Distribution Agreement

In presenting this thesis or dissertation as a partial fulfillment of the requirements for an advanced degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my thesis or dissertation in whole or in part in all forms of media, now or hereafter known, including display on the world wide web. I understand that I may select some access restrictions as part of the online submission of this thesis or dissertation. I retain all ownership rights to the copyright of the thesis or dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

Signature:

_____ _____

Zhifan Sang Date

Evaluating Scalable Markov Chain Monte Carlo Algorithms for Big Data Problems

By

Zhifan Sang
Master of Science in Public Health

Department of Biostatistics and Bioinformatics

_____

Qi Long, Ph.D.
Committee Chair

_____

Suprateek Kundu, Ph.D.
Committee Member

Extending and Evaluating Scalable Markov chain Monte Carlo Algorithms for Big Data Problems

By

Zhifan Sang

B.S., Nankai University, 2014

Thesis Committee Chair:

Qi Long, Ph.D.

An abstract of
A thesis submitted to the Faculty of the
Rollins School of Public Health of Emory University
in partial fulfillment of the requirements for the degree of
Master of Science in Public Health
in Department of Biostatistics and Bioinformatics
2016

Abstract

Evaluating Scalable Markov Chain Monte Carlo Algorithms for Big Data Problems

By Zhifan Sang

Advances in technology have led to generation of enormous amounts of data in many fields including medicine, presenting challenges in data analysis. The reason could be processor, memory, or disk storage bottlenecks in computational environments. These challenges are particularly pronounced for Bayesian analysis, which often entails the use of Markov chain Monte Carlo (MCMC) in computation. As such, a number of scalable MCMC algorithms have been developed to alleviate the computational challenges in three main directions. The first direction is to accelerate expensive gradient computation at each MCMC iteration. The second direction is to parallelize computation at each MCMC iteration, requiring potentially expensive communication within each iteration. The third direction is to divide a data set into small subsets and run independent MCMC for each subset before combining them. This work focuses on scalable MCMC algorithms developed in the third direction. We conduct simulation studies to evaluate and compare several parallel MCMC algorithms. Examples of scalable MCMC are shown for Bayesian linear regression and other regressions.

Evaluating Scalable Markov chain Monte Carlo Algorithms for Big Data Problems

By

Zhifan Sang

B.S., Nankai University, 2014

Thesis Committee Chair:
Qi Long, Ph.D.

A thesis submitted to the Faculty of the
Rollins School of Public Health of Emory University
in partial fulfillment of the requirements for the degree of
Master of Science in Public Health
in Department of Biostatistics and Bioinformatics
2016

Table of Contents

1. Introduction

1.1 Background

Big data often refer to large and complex data sets which cannot be appropriately processed by traditional data processing applications. The volume, velocity, and variety of data significantly increase because of the modern technology of data collecting, processing and storing. For example, business, medical, governmental and web data sets with millions of entries or thousands of attributes can be quite large and hard to manipulate and analyze.

There are challenges in conducting Bayesian inference, one of the popular analytical tools, on big data. Traditionally, Bayesian inference can produce efficient and robust results in parameter estimations in various models. Complex posteriors in Bayesian inference are commonly approximated by Markov chain Monte Carlo approach (MCMC). MCMC is a powerful simulation algorithm highly used in Bayesian inference to approximate an intractable posterior function by random sampling, which is useful for computing integrals or optimizing functions in large-dimensional spaces. In the context of statistics and machine learning, it translates into parameter estimation, prediction, and model selection. One of the challenges of Bayesian computation is that the examination of all the data is required to evaluate a new hypothesis regarding parameters or latent variables in the coherent procedure of inference. For instance, it is necessary to evaluate the target posterior density for each proposed parameter update based on whole dataset iteratively when we perform Metropolis-Hastings (MH).

Regular MCMC algorithms store all the data into the memory and disk. In other words, the method may fail frequently when the data sets and the temporarily generated data are larger than or equal to the volume of memory or disk, which is common in big data settings. Even if all data can be fitted in the hardware framework, standard MCMC methods are still too computationally expensive to be used for inferences on large datasets, which contain a large number of observations or a large number of parameters.

1.2 Current Approaches

There has been considerable interest in developing scalable MCMC algorithms for big data problems, which mostly falls into three main groups.

1.2.1 Nonparallel Accelerating Approach

The first strategy uses approximation methods like stochastic gradient process based on data or data subsets in order to accelerate the expensive gradient calculations in Monte Claro (Wang, X., Dunson, D. B. ,2013). This stagey can be implemented for a number of MCMC methods, namely Metropolis-Hastings (MH) Algorithm, Langevin Monte Carlo(LMC), and Hamiltonian Monte Carlo (HMC) (Neal, 2010; Welling and Teh, 2011; Ahn et al., 2012). It only requires a single machine, but the resulting parameter estimates can be unstable or even biased. There are mainly three related methods, which can scale up Metropolis-Hastings approach.

One method is the Firefly Monte Carlo (FlyMC) (Maclaurin and Adams, 2014). FlyMC is an exact MCMC procedure utilizing only a subset of the data at each iteration, and it keeps the full-data posterior invariant. The main idea of FlyMC is using a set of binary auxiliary variable to

control whether or not the data points are included in the calculation at each iteration. To ensure the validity of FlyMC, lower bounds for each likelihood term are used.

Another method is the Austerity framework (Korattikara et al., 2014). This method uses approximate accept and reject steps to accelerate Metropolis-Hasting algorithm. It is known to lead to biased result, since it samples from an approximate posterior distribution. Although it yields a bias estimate calculating from the biased posterior, it can obtain a lower mean square error (MSE) of Monte Claro estimator using the accept and reject steps. Under same situation, this method can collect more samples with the same computation cost.

The third method is the Metropolis-Hastings with subsampled likelihoods (MHSubLhd) (Bardenet et al., 2014). This method uses concentration inequalities to approximate the Metropolis-Hastings test, which evaluates the likelihood ratio and determines whether to accept proposed parameter values or not (Angelino, E., et al, 2016). It calculates the exact confidence intervals for the Monte Claro approximated parameter estimates, and quantifies the precision of the parameter estimates through concentration inequalities.

As for the differences among the aforementioned three methods, the FlyMC produces exact Monte Claro estimates whereas other methods produces approximated estimates. The downside of the FlyMC is the difficulty to find an appropriate lower bound on the likelihood. The Austerity method obtains an approximate confidence interval of parameter estimates, whereas the MHSubLhd obtains an exact confidence interval with some prior knowledge on the data. However, the MHSubLhd method could make a wrong accept/reject decision since the assumptions of the data and the predetermined boundary constraint could be inappropriate.

1.2.2 Communication-Intense Parallel Approach

The second strategy is to parallelize computation of likelihood function within each MCMC iteration. At each iteration, this approach conducts separate computation for mini batches, but requires intensive communications.

There are two computation schemes to achieve parallelism within each iteration. One scheme uses a multi-machine computation environment, and the other uses multi-core processing units.

The first scheme takes advantage of a multi-machine computation environment. Because of conditional independence of the partial likelihood function, we can conduct the calculation independently for predetermined data subsets stored on different machines and then combine the results from all machines in a central machine (master node) to obtain the full likelihood. As such, computational challenges associated with disk storage and memory are alleviated in this implementation. However, the communication among machines is needed to obtain the full likelihood function of interest, incurring communication cost that can be expensive.

The second scheme is to divide the workload among multiple cores on one chip. This local parallelism can be achieved by using a multi-core central processing unit (CPU), or a massively parallel graphics processing unit (GPU). While this scheme is relatively efficient and cheap, it has two major limitations. First, it cannot handle an oversized dataset that exceeds the limitation of random-access memory (RAM) or disk storage. Second, it may encounter difficulty in multi-threaded programming, which is needed for multi-core computation. For instance, GPU

programming with CODA code is difficult to abstract to high-level programming since it requires low-level memory management.

1.2.3 Communication-Free Parallel Approach

A third approach is motivated by the independent product equation (Wang and Dunson, 2013). The key idea is to divide a dataset into independent subsets, for which parallel computation can be used. Although MCMC is sequential by nature, it can be parallelized by running separated MCMC chains on different data subsets first. The main challenge is how to combine posterior samples obtained from separated MCMC on data subsets such that the combined posterior samples converge to the true posterior distribution in a precise fashion. To this end, a number of communication-free parallel MCMC algorithms have been developed. This approach solves both limited memory/disk problem by taking advantage of scalable memory, disk, and processor power produced by multi-machine systems. Meanwhile, the data are divided into multiple machine so that they can run the analysis separately and combine the result in the very last step. This parallel scheme is often known as the embarrassingly parallel method. The embarrassingly parallel procedure reduces the network traffic among machine to minimum and hence reduces communication cost.

