Chapter

A Review on the Exact Monte Carlo Simulation

Hongsheng Dai

Abstract

Perfect Monte Carlo sampling refers to sampling random realizations exactly from the target distributions (without any statistical error). Although many different methods have been developed and various applications have been implemented in the area of perfect Monte Carlo sampling, it is mostly referred by researchers to coupling from the past (CFTP) which can correct the statistical errors for the Monte Carlo samples generated by Markov chain Monte Carlo (MCMC) algorithms. This paper provides a brief review on the recent developments and applications in CFTP and other perfect Monte Carlo sampling methods.

Keywords: coupling from the past, diffusion, Monte Carlo, perfect sampling

1. Introduction

In the past 30 years, substantial progress has been made in popularizing Bayesian methods for the statistical analysis of complex data sets. An important driving force has been the availability of different types of Bayesian computational methods, such as Markov chain Monte Carlo (MCMC), sequential Monte Carlo (SMC), approximate Bayesian computation (ABC) and so on. For many practical examples, these computational methods can provide rapid and reliable approximations to the posterior distributions for unknown parameters.

The basic idea that lies behind these methods is to obtain Monte Carlo samples from the distribution of interest, in particular the posterior distribution. In Bayesian analysis of complex statistical models, the calculation of posterior normalizing constants and the evaluation of posterior estimates are typically infeasible either analytically or by numerical quadrature. Monte Carlo simulation provides an alternative. One of the most popular Bayesian computational methods is MCMC, which is based on the idea of constructing a Markov chain with the desired distribution as its stationary distribution.

By running a Markov chain, MCMC methods generate statistically dependent and approximate realizations from the limiting (target) distribution. A potential weakness of these methods is that the simulated trajectory of a Markov chain will depend on its initial state. A common practical recommendation is to ignore the early stages, the so-called *burn-in* phase, before collecting realizations of the state of the chain. How to choose the length of the *burn-in* phase is an active research area. Many methods have been proposed for *convergence diagnostics*; [10] gave a comparative review. Rigorous application of diagnostic methods requires either substantial empirical analysis of the Markov chain or complex mathematical analysis. In practice, judgments about convergence are often made by visual inspection of the realized chain or the application of simple rules of thumb.

Concerns about the *quality* of the sampled realizations of the simulated Markov chains have motivated the search for Monte Carlo methods that can be guaranteed to provide samples from the target distribution. This is usually referred to as *perfect sampling* or *exact sampling*. In some cases, for example, the multivariate normal, perfect samples are readily available. For more complicated distributions, perfect sampling can be achieved, in principle, by the *rejection method*. This involves sampling from a density that bounds a suitable multiple of the target density, followed by acceptance or rejection of the sampled value. The difficulty here is in finding a bounding density that is amenable to rapid sampling while at the same time providing sample values that are accepted with high probability. In general this is a challenging task, although efficient rejection sampling methods have been developed for the special class of log-concave densities; see, for example, [20, 22].

A surprising breakthrough in the search for perfect sampling methods was made by [37]. The method is known as *coupling from the past* (CFTP). This is a sophisticated MCMC-based algorithm that produces realizations exactly from the target distribution. CFTP transfers the difficulty of running the Markov chain for extensive periods (to ensure convergence) to the difficulty of establishing whether a potentially large number of coupled Markov chains have coalesced. An efficient CFTP algorithm depends on finding an appropriate Markov chain construction that will ensure fast coalescence. There have been a few further novel theoretical developments following the breakthrough of CFTP, including [20, 21, 38]. Since then, perfect sampling methods have attracted great attention in various Bayesian computational problems and applied probability areas.

Apart from coupling from the past, many other perfect sampling methods were proposed for specific problems, for example, perfect sampling for random spanning trees [2, 47] and path-space rejection sampler for diffusion processes [3–5]. Very recently, a type of divide-and-conquer method has been developed in [15, 16]. These methods use the technique for the exact simulation of diffusions and samples from simple sub-densities to obtain perfect samples from the target distribution.

Perfect samples are useful in Bayesian applications either as a complete replacement for MCMC-generated values or as a source of initial values that will guarantee that a conventional MCMC algorithm runs in equilibrium. Perfect samples can also be used as a means of quality control in judging a proposed MCMC implementation when there are questions about the speed of convergence of the MCMC algorithm or whether the chain is capable of exploring all parts of the sample space. Of course, when perfect samples can be obtained speedily, they will be preferred to MCMC values, since they eliminate such doubts. In addition, sophisticated perfect sampling methodology often motivates efficient approximate algorithms and computational techniques. For example, [43] uses regenerated Markov chains to obtain simple standard error estimates for importance sampling under MCMC context. The condition required there will allow us to carry out perfect sampling via multigamma coupling approach [23].

