Parallel Metropolis-Hastings Algorithm by Prefetching

Boyan Bejanov (bbejanov@bankofcanada.ca)

1. Introduction

Metropolis-Hastings (MH) is a Markov Chain Monte Carlo (MCMC) algorithm which is used to simulate random samples from probability distributions that are difficult to simulate otherwise.

Prefetching is a parallelization technique for the MH algorithm. It is applicable in situations where the target p.d.f., $\pi(\mathbf{x})$, **is**

- **computationally very expensive;**
- **impractical to parallelize.**

E.g. Bayesian inference for economic models.

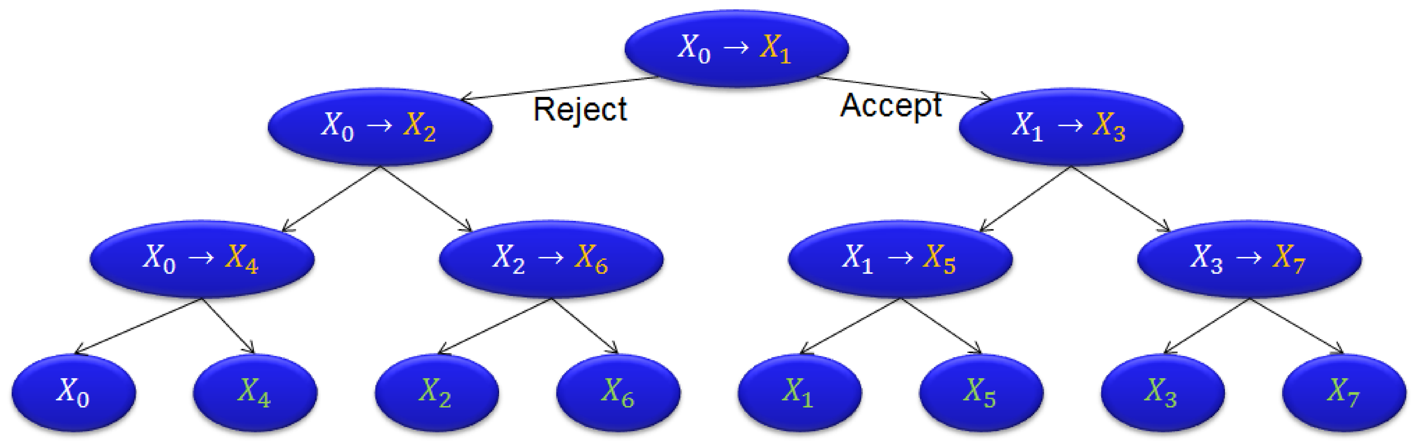
2. MH algorithm

Markov chain with accept-reject transition rule:

- Current state is X_0.
- Propose candidate $Y \sim q(X_0, \cdot)$.
- Accept the candidate with probability

$$\alpha = \frac{\pi(Y)q(Y, X_0)}{\pi(X_0)q(X_0, Y)}.$$

3. Prefetching tree



- At each node the current state is in white.
- The proposed candidate is in orange.

4. Implemetation

step s	0	1	2		3				4							
current c	$*$	0	0	1	0	1	2	3	0	1	2	3	4	5	6	7
proposed k	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

- X_k was proposed at step $s = \lfloor \log k \rfloor + 1$.
- X_k was proposed from X_c, $c = k - 2^{s-1}$.
- The two children of proposal X_k are X_a and X_r, where $a = k + 2^s$ and $r = a - 2^{s-1}$.

5. Full prefetching

- Consider all 2^n possible paths n steps ahead.
- Compute $2^n - 1$ evaluations of $\pi(\cdot)$ in parallel.
- Run n steps of the Markov chain sequentially.