To fix ideas, suppose there are N independent data points denoted by $\boldsymbol{x}$, and the parameter of interest is denoted by θ. The goal is to estimate the posterior distribution of θ

$$p(\theta|\boldsymbol{x}) \propto p(\theta)p(\boldsymbol{x}|\theta)$$

Suppose the original data are split into S data subsets and let x_s denote the s^{th} subset.

$$p(\theta|\boldsymbol{x}) \propto p(\theta)\prod_{s=1}^{S} p(\boldsymbol{x_s}|\theta) = \prod_{s=1}^{S} p(\theta)^{\frac{1}{S}} p(\boldsymbol{x_s}|\theta).$$

The subposterior $p(\theta|x_s)$ can be represented by

$$p(\theta|\boldsymbol{x_s}) \propto p(\theta)^{\frac{1}{S}} p\,.$$

It follows that $p(\theta|\boldsymbol{x}) \propto \prod_{s=1}^{S} p(\theta|\boldsymbol{x_s})$, which is the basis for combining subposterior samples.

The communication-free MCMC algorithms can be further divided into to three groups, namely parametric methods, nonparametric methods and semi-parametric methods.

The parametric methods recombine subposterior samples to approximate the full posterior if the sample follows a normal distribution. The central limit theory for Bayesian statistics, Bernstein-von Mises Theorem, guarantees the validity of this method. Given certain regularity conditions and the data realized from unique true parameter θ_0, the posterior can be approximated by $N\left(\theta_0\,, I^{-1}(\theta_0)\right)$ ($I(\theta)$ is the Fisher Information).

For instance, Neiswanger et al. (2013) assume a Normal distribution and combine subposteriors by the product of Gaussian to obtain approximated full posterior. Let $\hat{\mu}_s$ and $\hat{\Sigma}_s$ denote the sample mean and sample variance of the s[th] data subset. The distribution of each posterior can be approximated by $N(\hat{\mu}_s, \hat{\Sigma}_s)$.

The full posterior can then be calculated by multiplying these subposterior estimates $p_1 \dots p_S(\theta) \propto p(\theta|\boldsymbol{x})$.

$$\widehat{p_1 \dots p_S}(\theta) = \widehat{p_1} \dots \widehat{p_S}(\theta) \propto N\left(\theta\middle|\hat{\mu}_S, \hat{\Sigma}_S\right), \quad where$$

$$\widehat{\Sigma} = \left(\sum_{s=1}^{S} \widehat{\Sigma}_s\right)^{-1}, \qquad \hat{\mu} = \widehat{\Sigma}\left(\sum_{s=1}^{S} \widehat{\Sigma}_s^{-1}\,\hat{\mu}_s\right) \qquad (*)$$

A similar approach is proposed by Scott et al.(2013), which combines samples using averaging (known as the consensus Monte Carlo). Let θ_{sj}be the jth sample from subposterior s, and W_s be the subposterior weight for subposterior s. The j^{th} draw from consensus approximation of the posterior can be calculated by:

$$\hat{\theta}_j = \left(\sum_{s=1}^{S} W_s\right)^{-1} \sum_{s=1}^{S} W_s\theta_{sj}\,, \qquad where\ W_s = Var(\theta|x_s)$$

The approaches of Scott and Neiswanger are fast and easy to converge if the model is close to Gaussian. However, these two approaches do not perform well on non-Gaussian data since their prediction are based on the normal assumption.

One extension of the consensus Monte Carlo approach is using averaging to relax the aggregation restriction; see Rabinovich et al.(2015). Let $F(\theta_1, \dots, \theta_S)$ be the aggregation function, $\theta_1, \dots, \theta_S$ be draws from each subposterior. We have:

$$F(\theta_1, \dots, \theta_S) = \left(\sum_{s=1}^{S} W_s\right)^{-1} \sum_{s=1}^{S} W_s\theta_s$$

However, this improved method needs to add an optimization step, increasing computational costs.

If the subposteriors cannot be approximated by Gaussian because the distribution is far away from a normal distribution or the sample size is not large enough, the aforementioned parametric methods are questionable. An alternative strategy is to use nonparametric methods such as the one suggested by Neiwanger et al.(2013), where a kernel density estimator is used to approximate the full posterior. Let $x_1, \dots, x_N$ be p-dimension sample from density f. The kernel density of f at point x is:

$$f(x) = \frac{1}{N}\sum_{i=1}^{N} K_H(x - x_i),$$

$$where\ K_H(x) = |H|^{-\frac{1}{2}} K\left(H^{-\frac{1}{2}}x\right), \qquad H\ is\ a\ symmetric, postivie - definite\ matrix$$

The kernel function can be chosen from Gaussian or Non-Gaussian. To determine the accuracy of a kernel density estimate, we have to take bandwidth into account since it controls the estimate's smoothing.

To illustrate the idea of kernel function in nonparametric methods, Neiswanger et al. (2013) proposes an approach using Gaussian kernel with a diagonal bandwidth matrix h^2I to estimate the subposterior. The approximated estimation is given by:

$$\hat{p}_s(\theta) = \frac{1}{M}\sum_{m=1}^{M} N(\theta|\theta_{ms}, h^2 I),$$

$where\ \{\theta_{ms}\}_{m=1}^{M}\ sample\ from\ subposterior\ s, M\ is\ the\ number\ of\ samples\ in\ the\ data\ subset.$

The posterior is the product of estimated subposteriors:

$$p(\theta|x) = \prod_{s=1}^{S} \hat{p}_s(\theta) = \frac{1}{M^S} \prod_{s=1}^{S} \sum_{m=1}^{M} N(\theta|\theta_{ms}, h^2 I).$$

These algorithms yield good performance for non-Gaussian models, but their limitation is the poor performance of the kernel density estimation for high dimension data, also known as the curse of dimensionality.

To accelerate the convergence of the nonparametric method Neiswanger et al. propose a semiparametric estimator for each subposterior.

$$\hat{\mathrm{p}}_{\mathrm{s}} = \hat{f}_s(\theta)\hat{r}(\theta), \text{where } \hat{f}_s(\theta) = N\left(\theta|\hat{\mu}_s, \hat{\Sigma}_s\right), r(\theta) = \frac{p_s(\theta)}{\hat{f}_s(\theta)}.$$

The $\hat{\mathrm{r}}(\theta)$ represents the nonparametric estimator of $\mathrm{r}(\theta)$.

Given the Gaussian kernel for $r(\theta)$ suggested by Neiswanger et al. (2013), we can derive

$$p_s(\theta) = \frac{1}{M} \sum_{m=1}^{M} \frac{N\left(\theta|\theta_{m,s}, h^2 I\right) N\left(\theta|\hat{\mu}_s, \hat{\Sigma}_s\right)}{\hat{f}_s\left(\theta_{m,s}\right)} = \frac{1}{M} \sum_{m=1}^{M} \frac{N\left(\theta|\theta_{m,s}, h^2 I\right) N\left(\theta|\hat{\mu}_s, \hat{\Sigma}_s\right)}{N\left(\theta_{m,s}|\hat{\mu}_s, \hat{\Sigma}_s\right)}.$$

Full posterior $p\left(\hat{\theta}|x\right)$ can be estimated by the product of subposterior estimates

$$\Sigma_L = \left(\frac{S}{h} I + \hat{\Sigma}^{-1}\right)^{-1}, \qquad \mu_L = \Sigma_L \left(\frac{S}{h} I\theta_L + \hat{\Sigma}^{-1}\hat{\mu}\right), where\ \hat{\mu}\ and\ \hat{\Sigma}\ are\ defined\ in\ (*).$$

This method combines the parametric component and nonparametric method. If h approaches zero, the semiparametric estimator $\hat{\mu}\ and\ \hat{\Sigma}$ approach the corresponding nonparametric estimators. Except for solving the slow convergence problem of nonparametric methods, this semiparametric method is not well studied in terms of the balance between accuracy and speed.

Wang and Dunsin (2013) argued that methods proposed by Neiswanger et. al (2013) may suffer from the curse of dimensionality in the number of subsets; degeneration of tail distribution; mode misspecification and support inconsistency. These aforementioned aspects need further investigation to determine their validation.

To avoid some of the drawbacks of Neiswanger et. al (2013), Wang and Dunson(2013) proposed an alternative approach of using multi-tiered Gibbs sampler to simulate posterior from the Weierstrass transform's product. In this way, the parameters are simulated from the controlled kernel, and there is only one-time parameter simulation. In this sampling approach, the simulated subposteriors are combined into a normal proposal on θ, which is accepted with a probability depending on the subposterior simulations.

More recently, Wang and Dunson (2015) proposed to use random partition trees to parallelize MCMC(PART). This method is distribution-free, and it is easy to resample from the raw data. In addition, the PART algorithm can be easily and potentially adopted to multiple scales, which Neiswanger et. al (2013) could not achieve. Thus, the PART method using similar embarrassingly parallel MCMC baseline scheme to avoid the limited approximation accuracy and resampling difficulties that Neiswanger et. al (2013) has.

Another alternative to parametric/nonparametric combination approaches is utilizing barycenters of subposteriors without introducing kernel or tuning parameters. Srivastava et al (2015) uses Wasserstein barycenter of subposteriors in the combination step along with the same embarrassingly parallel MCMC scheme. The result of Wasserstein posterior(WASP) have good properties such as atomic form, straightforward estimation. This method is scalable, and it does not require specifying the associated probability density functions, which requires examination on the data.