This paper will present a brief review for perfect Monte Carlo sampling and explain the advantages and drawbacks of different types of methods. Section 2 will present rejection sampling techniques, and then CFTP will be covered in Section 3. Recent divide-and-conquer methods will be reviewed in Section 4. The paper ends with a discussion in Section 5.

2. Rejection sampling techniques

Rejection sampling, also known as ‘acceptance-rejection sampling’, generates realizations from a target probability density function $f(x)$ by using a hat function

$Mg(x)$, where $f(x) \le Mg(x)$ and $g(x)$ are a probability density function from which samples can be readily simulated. The basic rejection sampling algorithm is as follows:

Algorithm 2.1 (Basic rejection sampling)

```
Sample $x$ from $g(x)$ and $U$ from Unif$[0, 1]$.                    01
If $U \le \frac{f(x)}{Mg(x)}$,                                        02
    accept $x$ as a realisation of $f(x)$ and stop;                  03
else                                                                 04
    reject the value of $x$ and go back to step 01.                  05
```

Many other perfect sampling methods are actually equivalent to rejection sampling. For example, ratio-of-uniform (RoU) method [45] may have to be implemented via a rejection sampling approach.

The efficiency of rejection sampling depends on the acceptance probability, which is $1/M$. To perform rejection sampling efficiently, it is very important to find hat functions which provides higher acceptance probabilities. In other words, we shall choose M as small as possible [39]; however for many complicated problems, there is no easy way to find M small enough to guarantee high acceptance probability. A number of sophisticated rejection sampling methods have been suggested for dealing with complex Bayesian posterior densities, which we discuss below.

2.1 Log-concave densities

A function $h(x)$ is called log-concave if

$$\log h(\lambda x + (1-\lambda)y) \ge \lambda \log h(x) + (1-\lambda)\log h(y),$$

for all x, y and $\lambda \in [0, 1]$. For the special class of log-concave densities, Gilks and wild [22] developed the adaptive rejection sampling (ARS) method. The method constructs an envelope function for the logarithm of the target density, $f(x)$, by using tangents to $\log f(x)$ at an increasing number, n, of points. The envelope function $u_n(x)$ is the piecewise linear *upper hull* formed from the tangents. Note that, the envelope can be easily constructed due to the concavity of $\log f(x)$. The method also constructs a squeeze function $l_n(x)$ which is formed from the chords of the tangent points. The sampling steps of the ARS algorithm are as follows.

Algorithm 2.2 (Adaptive rejection sampling).

```
Outputs a stream of perfect samples from $f(x)$.
Initialise $u_n(x)$ and $l_n(x)$ by using tangents at several points.                 01
Sample $x^*$ from density $\propto \exp(u_n(x))$ and $U \sim$ Unif$(0, 1)$              02
If $U \le \exp(l_n(x^*) - u_n(x^*))$, Output $x^*$;                                    03
else if $U \le f(x^*)/\exp(u_n(x^*))$,                                                 04
    Output $x^*$; Update $(u_n, l_n)$ to $(u_{n+1}, l_{n+1})$ using $x^*$;               05
Goto 02                                                                                06
```

The ARS algorithm is adaptive and the sampling density is modified whenever $f(x^*)$ is evaluated. In this way, the method becomes more efficient as the sampling continues. Leydold [29] extends ARS to log-concave multivariate densities. Leydold's algorithm is based on decomposing the domain of the density into cones and then computing tangent hyperplanes for these cones. Generic computer code for sampling from a multivariate log-concave density is available on the author's website; it is only necessary to code a subroutine for the target density. Leydold's algorithm works well for simple low-dimensional densities. The drawback of ARS algorithm is that it only works for log-concave densities, which is a very small class of posteriors in practice. Also computationally it is very challenging to implement ARS algorithm for high-dimensional densities [11].

2.2 Fill's rejection sampling algorithm

We consider a discrete Markov chain with transition probability $\mathbf{P}(x,y)$ and stationary distribution $\pi(x), x \in S$. Let $\tilde{\mathbf{P}}(x, y) = \pi(y)\mathbf{P}(y,x)/\pi(x)$ be the transition probability for the time-reversed chain. Suppose that there is a partial ordering on the states S, denoted by $x \preccurlyeq y$. Let $\hat{0}$ and $\hat{1}$ be unique extremal points of the partial ordering.

