vascular trees from retinal slides

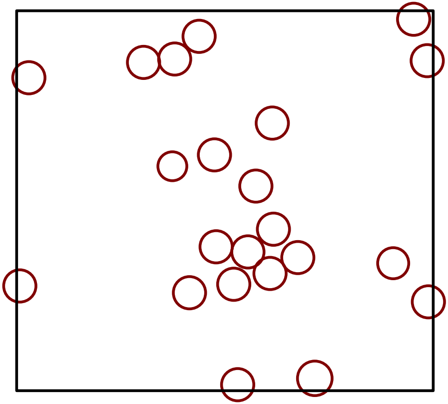
The MCMC Program Cycle



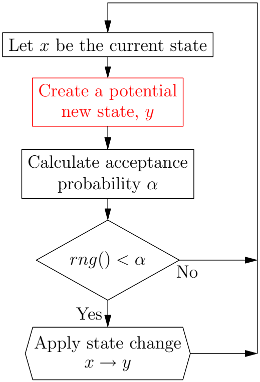


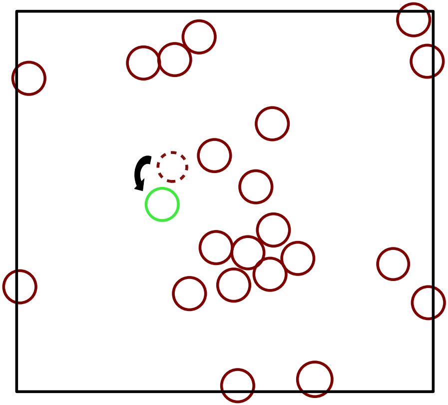
The MCMC Program Cycle



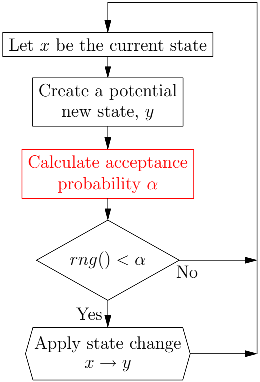


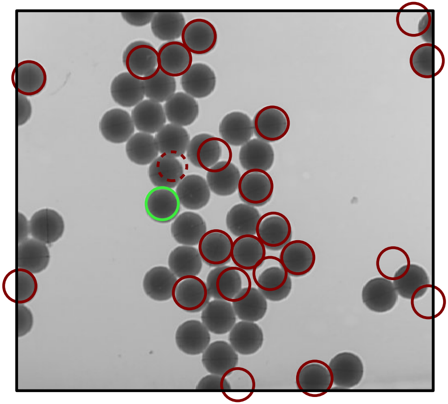
The MCMC Program Cycle



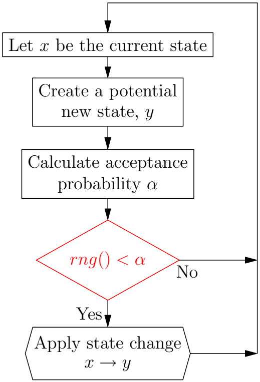


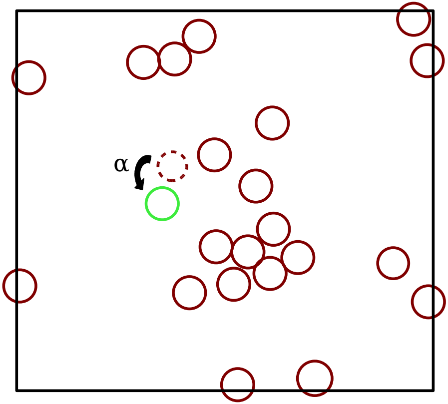
The MCMC Program Cycle



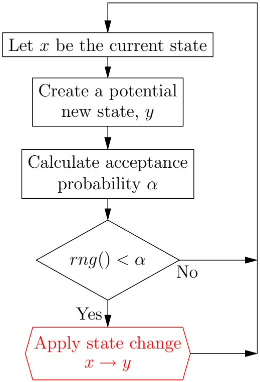


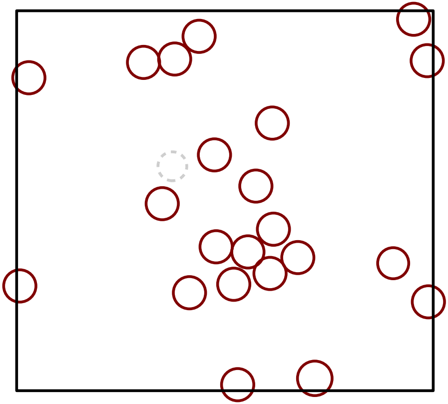
The MCMC Program Cycle



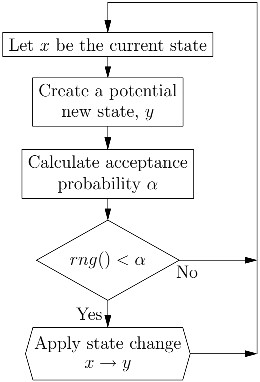


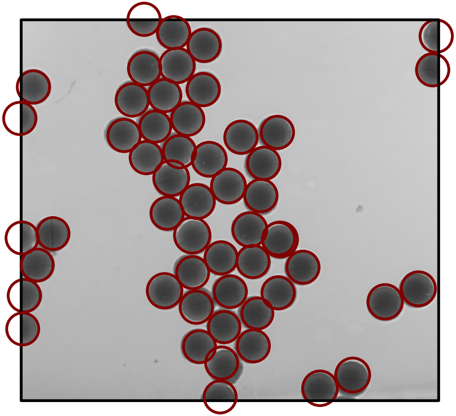
The MCMC Program Cycle





The MCMC Program Cycle





Existing Parallelisation

- Execute multiple chains. Take samples from all of them.
  - Embarrassingly parallel
  - Does not reduce burn-in time.
  - Does not help escape local optima.
- Metropolis-Coupled MCMC
  - Execute multiple chains.
  - Coarse parallelisation, machines connected by LAN.
  - Modifies algorithm to improve rate of convergence.
  - Good for escaping local optima.
  - Hard to predict benefits.

Existing Parallelisation

- Execute multiple chains. Take samples from all of them.
    - Embarrassingly parallel
    - Does not reduce burn-in time.