Instead of implementing single layer of parallel MCMC chains, Martino, L., et al (2015) propose orthogonal parallel MCMC(O-MCMC) to employ multiple parallel MCMC chains with additional layer of MCMC techniques. Generally the O-MCMC scheme consists of vertical parallel MCMC chains sharing information by horizontal MCMC methods. To be specific, the horizontal MCMC employ independent proposal on the parameter θ while the vertical chains are based on random-walk proposals. Additional algorithms like parallel simulated annealing is utilized in the O-MCMC optimization. The study shows that the O-MCMC implementation reduce the computation cost of parallel Metropolis chains approach.

There are some approaches to accelerate the accept/reject step by modifying the mechanism of rejection in MCMC. For instance, Quiroz, M. (2015) utilizes the procedure of delayed acceptance in the MH algorithm to speed up the MCMC. The acceptance decision is processed in two stage in this method. The first stage evaluates conditional likelihood of interested parameter only from a random subsample. The second stage evaluates the likelihood using the full sample, and this stage is executed only the proposed posterior density accepted in the first stage. Hence, the computation is highly reduced because of the early rejection of proposals (Payne and Mallick, 2014). Beside the basic delayed acceptance procedure, Quiroz, M. (2015) also incorporates auxiliary information about the full sample while only evaluating based on data subset.

Banterle, M., et al. (2014) also modify the accept/reject step to reduce the computational cost by multiple acceptance steps. Divide-and-conquer strategy is employed to conduct multiple acceptance procedure. The prior and likelihood product can be decomposed into product components, some of which are cheaper with regards of computational cost. The components are compared with uniform random variate sequentially, and stop after first rejection to avoid further time consumption.

1.3 General Comparisons

We compare aforementioned three approaches with regard of their methodology and performances over different settings. Tables 1 and Table 2 present a summary of parallel MCMC algorithms and a summary of nonparallel MCMC algorithms, respectively.

Most of the methods are designed to be parallel MCMC algorithms, which reduce the computation cost to accommodate the rapid increasing data volume. The Scott, S. L. et al. (2013) address the practical idea of parallel MCMC implementation, which was followed by the statistical introduction of embarrassingly parallel MCMC (Neiswanger et. al, 2013). Most recent related works treat the two papers as start point or baseline comparison standard as the scheme and methodology of the embarrassing parallel MCMC is intuitive, straightforward, and practical. Multiple alternatives for embarrassingly parallel MCMC are developed to address its limitations as well as providing new methods to generate and sample posteriors (Wang,X. et al, 2013, 2015, Minsker, S. et al., 2014, Srivastava, S.,et al.,2015). There are packages in Matlab, R, Python developed to implement these methods in different settings.

Other novel techniques like delayed acceptance, multiple try, random walk have been adopted to accelerate the updating process of MCMC (Quiroz, M. ,2015, Martino, L., et al,2015, Banterle, M., et al. ,2014).

There are some recent works on nonparallel optimization of MCMC methods to obtain better performance (Maclaurin and Adams, 2014, Korattikara et al., 2014, Banterle et al. 2014). Although the scalability of these methods are not as promising as the parallel algorithms, the ideas still counts as it is practical and useful in single-thread computation, and can be potentially used together with other methods.

Table 1: Comparison of Parallel MCMC Algorithms

Author	Algorithm	Method	Posterior Sampling	Package	Language	Parallel
Neiswanger, W., Wang, C., & Xing, E. (2013)	Simple Average	Average of subset	Approximate	parallelMC MCcombine	R	Yes
	MC Consensus parametric	Bernstein-von Mises	Approximate		R	Yes
	MC Consensus nonparametric	Independent Metropolis Gibbs(IMG) sampler	Approximate		R	Yes
	Semiparametric Consensus	Combine	Asymptotically exact		R	Yes
Minsker, S.,et al., & Dunson, D. B. (2014)	Median Subset Posterior	geometric median	Approximate	Mposterior	R	Yes
Wang, X., Dunson, D. B. (2013)	Weierstrass Sampler	Weierstrass transformation	Approximate	Weierstrass	R	Yes
Luengo, D.,et al. (2014)	Combine partial MMSE estimators	Combine partial MMSE estimators	Approximate	N/A	N/A	Yes
Martino, L.,et al. (2015)	Orthogonal parallel MCMC	Multiple Try Metropolis, Block Independent Metropolis, Simulated Annealing	Approximate	N/A	N/A	yes
Wang, X., & Dunson, D. B.(2015).	Random Partition Tree	Embarrassingly parallel MCMC, random partition tree	Approximate	github	Matlab	yes
Srivastava, S.,& Dunson, D. B. (2015).	WASP	Wasserstein posterior	Approximate	N/A	N/A	yes

Quiroz, M. (2015)	Delayed Acceptance and Data Subsample	delayed acceptance ,auxiliary information	Approximate	obtained	python	yes
Quiroz, M et al. (2015)	Data Subsamping and the Difference Estimator	highly efficient difference estimator, Pseudo-marginal MCMC	Approximate	N/A	N/A	yes
Nishihara, R.,et al. (2014)	Generalized Elliptical Slice Sampling	parallelism, slice sampling, elliptical slice sampling	Approximate	N/A	N/A	yes
Scott, S. L. et al. (2013)	consensus Monte Carlo	Average of subset posterior sample	Approximate	parallelMCMCcombine	R	yes

Table2: Comparison of Nonparallel MCMC Algorithms

Author	**Algorithm**	**Method**	**Posterior Sampling**	**Package**	**Lang**	**Parallel**
Maclaurin, D., et al. (2014)	FireFly	query subset	Exact	FireFly	Python	No
Banterle, M., (2014)	Delayed acceptance decomposition, prefectching	Delayed acceptance ,Likelihood decomposition, prefectching	Approximate	N/A	N/A	No
Angelino, E. et al. (2014)	prefectching	Fetching ,speculative execution to parallelize MCMC.	Approximate	fetching	python	No

2. Analysis of MCMC Algorithms and Computational Challenges

2.1 Overview

In Bayesian inference, we rely on the posterior $p(\theta|\boldsymbol{x}) \propto p(\boldsymbol{x}|\theta)p(\theta)$. In many applications, the posterior is intractable and we have to rely on approximations. A standard approach is to use MCMC where we construct a Markov chain with stationary distribution

$$p(\theta|x), \quad \theta = \theta_1, \theta_2, \theta_3, \ldots \text{ where } p(\theta_i|\theta_{i-1} \ldots \theta_1) = p(\theta_t|\theta_{t-1}) .$$

Calculate posterior expectations using a Monte Carlo estimate (unbiased)

$$E_{p(\theta|\boldsymbol{x})[f]} \approx \frac{1}{T}\sum_{t=1}^{T} f(\theta_t).$$

The data points $\boldsymbol{x} = \{x_1, \ldots, x_N\}$ are conditional independent given θ we can write the posterior as

$$p(\theta|\boldsymbol{x}) \propto p(\theta)p(\boldsymbol{x}|\theta) = p(\theta)\prod_{i=1}^{N} p(x_i|\theta).$$

If the data set is large (N $\gg$ 1), evaluating $p(\boldsymbol{x}|\theta)$ or $\nabla p(\boldsymbol{x}|\theta)$ is the computational bottleneck for most standard MCMC methods, including but not limited to Metropolis-Hastings (MH) Algorithm, Langevin Monte Carlo (LMC) and Hamiltonian Monte Carlo (HMC).

Here we focus on the Metropolis-Hastings algorithm and its computational challenge in big data settings. The MH algorithm uses the full joint density function and independent proposal distributions for each parameter of interest to generate posterior samples accordingly.

Algorithm 1 Metropolis-Hastings Algorithm

Initialize $\theta^{(0)} \sim q(\theta)$
for iteration i = 1,2, … do
  Propose: $\theta^{cond} \sim q(\theta^{(i)}|\theta^{(i-1)}, \boldsymbol{x})$
  Acceptance Probability:

$$\alpha\left(\theta^{cond}\middle|\theta^{(i-1)}\right) = \min\left\{1, \frac{q\left(\theta^{(i-1)}\middle|\theta^{cond}\right)p(\theta^{cond}|\boldsymbol{x})}{q(\theta^{cond}|\theta^{(i-1)})p(\theta^{(i-1)}|\boldsymbol{x})}\right\}$$

  $u \sim Uniform(u; 0,1)$
  if $u < \alpha$ **then**
    Accept the proposal: $\theta^{(i)} \leftarrow \theta^{cond}$
  else
    Reject the proposal: $\theta^{(i)} \leftarrow \theta^{(i-1)}$
  end if
end for

The first step of the Algorithm 1 initializes the sample for each random variable by prior distribution or other choices. The main iteration loop of MH algorithm consists of three parts: (1) generate proposal sample θ^{cond} from proposal distribution; (2) evaluate the posterior distribution and calculate the

acceptance rate α; (3) reject the proposal sample with probability of $1-\alpha$, or accept it with probability α.