To demonstrate the algorithm given by [20], we will assume that there are update functions ϕ and $\tilde{\phi}$ both mapping $S \times [0,1]$ to S such that

$$\mathbf{P}(x,y) = P(\phi(x,U) = y), \tag{1}$$

$$\tilde{\mathbf{P}}(x,y) = P(\tilde{\phi}(x,U) = y), \tag{2}$$

where $U \sim \text{Unif}[0,1]$ and

$$x \preccurlyeq y \Rightarrow \tilde{\phi}(x,u) \preccurlyeq \tilde{\phi}(y,u) \quad a.e. \quad u \in [0,1].$$

The algorithm is as follows:

Algorithm 2.3 (Fill's algorithm)

```
1. Choose an integer $t > 0$. Fix $x_0 = \hat{0}$ and $y_0 = \hat{1}$.
2. Generate $x_k = \phi(x_{k-1}, U_k)$, $k = 1, \cdots, t$, where $\{U_k\}$ are i.i.d. Unif[0,1].
3. Generate $\tilde{U}_k$ from the conditional distribution of $U$ given $\tilde{\phi}(x_{t-k+1}, U) = x_{t-k}, k = 1, \cdots, t$.
4. Generate $y_k = \tilde{\phi}(y_{k-1}, \tilde{U}_k), k = 1, \cdots, t$.
5. If $y_t = x_0$ then accept $x_t$; else double $t$ and repeat from step 2.
```

In Algorithm 2.3 (step 2) $z := x_t$ is sampled from $\mathbf{P}^t(\hat{0}, \cdot)$. If we find an upper bound M for $\pi(z)/\mathbf{P}^t(\hat{0}, z)$, then we can use rejection sampling. Fill [20] finds a bounding constant given by $M = \pi(\hat{0})/\tilde{\mathbf{P}}^t(\hat{1}, \hat{0})$ and proves that steps from 3 to 5 of Algorithm 2.3 are to accept z with probability $\frac{\pi(z)}{M\mathbf{P}^t(\hat{0}, z)}$. The output of this algorithm is indeed a perfect sample from π.

From Algorithm 2.3, we can see that rejection sampling can still be possible, even if we do not have a closed form of the hat function. The first limitation of Algorithm 2.3 is that it works only if the time-reversed chain is monotone, but [21] has extended the algorithm theoretically for general chains. The second limitation is that step 3 of Algorithm 2.3 is difficult to perform [20]. For these reasons, Fill's algorithm is not practical for realistic problems.

3. Coupling from the past

Coupling from the past was introduced in the landmark paper of [37] which showed how to provide perfect samples from the limiting distribution of a Markov chain.

3.1 Basic CFTP algorithms

Let $\{X_t\}$ be an ergodic Markov chain with state space $\mathcal{X} = \{1, \cdots, n\}$, where the probability of going from i to j is p_{ij} and the stationary distribution is π. Suppose we design an updating function $\phi(\cdot, U)$, which satisfies $P[\phi(i, U) = j] = p_{ij}$, where ϕ is a deterministic function and U is a random variable. To simulate the next state Y of the Markov chain, currently in state i, we draw a random variable U and let $Y = \phi(i, U)$.

Let $f_t(i) = \phi(i, U_t)$, and define the composition

$$F_{t_1}^{t_2} = f_{t_2-1} \circ f_{t_2-2} \circ \cdots \circ f_{t_1+1} \circ f_{t_1}, \tag{3}$$

for $t_1 < t_2$.

Proposition 3.1 [37] Suppose there exists a time $t = -T$, the backward coupling time, such that chains starting from any state in $\mathcal{X} = \{1, \cdots, n\}$, at time $t = -T$, and with the same sequence $\{U_t, t = -T, \cdots, -1\}$, arrive at the same state X_0^*. Then it must follow that X_0^* comes from π.

This proposition is easy to prove. If we run an ergodic Markov chain from time $t = -\infty$ and with the sequence $\{U_t, t = -T, \cdots, -1\}$ after $-T$, the Markov chain will arrive at X_0^*. Then X_0^* comes exactly from π since it is collected at time 0 and the Markov chain started from $-\infty$. The algorithm is as follows:

Algorithm 3.1 (Basic CFTP)

```
t = 0                                   01
repeat                                  02
  $t = t - 1$                           03
  generate $U_t$                        04
  $F_t^0 = F_{t+1}^0 \circ \phi(\cdot, U_t)$   05
until $F_t^0(\cdot)$ is a constant      06
return $F_t^0(\cdot)$                   07
```

Propp and Wilson [37] also proved that this algorithm is certain to terminate. The idea of simulating from the past is important. Note that if we collect $F_0^T(\cdot)$ as the result, where T is the smallest value that makes $F_0^T(\cdot)$ a constant, we will get a biased sample. This is because T is a stopping time, which is not independent of the Markov chain.

3.2 Read-once CFTP

The basic CFTP algorithm needs to restart the Markov chains at some points in the past if they have not coalesced by time 0. We must use the same random sequence $\{U_t\}_{-\infty}^{-1}$ when we restart the Markov chains. In Monte Carlo simulations, we usually use *pseudo*random number generators, which are deterministic algorithms. So if we give the same random seed, we will get the same random sequence. This means that the same sequence $\{U_t\}$ can be regenerated in each coupling procedure.

If we can run the Markov chain forward starting at 0 and collect a perfect sample in the future, we will not have to regenerate $\{U_t\}$. Wilson [48] developed a read-once CFTP method to implement the forward coupling idea. A simple example is provided by [6]. In fact, the multigamma coupler in [23] can be implemented via the more efficient read-once CFTP algorithm.