    - Does not help escape local optima.
- Metropolis-Coupled MCMC
    - Execute multiple chains.
    - Coarse parallelisation, machines connected by LAN.
    - Modifies algorithm to improve rate of convergence.
    - Good for escaping local optima.
    - Hard to predict benefits.

Introducing Speculative Moves

1. Each state in a Markov Chain must depend on only the preceding state.
2. But, typically only $\frac{1}{4}$ of iterations accept the proposed state-change.
3. Consecutive rejected iterations could have been performed in parallel.
4. => Assume all iterations will be rejected.

Introducing Speculative Moves

1. Each state in a Markov Chain must depend on only the preceding state.
2. But, typically only $\frac{1}{4}$ of iterations accept the proposed state-change.
3. Consecutive rejected iterations could have been performed in parallel.
4. => Assume all iterations will be rejected.

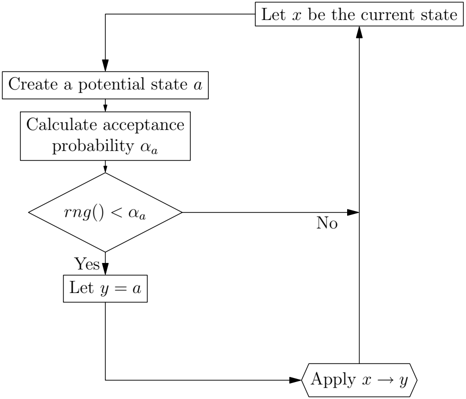
Introducing Speculative Moves

1. Each state in a Markov Chain must depend on only the preceding state.
2. But, typically only $\frac{1}{4}$ of iterations accept the proposed state-change.
3. Consecutive rejected iterations could have been performed in parallel.
4. => Assume all iterations will be rejected.