The computation challenge mainly lies in the second part of the main loop iteration. In this part, evaluating the posterior density $p(\theta^{cond}|\boldsymbol{x})$ could be time consuming and memory consuming since it uses all the data in the computation. To alleviate this part of tremendous computation, the approaches introduced in chapter 1 have their technical solutions with regard of algorithm implementations.

2.2 Algorithms in Nonparallel Accelerating Approach

Nonparallel accelerate approach reduce the size of data used in the posterior evaluation by only using data subsets. In this way, the computational cost is reduced while compromising the accuracy of the full posterior.

For instance, the Firefly Monte Carlo algorithm use rejection sampling to avoid using all data in evaluating the posterior density(Algorithm 2, Maclaurin, D., & Adams, R. P. 2014). The actual sample used in evaluating the likelihood is constraint by auxiliary variable and lower bound of likelihood for each data points.

Algorithm 2 Firefly Monte Carlo

Set θ_0 by initial distribution
for i=1 to numberOfMarkovChain **do**
  for j=1 to N*T **do**
    $n \sim RandomInterger(1, N)$
    $z_n \sim Bernoulli(1 - \frac{B_n(\theta_{i-1})}{L} - L_n(\theta_{i-1}))$
  end for
  $\theta' \leftarrow \theta_{i-1} + \eta \; where \; \eta \sim N(0, \epsilon^2 I_D)$
  $u \sim Uniform(0,1)$
  if $\frac{jointPosterior(\theta';\{z_n\}_{n=1}^N)}{jointPosterior(\theta;\{z_n\}_{n=1}^N)} > u$ **then**
    $\theta_i \leftarrow \theta'$
  else
    $\theta_i \leftarrow \theta_{i-1}$
  end if
end for
function jointPosterior($\theta; \{z_n\}_{n=1}^N$)
  $P \leftarrow p(\theta) \prod_{n=1}^N B_n(\theta)$
  for each n a.s. $z_n = 1$ **do**
    $P \leftarrow P \times (\frac{L_n}{B_n(\theta)} - 1)$
  end for
  return P

end functions

2.3 Communication-Intense Parallel Algorithms

Communication intense approach parallel the computation of evaluating the conditional posterior within each MCMC iterations. Specifically, the conditional posterior distribution calculation $p(\theta)\prod_{i=1}^{N} p(x_i|\theta)$ could be calculated simultaneously on multiple machines. Each node (machine) only takes a data subset in the calculation, which reduce the time and memory cost by having multiple machines working on separated portion of the evaluation.

This approach could be implemented easily in the parallel computing environment because of the independent assumption on conditional probabilities. However, this method has poor performance in real network settings since the data flow in the network slows the whole process down significantly.

2.4 Communication-Free Parallel Algorithms

The communication free approach obtains the subposterior sample from data subset first, and combine the subposterior samples to obtain the full posterior samples. This approach avoids redundant communication over network and expensive and intensive computation within single machine. The difference within communication-free parallel algorithms is the techniques used in the combination step.

The simplest way of combination is to draw the posterior sample from the average of all subposterior samples. Since only sample average calculation is involved in this case, the algorithm is fast and scalable. However, this method usually overestimate the variance of the posterior, and it can hardly recover the true posterior distribution except for simple statistics like sample mean.

For instance, Algorithm 3 is asymptotically exact, embarrassingly parallel MCMC algorithm utilizes the communication free approach (Neiswanger, W., et al, 2013). It obtains the posterior samples from nonparametric density product estimate by combining subposterior samples.

Algorithm 3 Asymptotically Exact Sampling via Nonparametric Density Product Estimation

---

Input: Subposterior samples $\{\theta_{t1}^{1}\}_{t_1=1}^{T}\sim p_1(\theta), \dots, \{\theta_{t_M}^{M}\}_{t_M=1}^{T}\sim p_M(\theta)$

Output: Full posterior samples $\{\theta_i\}_i^T \sim p_1 \dots p_M(\theta|x^N)$

Set $h \leftarrow 1$
Generate **t**= $\{t_1, \dots, t_M\} \overset{iid}{\leftarrow} Unif(\{1,2,\dots,T\})$
Set c $\leftarrow t$
Generate $\theta_1 \sim N\left(\bar{\theta}_t, \frac{h}{M} I_d\right)$
for I =2 **to** T **do**
  for m=1 **to** M **do**
    Set $t \leftarrow c$
    Generate $t_m \sim Unif(\{1,2,\dots,T\})$
    Set $h \leftarrow i^{-1/(4+d)}$
    Generate $u \sim Unif([0,1])$
    if $u < w_t/w_c$ then
      Generate $\theta_t \sim N\left(\bar{\theta}_t, \frac{h}{M} I_d\right)$
      Set $c \leftarrow t$
    else
      Generate $\theta_t \sim N\left(\bar{\theta}_c, \frac{h}{M} I_d\right)$

---

end if
end for
end for

Another way to implement communication free approach is using Weierstrass refinement sampling. Algorithm 5 and Algorithm 6 show how to implement Weiserstrass sampling in the parallel MCMC process (Wang, X., Dunson, D. B. ,2013). Algorithm 5 assumes that we have the approximated density estimate of the parameter, so we can obtain the initial parameter by evaluating the approximated density functions. By contrast, Algorithm 6 illustrates how it samples sequentially with rejection when we do not have prior approximated density.

Algorithm 5 Weierstrass Refinement Sampling

Input $H_i \; for \; i = 1,2 \ldots m$
for k=1 to N **dos**
$\theta_k \sim \hat{f}(\theta)$
end for
for k=1 **to** N **do**
Sample $t_i^{(k)} = \left(t_{i1}^{(k)}, \ldots, t_{ip}^{(k)}\right)^T \sim dN(t_i|\theta_k, H_i) \cdot f_i(t_i)$
$T_i \leftarrow T_i \cup \left\{t_i^{(k)}\right\}$
end for
for k=1 **to** N **do**
$\theta \sim N\left(m^{-1}\sum_{i=1}^{m} t_i^{(k)}, \left(\sum_{i=1}^{m} H_i^{-1}\right)^{-1}\right)$
$MCMC \leftarrow MCMC \cup \{\theta\}$ #MCMC is the set of posterior samples with initial value of $\emptyset$
end for
return MCMC

Algorithm 6 Weierstrass Sequential Rejection Sampling

Input $N_0 \; h_j for \; j = 1,2 \ldots p$
Set $h_{ij} = \sqrt{m}h_j, C_j = 0, j = 0, \ldots, p-1$
for j=1 **to** p **do**
for t=1 **to** N_0 **do**
Sample $\theta_{ij}^{(t)} \sim \widehat{f}_l(\theta_j|\theta_1^*, \ldots, \theta_{j-1}^*)$
$T_i \leftarrow T_i \cup \left\{\theta_{ij}^{(t)}\right\}$
end for
Calculate θ_j^* by combining $\theta_{ij}^{(t)}, i = 1,2, \ldots, m$ via Weierstrass rejection sampling
Calculate C_{j-1} as $C_{j-1} = \int \prod_{i=1}^{m} \widehat{f}_l\left(\theta_j \middle| \theta_1^*, \ldots, \theta_{j-1}^*\right) d\theta_j$
$MCMC \leftarrow MCMC \cup \{(\theta_j^*, C_{j-1})\}$ #MCMC is posterior samples with initial value of $\emptyset$
end for

return MCMC

More sophisticatedly, the geometric median of subposterior sample distribution can be used to generate the full posterior samples (Minsker, S., et al, 2014). The Algorithm 7 define a function that can obtain the geometric median of a discrete measure, and iteratively use the function to evaluate weights and the final posterior density.

Algorithm 7 Approximating the Median-Posterior Distribution

Function geometricMedian(Q)

Weiszfeld's algorithm in evaluating the geometric median of probability distributions

Input discrete measure $Q_1, \dots, Q_m$

Input the kernel $k(\cdot,\cdot)$: $R^p \times R^p \to R$

Input threshold $\varepsilon > 0$

Set $w_j^{(0)} = \frac{1}{m}, j = 1, \dots, m;\ Q_*^{(0)} = \frac{1}{m}\sum_{j=1}^{m} Q_j;$ t=0

Do

for j=1 **to** m **do**

$$w_j^{(t+1)} = \frac{\left\|Q_*^{(t)} - Q_j\right\|_{F_k}^{-1}}{\sum_{i=1}^{m}\left\|Q_*^{(t)} - Q_j\right\|_{F_k}^{-1}}$$

$$Q_*^{(t+1)} = \sum_{j=1}^{m} w_j^{(t+1)} Q_j$$

until $\left\|Q_*^{(t)} - Q_j\right\|_{F_k} \le \varepsilon$

return $w_* = (w_1^{(t+1}, \dots, w_m^{(t+1)})$

Input samples $\{Z_{j,i}\}_{i=1}^{N_j} \sim \Pi_{n,m}(\cdot \mid G_j), j = 1, \dots, m$

Do

$Q_j = \frac{1}{N_j}\sum_{i=1}^{N_j} \delta_{Z_{j,i}}, j = 1, \dots, m$

apply function geometricMedian to $Q_1, \dots, Q_m$ return $w_* = (w_{*,1}, \dots, w_{*,m})$

for j = 1 **to** m **do**

$\bar{w}_j = w_{*,j} I\left\{w_{*,j} \ge \frac{1}{2m}\right\}$

define $\widehat{w_j^*} = \frac{\bar{w}_j}{\sum_{i=1}^{m} \bar{w}_i}$

end for

end do

return $\widehat{\Pi}_{n,g}^{st} = \sum_{1=1}^{m} \widehat{w_i^*} Q_i$

3. Simulation Study

3.1 Goals

In this section, we focus on communication-free parallel MCMC methods since it can handle huge volume of data efficiently. The performances of related methods discussed above are compared in identical simulation settings, and the main goal is to evaluate the quality of posterior samples as well as running time.