3.3 Improvement of CFTP algorithms

Propp and Wilson [37] showed that the computational cost of the algorithm can be reduced if there is a partial order for the state space $\mathcal{X}$ that is preserved by the update function ϕ, i.e. if $x \leq y$ then $\phi(x, U) \leq \phi(y, U)$. Their procedure is outlined in Algorithm 3.2, whereas before $\hat{0}$ and $\hat{1}$ are the unique extremals. Note that their algorithm needs a monotone update function ϕ for the Markov chain, while Algorithm 2.3 requires a monotone update function $\tilde{\phi}$ for the time-reversed chain.

Algorithm 3.2 (Monotone CFTP)

```
$T = 1$                                   01
Repeat                                    02
    upper = $\hat{1}$                      03
    lower = $\hat{0}$                      04
    for $t = -T$ to $t = -1$               05
        upper = $\phi_t(\text{upper}, U_t)$   06
        lower = $\phi_t(\text{lower}, U_t)$   07
    $T = 2T$                               08
until upper = lower                       09
return upper                              10
```

Algorithm 3.2 is much simpler than Algorithm 3.1, since only two chains have to be run at the same time, but the requirement of monotonicity is very restrictive. Markov chains with transitions given by independent Metropolis-Hastings and perfect slice sampling have been shown to have this property, by [9, 32], respectively. However [32, 34] have also noticed that such independent M-H CFTP is equivalent to simple rejection sampler.

In general it is hard to code perfect slice samplers correctly. For example, Hörmann and Leydold [26] have pointed out that the perfect slice samplers in [7, 36] are incorrect. The challenge of monotone CFTP is usually to construct the detailed updating function with a guarantee of preserving the partial order.

Finding a partial order preserved by the Markov chains is a non-trivial task in many cases. An alternative improvement is to use CFTP with bounding chains, such as that in [27, 33]. If the bounding chains, which bound all the Markov chains, coalesce, then all Markov chains coalesce. Thus if only a few bounding chains are required, the efficiency of the CFTP algorithm can be improved significantly. Sometimes, it may be impossible to define an explicit bounding chain (the maximum of the state space may be infinity, and the upper bound chain cannot start from infinity), but it is possible to use a dominated process to bound all Markov chains [28].

3.4 Applications and challenges

Although CFTP is extremely challenging to be implemented for many practical problems, it did find a few applications in certain discrete state space problems, for example, the Ising model [37]. Also [18] applied CFTP to ancestral selection graph to simulate samples from population genetic models. Refs. [12, 14] applied CFTP to a class of fork-join type queuing system problems. Connor and Kendal [8] applied CFTP for the perfect simulation of M/G/c queues. CFTP also finds its application in signal processing [17].

CFTP for continuous state space Markov chains is very challenging, since a random map from an interval to a finite number of points is required. In recent years, many methods have been developed for unbounded continuous state space Markov chains, such as perfect slice sampler in [32], multigamma coupler and the bounded M-H coupler in [23, 34]. Wilson [49] developed a layered multi-shift coupling, which shifts states in an interval to a finite number of points. However, none of these methods can solve any practical problems.

4. Recent advances in perfect sampling

Recently, a new type of perfect Monte Carlo sampling method based on the decomposition of the target density f, as $f(\cdot) = g_1(\cdot)g_2(\cdot)$, was proposed in [15], where g_1 and g_2 are also (proportional to) proper density functions. Note that here g_1 and g_2 are continuous density functions which are easy to simulate from. Suppose that q-dimensional vector values $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are independently drawn from g_1 and g_2, respectively. If the two independent samples are equal, i.e. $\boldsymbol{x}_1 = \boldsymbol{x}_2 = \boldsymbol{y}$ then we have $\boldsymbol{y}$ must be from $f(\cdot) \propto g_1(\cdot)g_2(\cdot)$. Note that such a naive approach may be practical for discrete random variables with low-dimensional state space, but for continuous random variables, it is impossible since $\mathrm{P}(\boldsymbol{x}_1 = \boldsymbol{x}_2) = 0$. Dai [15] proposed a novel approach to deal with this, which is explained in the following subsection.