6. Complexity of full prefetching

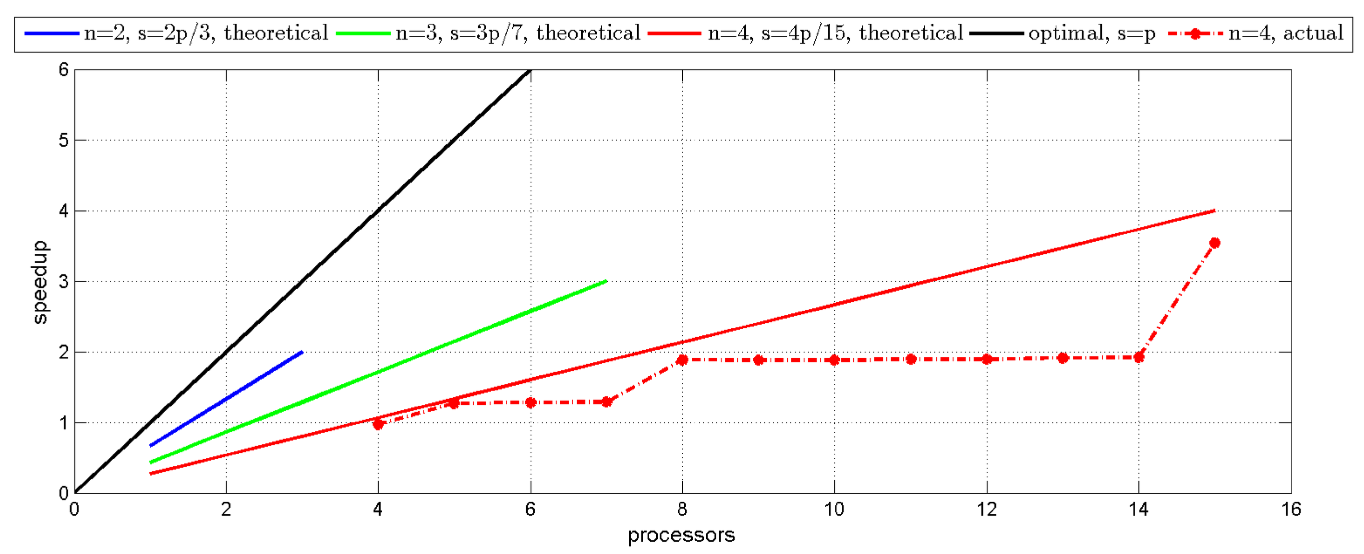
Sequential time $T_s = n$

Parallel time $T(n, p) = \dfrac{2^n - 1}{p}$

Speedup $s(n, p) = \dfrac{np}{2^n - 1}$

N.B. *Speedup is not logarithmic in p.*

7. Speedup graph: full prefetching



8. Incomplete prefetching

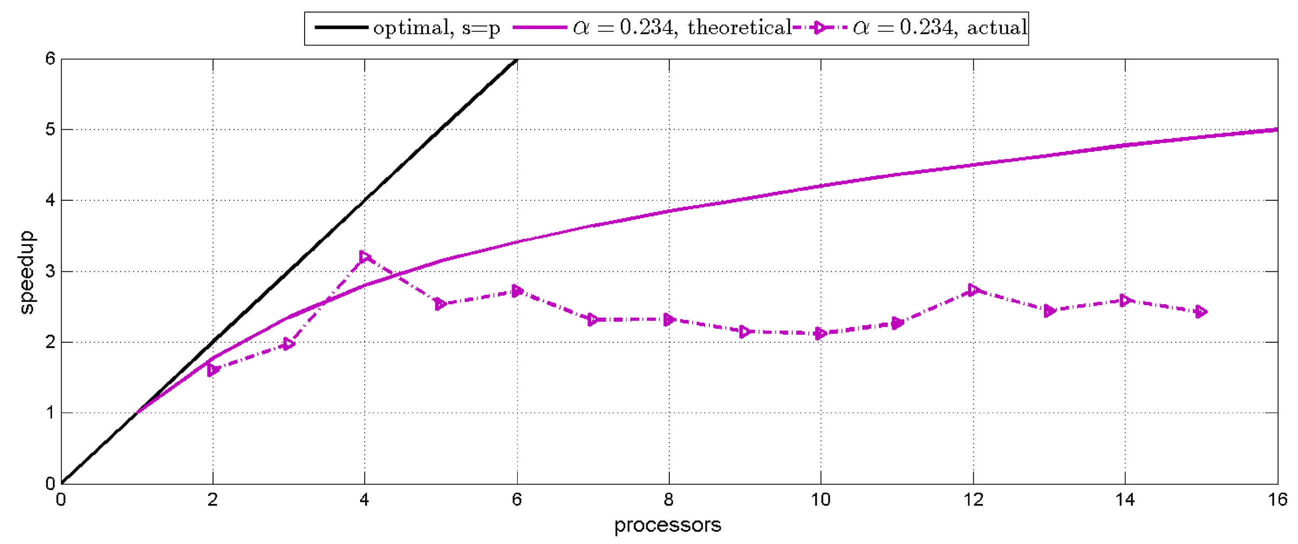
- Prefetch only p proposals, not the full tree.
- By scaling the proposal distribution, $q(x, y)$, we can control the acceptance rate, denoted α^*.
- Choose the p candidates to maximize the expected depth to be reached, denoted $D(p)$.

9. Complexity of incomplete prefetching

Parallel time $T(p) = \frac{n}{D(p)}$

Speedup $s(p) = D(p)$

10. Speedup graph: incomplete



11. Caveat emptor

- Full prefetching provides speedup at the cost of exponential number of processors $\Rightarrow$ inefficient.
- Incomplete prefetching gives improved *expected* efficiency, although not guaranteed.

12. References

- A.E. Brockwell, *JCGS*, 2006, 15(1), 246-261.
- J. Byrd, S. Jarvis, A.H. Bhalerao, *ISPDP*, 2008, 1-8.
- I. Strid, *CSDA*, 2010, 54(11), 2814-2835.