3.2 Simulation Settings

In the simulation study, we consider the data setup using a multivariate regression setting. We assume that $\boldsymbol{X}$ comes from multivariate normal distribution $MVN(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, the parameter of interest is denoted by $\beta = (\mathbf{1}, \mathbf{1}, \dots, \mathbf{1})_{\boldsymbol{p}}$, the random error is denoted by $\varepsilon_i \sim N(0,1), i = 1, \dots, N$. The response variable $\boldsymbol{Y}$ is generated using regression model as follows:

$$\boldsymbol{Y}_{n\times 1} = \boldsymbol{X}_{n\times p}\,\boldsymbol{\beta}_{p\times 1} + \boldsymbol{\varepsilon}_{N\times 1}.$$

We consider fitting the following multiple linear regression model using Bayesian approach. For $i = 1, \dots, n$, the decomposed model is:

$$y_i = \beta_1 x_{i1} + \beta_2\, x_{i2} + \ \dots + \beta_p x_{ip} + \varepsilon_i,\ \ \varepsilon_i \sim\ N(0, \sigma^2)$$
$$\boldsymbol{y}\ \sim\ N(\boldsymbol{X\beta}, \sigma^2 \boldsymbol{I}_{n\,\mathrm{x}\,n})$$

with likelihood

$$p(\boldsymbol{y} \mid \beta, \sigma^2) = (2\pi\sigma^2)^{-\frac{n}{2}} \exp[-1/((2\sigma^2)(\boldsymbol{y} - \boldsymbol{X\beta})'\ (\boldsymbol{y} - \boldsymbol{X\beta}).$$

The standard priors for linear regression parameters are non-informative multivariate normal for β and inverse-gamma for σ^2

$$\boldsymbol{\beta} \sim \mathrm{N}(\mu_0, \tau^2 \boldsymbol{I}_{p\times p}) \ \text{ and } \ \sigma^2 \sim \text{Inv-gamma}(c_0, d_0),$$

where τ^2 is set to be large, and c_0, d_0 are set to be small, μ_0 is set to be 0. Assuming the calculation consists of S subposterior samples, the parameter estimate $\hat{\beta}$ is obtained from the posterior distribution which can be calculated as below.

$$p(\beta|\boldsymbol{x}) \propto p(\beta) \prod_{s=1}^{S} p([\boldsymbol{x}, \boldsymbol{y}]_s|\beta) = \prod_{s=1}^{S} p(\beta)^{\frac{1}{S}} p([\boldsymbol{x}, \boldsymbol{y}]_s|\beta).$$

For comparison purposes, we use MCMCpack to generate posterior of linear regression coefficients with Gaussian errors using Gibbs sampling. We use multivariate Gaussian prior on the beta vector and an inverse Gamma prior on the conditional error variance. The estimate result from of the standard MCMC is set to be the golden standard. Other combined parallel MCMC results are compared to the golden standard based on standard MCMC.

In section 1.2, we discuss three groups of approaches to obtain the full posterior samples of high volume of data. The approaches in section 1.2.2 and 1.2.3 involve parallel computing on data subsets. In order to compare the performance among each categories with emphasis on communication-free methods, we choose one method from section 1.2.1, one method from section 1.2.2, and four methods from section 1.2.3 for the simulations. The methods we use in the simulation are Firefly method, communication-intense parallel method, Embarrassingly Parallel method, parallel Weierstrass Sampling method, parallel Random Partition Trees method, Parallel Predictive Prefetching method.

The simulation study compares different parallel MCMC algorithms in terms of their performance as the following scenarios change: sample size, number of sample subsets, dimension of sample, the property of generated sample (correlation matrix parameter, standard deviation). The number of iterations and the

number of the burn-in are predetermined by preliminary attempts. The effective sample size is calculated using full posterior sample in order to compare performance of different approaches.

In all simulations, the number of MCMC iterations and the number of burn-in are predetermined by preliminary experiments. The sample size N is chosen from 50 thousands, 1 million or 10 million. The number of MC datasets (K) is 500. The number of data subsets (M) are 1, 10, 20 and 50, and we fix p to 5.

All the examples are coded in R and Matlab. The Bayesian linear regression examples are run on a cluster composed of 4-cores (Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30GHz) nodes, using up to 8 nodes for a total of 32 cores, and make use of Open-MPI and R package "parallel" for communications among cores. The standard MCMC example is run on a one core (Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30GHz) on a single machine.

The basic workflow of the simulation is summarized in Figure1. First we generate data with different correlation parameters and sample size. In parallel MCMC cases, we split the data, calculate subposteriors of data subsets, and combine them using one of the combinations techniques to obtain full posteriors. In other cases, we conduct the MCMC algorithms on the full data and obtain the posteriors. The last step is to examine and compare the summary statistics of full posteriors obtained from previous steps.

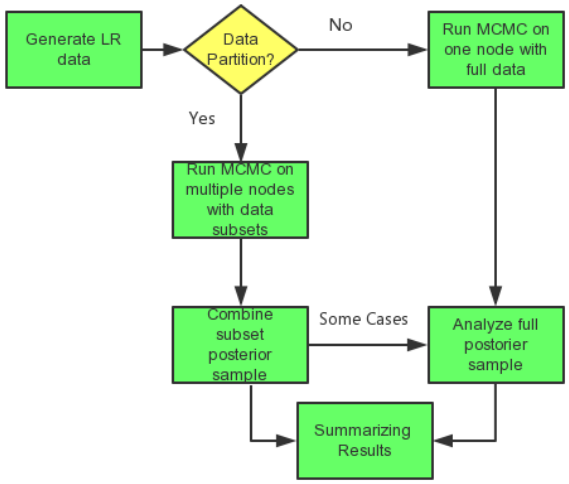

Figure 1: Pipeline of Parallel/Nonparallel MCMC Simulations

3.3 Performance Metrics

The statistical inference measures of the parallel MCMC algorithms we implemented are listed below:

- Bias of i[th] parameter posterior $\hat{\beta}_i$:$Bias_i = \frac{1}{S}\sum_{s=1}^{S}(\hat{\beta}_{is} - \beta_i)$
- Mean standard error(SE) of $\hat{\beta}_i$: $SE_i = \frac{\sum_{s=1}^{S} SE_{is}}{S} = \frac{\sum_{s=1}^{S}\sqrt{\hat{\sigma}_i^2}}{S}$
- Monte Carlo standard deviation (SD) of $\hat{\beta}_i$: $SD_i = \frac{\sqrt{\sum_{k=1}^{K}(\hat{\beta}_{ik} - \overline{\beta_i})^2}}{\sqrt{K}}$
- Mean Square Error(MSE) of i[th] parameter posterior $\hat{\beta}_i$:$MSE_i = \frac{1}{K}\sum_{k=1}^{K}\left(\hat{\beta}_{ik} - \beta_i\right)^2$, where $\hat{\beta}_{ij}$ is the parameter estimates for parameter i from j[th] MC dataset.

- Coverage rate(CR_i): The probability of 95% confidence interval for $\hat{\beta}_i$ that contains true value

The running performance of algorithms we implement is measured by the following metrics:

- Effective sample size (ESS)
- Running time

Subposterior sample average (sample average), Consensus Monte Carlo Algorithm for correlated parameters (consensusMCcov), Consensus Monte Carlo Algorithm for independent parameters (consensusMCindep), parallel Weierstrass Sampling method, parallel Random Partition Trees method (PART) are used in the simulation.

3.4 Results

Table 3 and Figure 2-6 summarize the simulation results from the experiment. In general, The MC consensus algorithms generally perform quite well in low dimensional scenarios (p=5), as it has relatively low bias and standard error as well as largest confidence interval coverage rate. The Weierstrass sampling method has good performance regarding bias and SE, which is better than PART, and close to MC consensus method in this case. The PART method have less promising result compared with other methods since it has larger bias and standard error with less effective sample size. This shortage can be explained by the compromise made to boost the speed of algorithm.

For the small sample scenario (N=50,000), standard MCMC method utilizing all the sample generally has the best performance, and it is set to be golden standard for comparison purpose. As the number of subset increases, biases of all the parallel MCMC methods increases gradually; the SE,SD and MSE remain similar level. As the correlation parameter ρ increases, the bias of all methods increases, and the biases of PART method increase significantly. The bias, SE, SD of sample average, MC consensus methods are similar. When the number of subsample is large, Weistrass method achieve the lowest bias and SE. The SD is the standard to which we compare SE, and the values of SD and SE are very similar in value. The ESS of all methods except for PART are close to the posterior sample size 10,000 while the PART method has a significantly small ESS of 300-500. The MSE of MC consensus, sample average, and Weistrass method are generally small and close to the MSE of standard MCMC method, while PART has a relatively high MSE.