4.1 Perfect distributed Monte Carlo without using hat functions

First we introduce the following notations related to the logarithm of the sub-densities:

$$\boldsymbol{\alpha}(\boldsymbol{x}) = \left(\alpha^{(1)}, \cdots, \alpha^{(q)}\right)^{tr}(\boldsymbol{x}) = \nabla \log g_1(\boldsymbol{x}) \tag{4}$$

where ∇ is the partial derivative operator for each component of $\boldsymbol{x}$. Then we consider a q-dimensional diffusion process $\boldsymbol{X}_t\left(\vec{\boldsymbol{\omega}}\right), t \in [0, T]$ $(T < \infty)$, defined on the q-dimensional continuous function space $\boldsymbol{\Omega}$, given by:

$$d\boldsymbol{X}_t = \boldsymbol{\alpha}(\boldsymbol{X}_t)dt + d\boldsymbol{B}_t, \tag{5}$$

where $\boldsymbol{B}_t\left(\vec{\boldsymbol{\omega}}\right) = \boldsymbol{\omega}_t$ is a Brownian motion and $\vec{\boldsymbol{\omega}} = \{\boldsymbol{\omega}_t, t \in [0, T]\}$ is a typical element of $\boldsymbol{\Omega}$. Let $\mathbb{W}$ be the probability measure for a Brownian motion with the initial probability distribution $\boldsymbol{B}_0 = \boldsymbol{w}_0 \sim f_1(\cdot) = g_1^2(\cdot)$.

Clearly $\boldsymbol{X}_t$ has the invariant distribution $f_1(\boldsymbol{x})$ (using the Langevin diffusion results [24]). Let $\mathbb{Q}$ be the probability law induced by $\boldsymbol{X}_t, t \in [0, T]$, with $\boldsymbol{X}_0 = \boldsymbol{\omega}_0 \sim f_1(\cdot)$, i.e. under $\mathbb{Q}$ we have $\boldsymbol{X}_t \sim f_1(\boldsymbol{x})$ for any $t \in [0, T]$.

The idea in [15] is to use a *biased* diffusion process $\overline{\mathbf{X}} = \left\{\overline{\boldsymbol{X}}_t; 0 \leq t \leq T\right\}$ to simulate from the target function f. It is defined as follows.

Definition 4.1 (Biased Langevin diffusions) The joint density for the pair $\left(\overline{\boldsymbol{X}}_0, \overline{\boldsymbol{X}}_T\right)$ (the starting and ending points of the biased diffusion process), evaluated at point $(\boldsymbol{x}, \boldsymbol{y})$, is $f_1(\boldsymbol{x})\mathbf{t}^*(\boldsymbol{y}|\boldsymbol{x})f_2(\boldsymbol{y})$, where $\mathbf{t}^*(\boldsymbol{y}|\boldsymbol{x})$ is the transition density for the diffusion process X defined in Eq. (6) from $\boldsymbol{X}_0 = \boldsymbol{x}$ to $\boldsymbol{X}_T = \boldsymbol{y}$ and $f_2(\boldsymbol{y}) = g_2(\boldsymbol{y})/g_1(\boldsymbol{y})$.

Given $\left(\overline{\boldsymbol{X}}_0, \overline{\boldsymbol{X}}_T\right)$ the process $\left\{\overline{\boldsymbol{X}}_t, 0 < t < T\right\}$ is given by the diffusion bridge driven by Eq. (6).

The marginal distribution for $\overline{\boldsymbol{X}}_T$ is $f(\boldsymbol{y})$. Therefore, to draw a sample from the target distribution $f(\boldsymbol{x})$, we need to simulate a process $\overline{\boldsymbol{X}}_t, t \in [0, T]$ from $\overline{\mathbb{Q}}$ (the law induced by $\overline{\boldsymbol{X}}$) and then $\overline{X}_T \sim f(\boldsymbol{x})$.

Simulation from $\overline{\mathbb{Q}}$ can be done via rejection sampling. We can use a *biased Brownian motion* $\left\{\overline{\boldsymbol{B}}_t; 0 \le t \le T\right\}$ as the proposal diffusion:

Definition 4.2 (Biased Brownian motion) The starting and ending points $\left(\overline{\boldsymbol{B}}_0, \overline{\boldsymbol{B}}_T\right)$ follow a distribution with a density $h(\boldsymbol{x}, \boldsymbol{y})$, and $\left\{\overline{\boldsymbol{B}}_t; 0 < t < T\right\}$ is a Brownian bridge given $\left(\overline{\boldsymbol{B}}_0, \overline{\boldsymbol{B}}_T\right)$.

Under certain mild conditions, Dai [15] proved the following lemma.

Lemma 4.1 Let $\mathbb{Z}$ be the probability law induced by $\left\{\overline{\boldsymbol{B}}_t; 0 \le t \le T\right\}$. By letting

$$h(\boldsymbol{\omega}_0, \boldsymbol{\omega}_T) = g_2(\boldsymbol{\omega}_T)g_1(\boldsymbol{\omega}_0)\frac{1}{\sqrt{2\pi T}}e^{-\frac{\|\boldsymbol{\omega}_T - \boldsymbol{\omega}_0\|^2}{2T}} \tag{6}$$

we have

$$\frac{d\overline{\mathbb{Q}}}{d\mathbb{Z}}\left(\vec{\boldsymbol{\omega}}\right) \propto \exp\left[-\frac{1}{2}\int_0^T \left(\|\boldsymbol{\alpha}\|^2 + \operatorname{div}\boldsymbol{\alpha}\right)(\boldsymbol{\omega}_t)dt\right] \tag{7}$$

where div is the divergence operator.