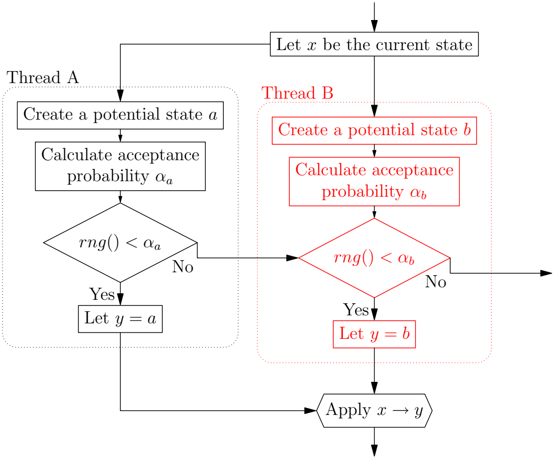
Introducing Speculative Moves

1. Each state in a Markov Chain must depend on only the preceding state.
2. But, typically only $\frac{1}{4}$ of iterations accept the proposed state-change.
3. Consecutive rejected iterations could have been performed in parallel.
4. => Assume all iterations will be rejected.

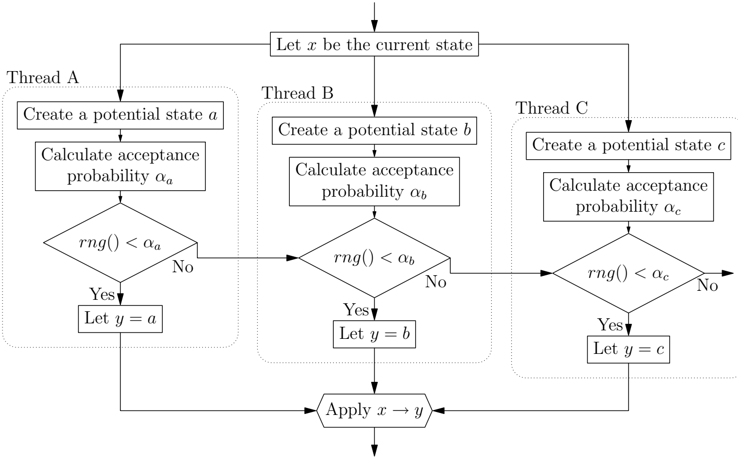
Speculative Moves Program Cycle



Speculative Moves Program Cycle



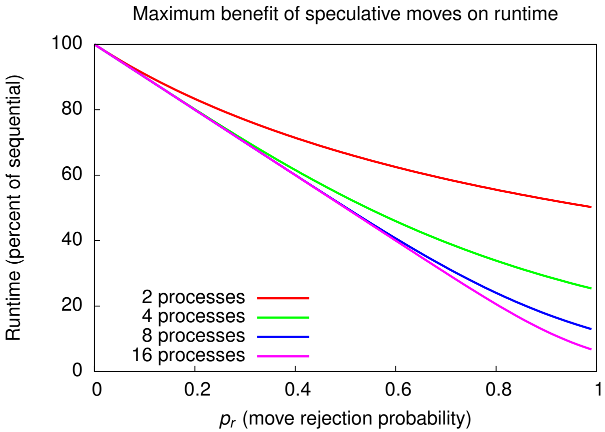
Speculative Moves Program Cycle



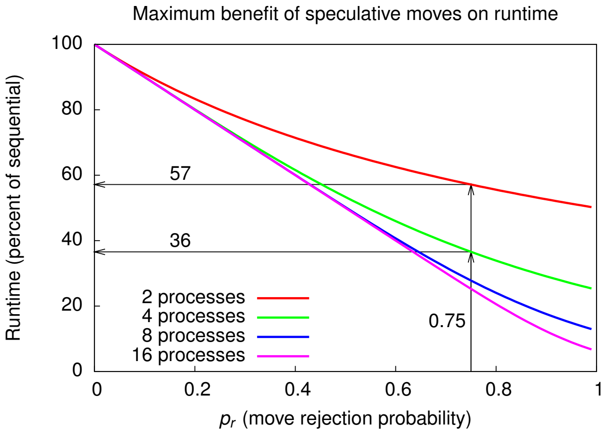
Theoretical Benefits of Speculative Moves

- Let:
  - n be the number of iterations considered concurrently.
  - p_r be the average state-change rejection probability.
- Each program cycle performs $1..n$ MCMC iterations.
- On average $\frac{1-p_r^n}{1-p_r}$ MCMC iterations are performed at each loop of the program cycle.
- If multithreading overhead negligible, time for 1 program cycle $\approx$ time for 1 MCMC iteration.