As for the 95% coverage rate, consensus MC and sample average almost always has coverage rate of 1, whereas PART and Weierstrass have relatively reasonable coverage rate (around 0.95). The standard MCMC also has a 95% CR coverage rate near 0.95. Since it is hard to estimate the distribution of sample using simple methods like MC consensus and sample average, even though it is easy to obtain the mean estimates. These consensus MC methods tend to overestimate the variance of posteriors.

For the large sample scenario (N=1,000,000), the ascending trend of bias with increasing number of data subset is also observed. As the correlation parameter ρ increases, the bias of all methods increases, and the biases of PART method increase significantly. We observed the performance of MC consensus (Consensus Monte Carlo Algorithm), sample average and Weierstrass have similar performance with regard of SD, SE, MSE as they have in small sample scenario.

As for the running performance, sample average method has the shortest average running time. This method simply takes the average of all subposteriors in the combination step, which uses a relatively

small amount of computation resources. PART method generally have the second shortest average running time. The Weierstrass has the longest average running time in general, which is more than 10 times slower than sample average and PART method. The original MCMC actually runs faster than Weierstrass method in small sample size case (N=50,000), but it runs significantly slower than parallel MCMC methods when sample size is large (N=1,000,000). The standard MCMC takes a long time to calculate posterior likelihood of large amount of sample, which is the major time cost. By contrast, other MCMC algorithms with embarrassingly parallel scheme utilize multiple machines to run the likelihood calculation on data subsets simultaneously. This approaches usually save a considerable amount of time, while the time consumption of combination step is relatively small in large sample scenarios.

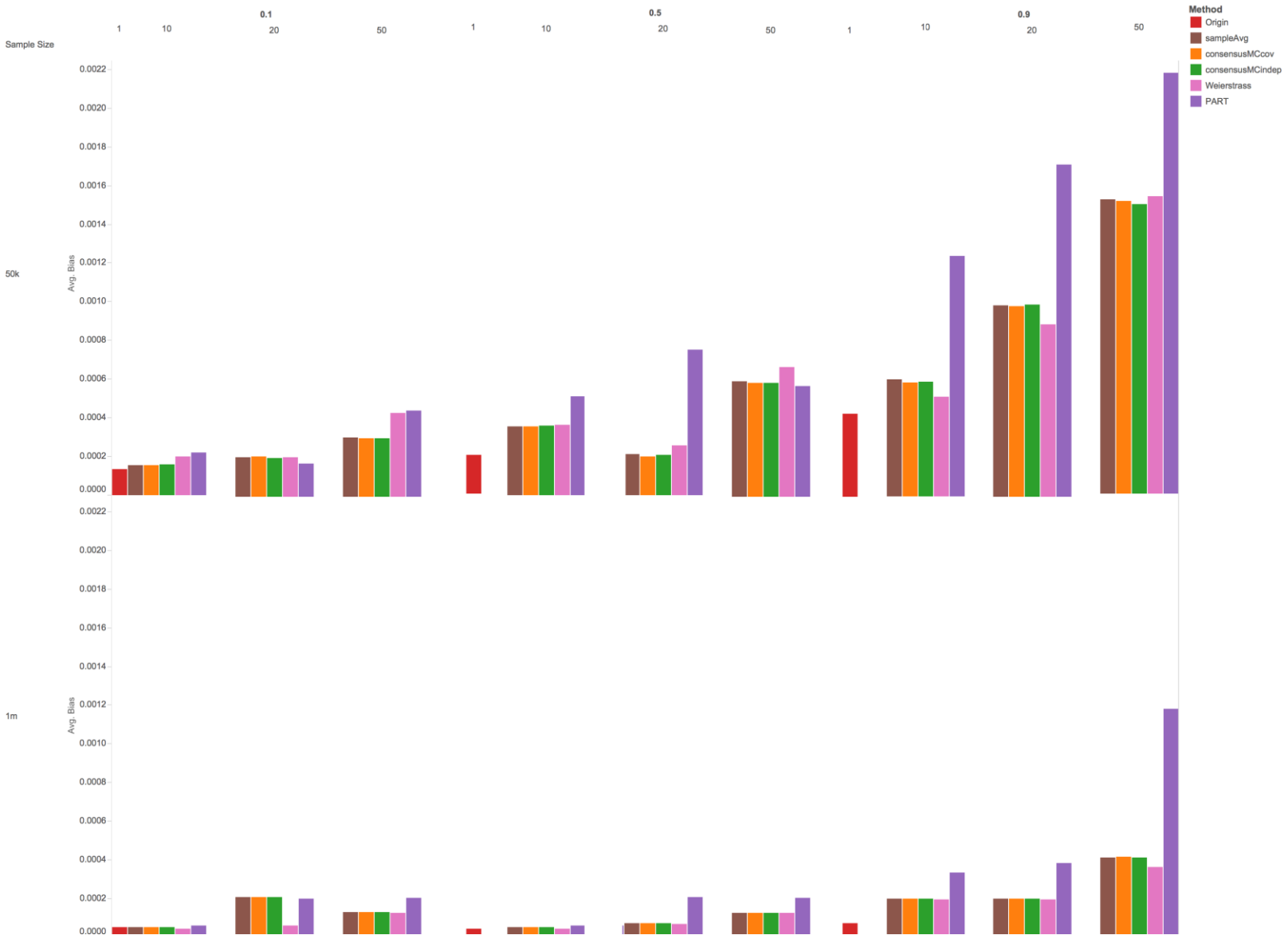

Figure 2: Simulation Result Comparison of Parallel MCMC (Bias)

Original: Single chain full-data posterior sample, sample average :Subposterior sample average, consensusMCcov: Consensus Monte Carlo Algorithm (for correlated parameters), consensusMCindep: Consensus Monte Carlo Algorithm (for independent parameters) , parallel Weierstrass Sampling method, PART: parallel Random Partition Trees

Figure 3: Simulation Result Comparison of Parallel MCMC (SE and SD)

Original: Single Original: Single chain full-data posterior sample, sample average :Subposterior sample average, consensusMCcov: Consensus Monte Carlo Algorithm (for correlated parameters), consensusMCindep: Consensus Monte Carlo Algorithm (for independent parameters) , parallel Weierstrass Sampling method, PART: parallel Random Partition Trees