Condition 4.1 There exists $l > -\infty$ such that

$$\frac{1}{2}\left(\|\boldsymbol{\alpha}\|^2 + \mathbf{div}\,\boldsymbol{\alpha}\right)(\boldsymbol{x}) - l \ge 0. \tag{8}$$

Under Condition 4.1 the ratio (8) can be rewritten as

$$\frac{d\overline{\mathbb{Q}}}{d\mathbb{Z}}\left(\vec{\boldsymbol{\omega}}\right) \propto \exp\left[-\int_0^T \left(\frac{1}{2}\left(\|\boldsymbol{\alpha}\|^2 + \mathbf{div}\,\boldsymbol{\alpha}\right)(\boldsymbol{\omega}_t) - l\right)dt\right], \tag{9}$$

which has a value no more than 1.

Therefore we can use rejection sampling to simulate from $\overline{\mathbb{Q}}$, with proposal measure $\mathbb{Z}$. This acceptance probability (10) can be dealt with using similar methods ~~as that in [3, 5]. The algorithm is presented below; see [13, 15] for more details.~~

Algorithm 4.1 (Simple distributed sampler)

```
Simulate $(\omega_0, \omega_T)$ from density $h$                                              01
Simulate the biased Brownian bridge $(\overline{B}_t, t \in (0, T))$                         02
Accept $\omega_T$ as a sample from $f$, with probability (6); If rejected, go back to step 01.  03
```

Such a method is a rejection sampling algorithm, but it does not require finding a hat function to bound the target density, which is usually the main challenge of the traditional rejection sampling for complicated target densities. The above algorithm uses g_2 as the proposal density function, which does not have to bound the target f. However, it requires a bound for the derivatives of the logarithm of the sub-densities (see Condition 4.1). This is usually easier to get in practice, since the logarithm of the posterior is usually easy to deal with. Also [15] noted that we should choose sub-densities g_1 and g_2 as similar as possible, in order to achieve high acceptance probability.

Dai [15] focused on the simple decomposition of $f = g_1 g_2$, although it mentioned that for more general decomposition of $f = g_1 g_2 \cdots g_C$, a recursive method can be used. Unfortunately, a naive recursive method is very inefficient. A more sophisticated method is introduced in the following section.

4.2 Monte Carlo fusion for distributed analysis

A more efficient and sophisticated methods were proposed recently in [16], named as Monte Carlo fusion. Suppose that we consider

$$f(\boldsymbol{x}) \propto g_1(\boldsymbol{x}) \cdots g_C(\boldsymbol{x}), \tag{10}$$

where each $g_c(\boldsymbol{x})$ ($c \in \{1, \dots, C\}$) is a density (up to a multiplicative constant). Here C denotes the number of parallel computing cores available in big data problems, and each $g_c(\boldsymbol{x})$ means the sub-posterior density based on a subset of the big data. In group decision problems, C means the number of different decisions which should be combined and $g_c(\boldsymbol{x})$ stands for the decision from each group member.

Dai et al. [16] considered simulating from the following density on extended space,

$$g\left(\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(C)}, \boldsymbol{y}\right) = \prod_{c=1}^{C} \left[g_c^2\left(\boldsymbol{x}^{(c)}\right) \cdot p_c\left(\boldsymbol{y} \mid \boldsymbol{x}^{(c)}\right) \cdot \frac{1}{g_c(\boldsymbol{y})} \right], \tag{11}$$

which admits the marginal density f for $\boldsymbol{y}$. Here $p_c\left(\boldsymbol{y} \mid \boldsymbol{x}^{(c)}\right)$ is the transition density from $\boldsymbol{x}^{(c)}$ to $\boldsymbol{y}$ for the Langevin diffusion defined in Eq. (6) associated with each sub-density g_c.

Dai et al. [16] considered a rejection sampling approach with proposal density proportional to the function

$$h\left(\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(C)}, \boldsymbol{y}\right) = \prod_{c=1}^{C} \left[g_c\left(\boldsymbol{x}^{(c)}\right) \right] \cdot \exp\left(-\frac{C \cdot \|\boldsymbol{y} - \overline{\boldsymbol{x}}\|^2}{2T} \right), \tag{12}$$

where $\overline{\boldsymbol{x}} = C^{-1} \sum_{c=1}^{C} \boldsymbol{x}^{(c)}$ and T is an arbitrary positive constant.

Simulation from the proposal h can be achieved directly. In particular, $\boldsymbol{x}^{(1)}, \dots \boldsymbol{x}^{(C)}$ are first drawn independently from $g_1, \dots, g_C$, respectively, and then $\boldsymbol{y}$ is simply a Gaussian random variable centred on $\overline{\boldsymbol{x}}$. This is a distributed analysis or divide-and-conquer approach. Detailed acceptance probabilities and rejection sampling algorithms can be found in [16].