Theoretical Results



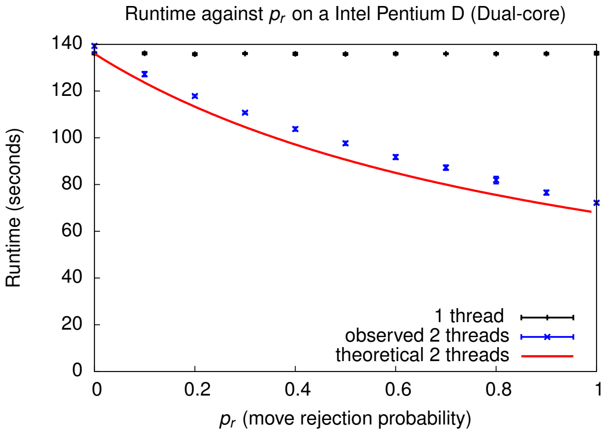
Theoretical Results



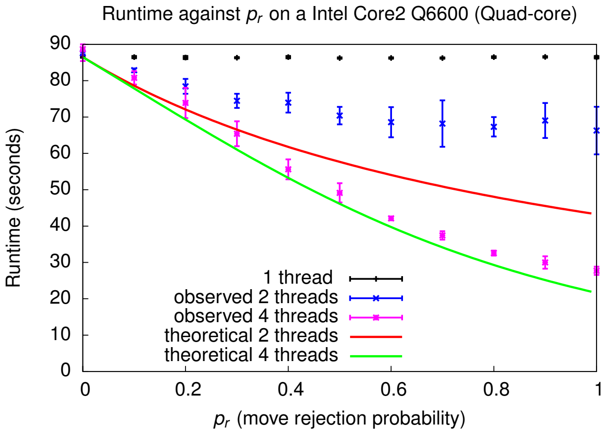
Practical Testing

- Circle detection algorithm used for testing
  - Fixed number of iterations
  - Autogenerated images
  - Runtime values averaged over 20 runs
- Hardware utilised:
  - AMD Athlon 64 X2 4400+ (dual-core)
  - Intel Xeon Dual-Processor
  - Intel Pentium-D (dual core)
  - Intel Core2 Quad Q6600 (2x dual-core dies)
  - 56 Itanium2 processor SGI Altix

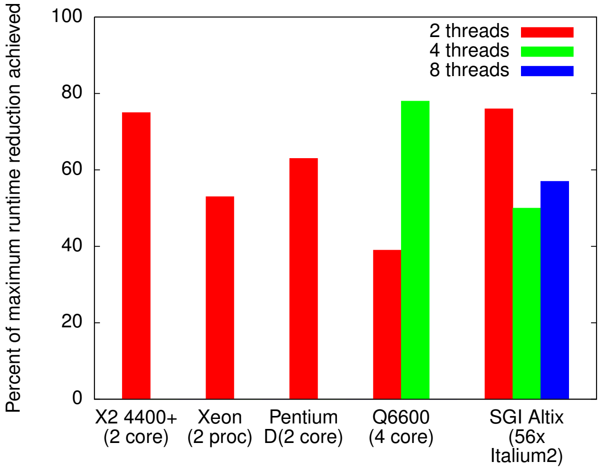
Comparing Practical with Theoretical (1)



Comparing Practical with Theoretical (2)



Preferable Architectures



Will I Benefit?

This table shows the iteration time at which the overhead from multithreading balances the benefits, when $p_r = 0.75$.

	Iteration Time (μs)	Iteration Rate (s^{-1})
Xeon Dual-Processor	70	14 285
Pentium-D (dual core)	55	18 181
Q6600 (using 2 threads)	75	13 333
Q6600 (using 4 threads)	25	40 000

Summary

- The speculative moves method uses increasingly available multiprocessor and multicore machines to reduce the runtime of MCMC program.
- The statistical algorithm is preserved. Speculative moves will not effect the results, only the real-time required to obtain them.
- Real-time reductions of 35% using a dual-core and 55% using quad-cores machines have been demonstrated.