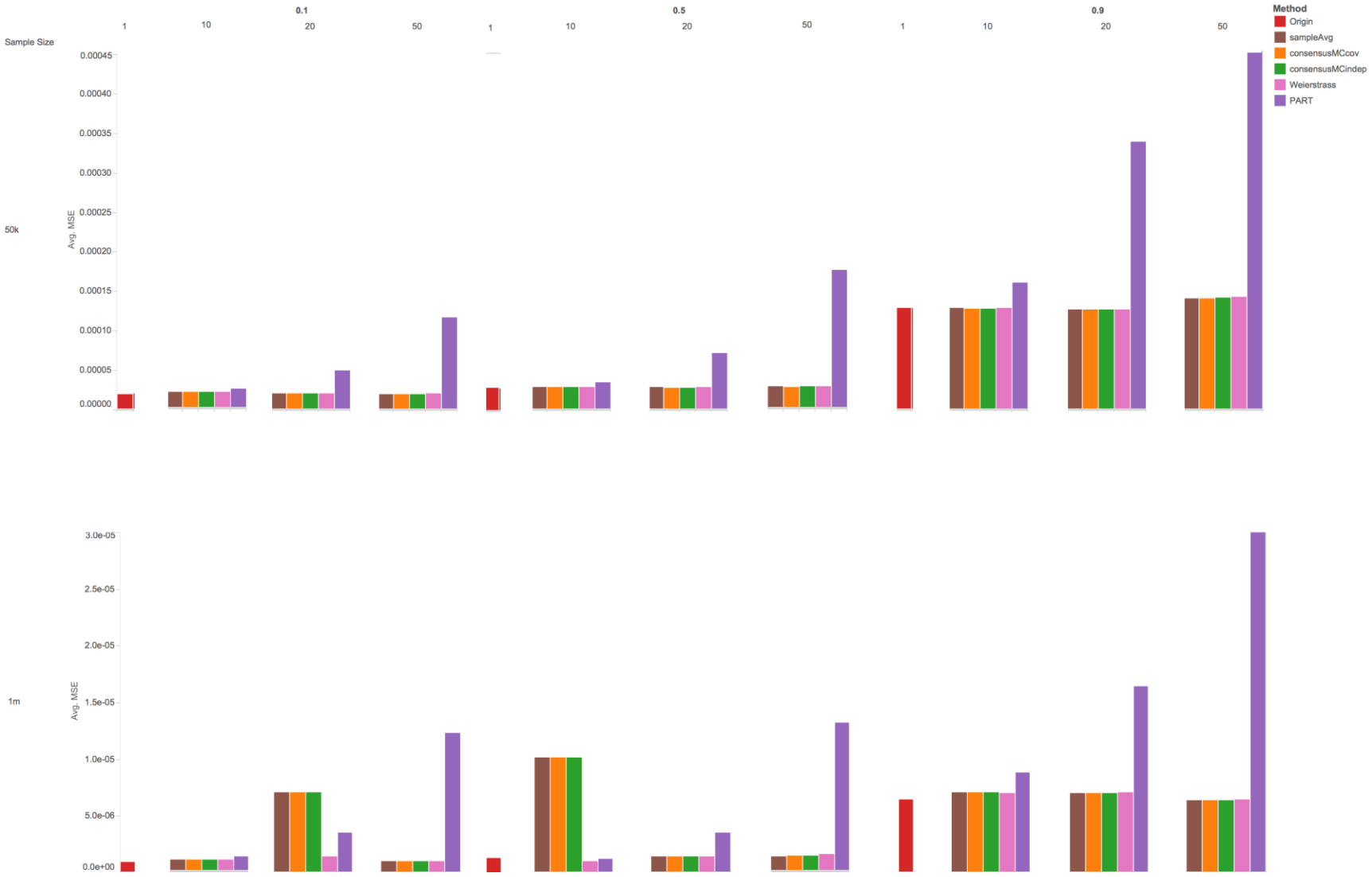

Figure 4: Simulation Result Comparison of Parallel MCMC (MSE)

Original: Single chain full-data posterior sample, sample average :Subposterior sample average, consensusMCcov: Consensus Monte Carlo Algorithm (for correlated parameters), consensusMCindep: Consensus Monte Carlo Algorithm (for independent parameters) , parallel Weierstrass Sampling method, PART: parallel Random Partition Trees

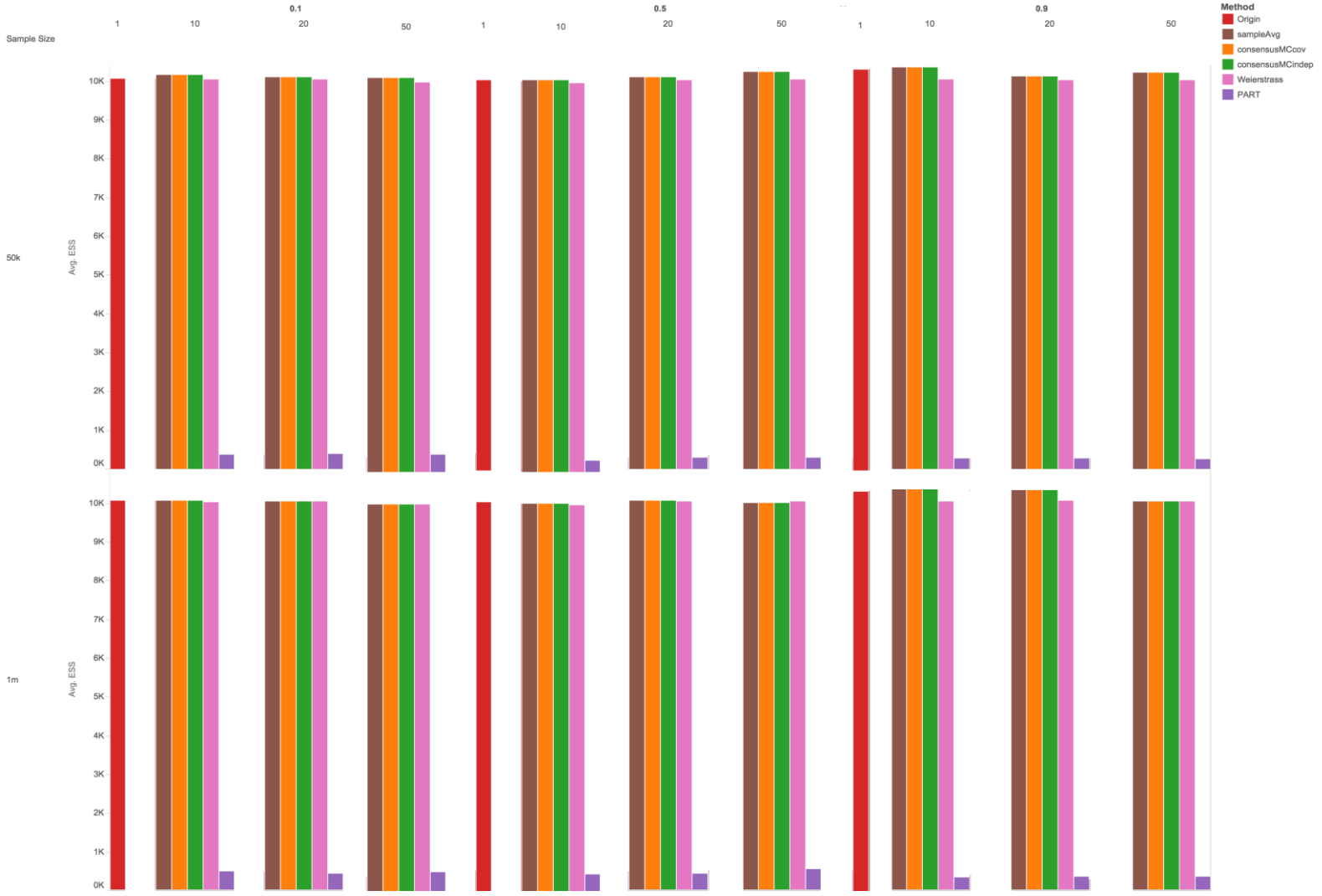

Figure 5: Simulation Result Comparison of Parallel MCMC (ESS)

Original: Single chain full-data posterior sample, sample average :Subposterior sample average, consensusMCcov: Consensus Monte Carlo Algorithm (for correlated parameters), consensusMCindep: Consensus Monte Carlo Algorithm (for independent parameters) , parallel Weierstrass Sampling method, PART: parallel Random Partition Trees

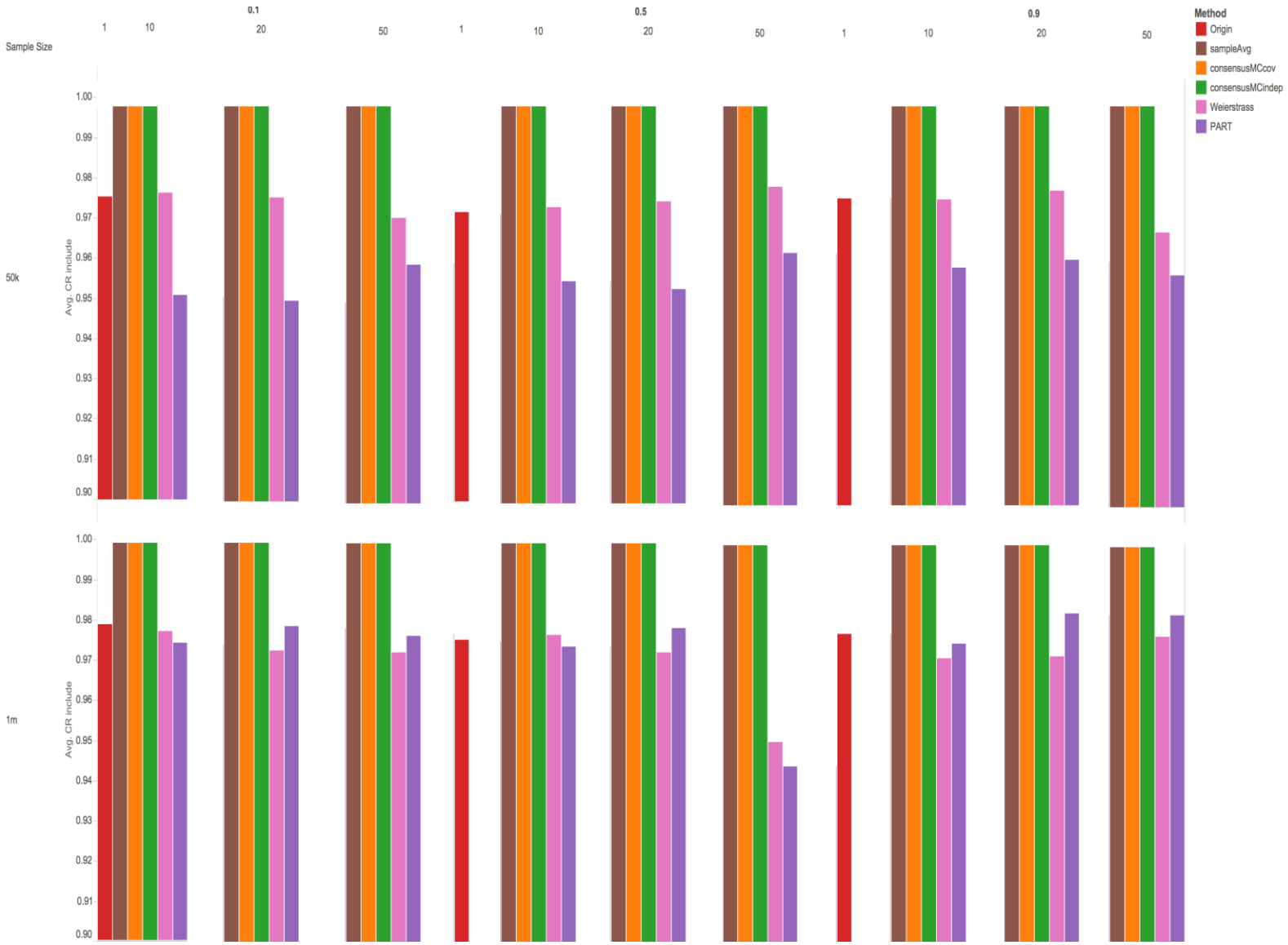

Figure 6: Simulation Result Comparison of Parallel MCMC (CR coverage rate)