The above fusion approach arises in modern statistical methodologies for ‘big data’. A full dataset will be artificially split into a large number of smaller data sets, and inference is then conducted on each smaller data set and combined (see, for instance, [1, 30, 31, 35, 40–42, 46]). The advantage for such an approach is that inference on each small data set can be conducted in parallel. Then the heavy computational cost of algorithms such as MCMC will not be a concern. Traditional methods suffer from the weakness that the fusion of the separately conducted inferences is inexact. However, the Monte Carlo fusion in [16] is an exact simulation algorithm and does not have any approximation weakness.

The above fusion approach also arises in a number of other settings, where distributed analysis came naturally. For example, in signal processing, distributed multi-sensor may be used for network fusion systems. Fusion approach arises naturally to combine results from different sensors [44].

5. Conclusion

Although perfect simulation usually refers to correcting the statistical errors for the samples drawn via MCMC, it actually covers a much wider area beyond CFTP. In fact for certain applications, it is often possible to construct other types of perfect sampling methods which are much more efficient than CFTP. For example, for the exact simulation of the posterior of simple mixture models, the geometric-arithmetic mean (GAM) method in [11] is much more efficient than CFTP in [25]. Details of GAM method is provided in Appendix. Also the random walk construction for exact simulation for random spanning trees [2] is much more efficient than the CFTP version.

Bayesian computational algorithms keep evolving, in particular under the current big data era. Although almost all newly developed algorithms are approximate simulation algorithms, perfect sampling is still one of the key wheel-driven forces for new Bayesian computational algorithms, and they usually can quickly motivate new class of 'mainstream' algorithms. More focus should be given to methods beyond CFTP, for example, the fusion type of algorithms.

The Monte Carlo fusion method has the potential to be used in many Bayesian big data applications. For example, for large car accident data, the response variable is usually a categorical variable representing the seriousness of the accident, and generalized linear regression model is often used. Under a Bayesian framework, we may estimate the posterior distribution for the regression parameters via such a fusion approach. Then the posterior mean, the posterior median, or other characteristics of the posterior distribution can be estimated using the Monte Carlo samples. Also such an algorithm is perfect sampling algorithm, and no convergence justification is needed, since it always provided realizations exactly from the target distribution.

Appendix

A. Geometric-arithmetic mean method for simple mixture model

Observations from a simple mixture model are assumed to be either discrete or continuous. The density function of an individual observation y has the form

$$f(y;\boldsymbol{p}) = \sum_{k=1}^{K} p_k f_k(y), \quad \text{where} \quad \sum_{k=1}^{K} p_k = 1, \quad \text{and} \quad p_k > 0, k = 1, \dots, K. \qquad (13)$$

We assume that the component weights $\boldsymbol{p} = (p_1, \dots, p_K)$ are unknown parameters and the number of components, K, and the component densities, f_k, are all known. We focus on the perfect sampling from the posterior distribution of $\mathbf{p}$.

Suppose that we have N observations, $y_1, \ldots, y_N$. Let $L_{nk} = f_k(y_n)$ and assume that the prior distribution of $\mathbf{p}$ is Dirichlet:

$$\pi_0(\mathbf{p}) \propto \prod_{k=1}^{K} p_k^{\alpha_k - 1}, \quad \alpha_k > 0, k = 1, \ldots, K. \tag{14}$$

Then the posterior distribution is given by

$$f(\mathbf{p}|y) \propto \pi_0(\mathbf{p}) \prod_{n=1}^{N} \left(\sum_{k=1}^{K} p_k L_{nk} \right) I_{\mathcal{X}}(\mathbf{p}), \tag{15}$$

where $\mathcal{X} = \left\{ \mathbf{p} | \sum_{k=1}^{K} p_k = 1, p_k > 0, k = 1, \ldots, K \right\}$.

There are several ways to carry out perfect sampling from Eq. (16). The first method is based on CFTP [25]. An alternative perfect sampling method for simple mixture models is introduced by [19]. The third approach is to use adaptive rejection sampling [22, 29], since the posterior is log-concave. We may also use the ratio-of-uniform method. However, none of these methods are more efficient than the geometric–arithmetic mean method in [11].

A.1 Geometric-arithmetic mean method

Suppose that $\mathbf{p}^*$, the MLE of $\mathbf{p}$ is unique and for simplicity, assume the prior $\pi_0(\mathbf{p})$ is uniform. Define $a_{nk} = L_{nk} / \sum p_k^* L_{nk}$. The posterior density of $\mathbf{p}$ is then given by

$$f(\mathbf{p}|y) \propto h(\mathbf{p}|y) = \prod_{n=1}^{N} \left(\sum_{k=1}^{K} p_k a_{nk} \right) I_{\mathcal{X}}(\mathbf{p}), \tag{16}$$

where $\mathcal{X}$ is defined in (16).