Original: Single chain full-data posterior sample, sample average :Subposterior sample average, consensusMCcov: Consensus Monte Carlo Algorithm (for correlated parameters), consensusMCindep: Consensus Monte Carlo Algorithm (for independent parameters) , parallel Weierstrass Sampling method, PART: parallel Random Partition Trees

Table 3: Simulation Running Time of Parallel MCMC (combination stage only)

			Rho			Sum Time
Method	Sample Size	Data subset	0.1	0.5	0.9	0.1 – 131.2
Origin	50000	1	1.0	3.9	1.1	
	100000	1	131.1	61.2	131.2	
sampleAvg	50000	10	0.1	0.1	0.1	
		20	0.1	0.1	0.1	
		50	0.1	0.1	0.1	
	100000	10	12.3	2.8	12.5	
		20	1.0	1.0	5.9	
		100	1.5	1.5	1.5	
consensusMCcov	50000	10	0.9	0.9	1.0	
		20	1.4	1.5	1.4	
		50	3.2	3.2	3.2	
	100000	10	13.1	3.6	13.3	
		20	2.4	2.4	7.2	
		100	6.8	6.4	6.8	
consensusMCindep	50000	10	0.4	0.4	0.4	
		20	0.3	0.4	0.3	
		50	0.4	0.4	0.4	
	100000	10	12.5	3.1	12.8	
		20	1.2	1.3	6.1	
		100	1.6	1.6	1.6	
Weierstrass	50000	10	3.0	3.1	4.5	
		20	6.0	6.1	8.9	
		50	17.5	14.8	21.9	
	100000	10	14.7	6.0	15.6	
		20	7.1	7.1	12.4	
		100	34.6	36.2	37.1	
PART	50000	10	0.4	0.3	0.5	
		20	0.5	0.5	0.3	
		50	0.4	0.5	0.5	
	100000	10	12.3	2.9	12.5	
		20	1.0	1.0	5.9	
		100	1.3	1.1	1.1	

Original: Single chain full-data posterior sample, sample average :Subposterior sample average, consensusMCcov: Consensus Monte Carlo Algorithm (for correlated parameters), consensusMCindep: Consensus Monte Carlo Algorithm (for independent parameters)
, parallel Weierstrass Sampling method, PART: parallel Random Partition Trees

4. Discussion

Since embarrassingly parallel MCMC is a natural fit for handling data with large number of observations, it is often used in scalable Bayesian computation for big data problems. Our numerical results demonstrate that the MC consensus embarrassingly parallel algorithms almost always yield the best performance in a Bayesian linear regression problem for low-dimensional data. Other methods all have strengths and limitations in this experiment. The PART method achieves fast running time with cost of accuracy and robustness. The Weierstrass method takes relatively longer time to finish iteration with reliable result.

This comparison results apply to other Bayesian problem since the same embarrassingly parallel schema is used to calculate subposteriors for each data subset. The only difference among these methods in experiments is that they use different approaches for combining subposteriors. The strength and weakness of these methods are likely remain unchanged in other scenarios as long as the sample size is constant. For instance, the PART method may still have high speed and low accuracy in logistical regression or other Bayesian problem. However, the assumption of normality may influence the performance of the methods since some methods highly rely on the assumption while other do not.

Although the non-parallel algorithms accelerates the updating process of MCMC, it is not as promising as the parallel methods since it fails to take advantage of the efficient distributed computing system. Single-thread methods still cannot handle data that are too large to store in the storage or memory in single machine.

Researchers often address communication intense method rather than implement the method in related papers. The cost of frequent network communication could significantly deteriorate the performance of the MCMC procedure. The real performances of communication intense method need further investigations.

In parallel with the existing parallel MCMC methods, other optimization approaches can also be adopted to scale up MCMC algorithms. One possible approach is combining some of the nonparallel technique with parallel technique in distributed settings. For instance, researcher can split the sample into data subsets and use firefly algorithm to obtain subposteriors from each subset simultaneously, and combine them by embarrassingly parallel method. This proposed approach could achieve further computation cost reduction, while the accuracy of the model may be compromised. The balance between accuracy and computation performance should always be considered before implementing these types of combined approaches.

References

[1] Big data. In Wikipedia. Retrieved March 5, 2016, from https://en.wikipedia.org/wiki/Big_data
[2] Maclaurin, D., & Adams, R. P. (2014). Firefly Monte Carlo: Exact MCMC with Subsets of Data. arXiv preprint arXiv:1403.5693.
[3] Neiswanger, W., Wang, C., & Xing, E. (2013). Asymptotically exact, embarrassingly parallel MCMC. arXiv preprint arXiv:1311.4780.
[4] Wang, X., Dunson, D. B. (2013). Parallelizing MCMC via Weierstrass sampler. arXiv preprint arXiv:1312.4605.
[5] Minsker, S., Srivastava, S., Lin, L., & Dunson, D. B. (2014). Robust and scalable Bayes via a median of subset posterior measures. arXiv preprint arXiv:1403.2660.
[6] Wang, X., Guo, F., Heller, K. A., & Dunson, D. B. (2015). Parallelizing MCMC with Random Partition Trees. arXiv preprint arXiv:1506.03164.
[7] Baker, J., Fearnhead, P., & Fox, E. (2015). Computational Statistics for Big Data.
[8] Martino, L., Elvira, V., Luengo, D., Corander, J., & Louzada, F. (2015). Orthogonal parallel MCMC methods for sampling and optimization. arXiv preprint arXiv:1507.08577.
[9] Angelino, E. L. (2014). Accelerating Markov chain Monte Carlo via parallel predictive prefetching.
[10] Quiroz, M., Villani, M., & Kohn, R. (2015). Scalable MCMC for Large Data Problems using Data Subsampling and the Difference Estimator. arXiv preprint arXiv:1507.02971.
[11] Srivastava, S., Cevher, V., Tran-Dinh, Q., & Dunson, D. B. (2015). WASP: Scalable Bayes via barycenters of subset posteriors. In Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics (pp. 912-920).
[12] Luengo, D., Martino, L., Elvira, V., & Bugallo, M. (2014). Efficient Combination of Partial Monte Carlo Estimators. viXra, Oct.
[13] Angelino, E., Kohler, E., Waterland, A., Seltzer, M., & Adams, R. P. (2014). Accelerating MCMC via parallel predictive prefetching. arXiv preprint arXiv:1403.7265.
[14] Banterle, M., Grazian, C., & Robert, C. P. (2014). Accelerating Metropolis-Hastings algorithms: delayed acceptance with prefetching. arXiv preprint arXiv:1406.2660.
[15] Scott, S. L., Blocker, A. W., Bonassi, F. V., Chipman, H., George, E., & McCulloch, R. (2013, October). Bayes and big data: The consensus Monte Carlo algorithm. In EFaBBayes 250 conference (Vol. 16).
[16] Nishihara, R., Murray, I., & Adams, R. P. (2014). Parallel MCMC with generalized elliptical slice sampling. The Journal of Machine Learning Research, 15(1), 2087-2112.
[17] Quiroz, M. (2015). Speeding up MCMC by delayed acceptance and data subsampling. arXiv preprint arXiv:1507.06110.
[18] Neal, R. M. (2010). MCMC using Hamiltonian dynamics. In Handbook of Markov Chain Monte Carlo (eds S. Brooks, A. Gelman, G. Jones and X.-L Meng). Boca Raton: Chapman and Hall-CRC Press.
[19] Welling, M. and Teh, Y. (2011). Bayesian learning via stochastic gradient Langevin dynamics, Proceedings of the 28th International Conference on Machine Learning (ICML), pp. 681- 688.
[20] Ahn, S., Korattikara, A. and Welling, M. (2012). Bayesian posterior sampling via stochastic gradient fisher scoring. Proceedings of the 29th International Conference on Machine Learning, pp. 1591-1598.
[21] Bardenet, R., Doucet, A., & Holmes, C. (2014). Towards scaling up Markov chain Monte Carlo: an adaptive subsampling approach. In International Conference on Machine Learning (ICML) (pp. 405-413).

[22] Angelino, E., Johnson, M. J., & Adams, R. P. (2016). Patterns of Scalable Bayesian Inference. arXiv preprint arXiv:1602.05221.