Let I_n be a random element of $\arg\max_k L_{nk}$. Define $A_j = \{n : I_n = j\}$ and let $\mathbf{n} = (n_1, \ldots, n_k)$ where n_j is the number of elements in A_j.

Define $\mathbf{M} = \{m_{jk}\}$ with $m_{jk} = \left(\sum_{n \in A_j} a_{nk} \right) / n_j$. If $n_j = 0$, then set $m_{jj} = 1$ and $m_{jk} = 0$ for $j \neq k$. We now make two assumptions, which we will return to later on:

A: $\mathbf{M}$ is invertible.

B: The elements of $\mathbf{v} = \left(\mathbf{M}^T\right)^{-1}\mathbf{1}$ are all positive.

Under these assumptions, we will show that the following rejection sampler generates simulated values from the posterior distribution of $\mathbf{p}$. First we define $\mathbf{V}$ to be the diagonal matrix with diagonal elements $\mathbf{v}^T = (v_1 \ldots, v_K)$.

Algorithm 6.1 (GAM sampler)

```
Sample $\mathbf{q}$ from the Dirichlet distribution with parameter $\mathbf{n} + \mathbf{1}$.   01
Sample $U$ from Unif$[0, 1]$.                                                                  02
Calculate $\mathbf{p}$ with $\mathbf{p} = \mathbf{M}^{-1} V^{-1} \mathbf{q}$.                  03
If $U \le h(\mathbf{p}|y) / \prod_{j=1}^{K} \left(q_j / v_j\right)^{n_j}$,                      04
    Accept $\mathbf{p}$ and stop;                                                              05
else                                                                                           06
    reject $\mathbf{p}$ and go to 01.                                                          07
```

Proposition 6.1 Under assumptions **A** and **B**, Algorithm 6.1 samples $\mathbf{p}$ with probability density (17).

Proof: Since the geometric average is no larger than the arithmetic average, for $\mathbf{p} \in \mathcal{X}$, we have

$$h(\mathbf{p}|y) = \prod_{n=1}^{N} \left(\sum_{k=1}^{K} p_k a_{nk} \right) = \prod_{j=1}^{K} \prod_{n \in A_j} \left(\sum_{k=1}^{K} p_k a_{nk} \right) \tag{17}$$

$$\leq \prod_{j=1}^{K} \left(\frac{\sum_{n \in A_j} \left(\sum_{k=1}^{K} p_k a_{nk} \right)}{n_j} \right)^{n_j}, \tag{18}$$

where in the case $n_j = 0$, the product term is taken as 1. So that, for $\mathbf{p} \in \mathcal{X}$, with m_{jk} as previously defined, we have

$$h(\mathbf{p}|y) \leq \prod_{j=1}^{K} \left(\sum_{k=1}^{K} p_k m_{jk} \right)^{n_j} \tag{19}$$

$$= \left[\prod_{j=1}^{K} v_j^{-n_j} \right] \prod_{j=1}^{K} \left(v_j \sum_{k=1}^{K} p_k m_{jk} \right)^{n_j} \tag{20}$$

$$= \prod_{j=1}^{K} \left(q_j / v_j \right)^{n_j}, \tag{21}$$

where $q_j = v_j \sum_{k=1}^{K} p_k m_{jk}, j = 1, \dots, K$ or equivalently $\mathbf{q} = \mathbf{VMp}$.

Since $v_j > 0$ and $\sum_{k=1}^{K} p_k m_{jk} > 0$, it follows that $q_j > 0$ for $j = 1, \dots, K$. Furthermore

$$\sum_{j=1}^{K} q_j = \sum_{j=1}^{K} v_j \sum_{k=1}^{K} p_k m_{jk} = \mathbf{p}^T \mathbf{M}^T \mathbf{v} = \mathbf{p}^T \mathbf{1} = 1,$$

since $\mathbf{M}^T \mathbf{v} = \mathbf{1}$, from the definition of $\mathbf{v}$. It follows that $\mathbf{p} \in \mathcal{X}$ implies $\mathbf{q} \in \mathcal{X}$, so that

$$h(\mathbf{p}|y) I_{\mathcal{X}}(\mathbf{p}) \leq \prod_{j=1}^{K} \left(q_j / v_j \right)^{n_j} I_{\mathcal{X}}(\mathbf{q}). \tag{22}$$

Note that the right-hand side of Eq. (22) is proportional to a Dirichlet distribution with parameters $(n_1 + 1, \dots, n_K + 1)$.

Rejection sampling then proceeds as usual:

- A sample $\mathbf{q}$ is drawn from Dirichlet$(\mathbf{n} + \mathbf{1})$.
- The value $\mathbf{p} = \mathbf{M}^{-1} \mathbf{V}^{-1} \mathbf{q}$ is calculated.
- It is accepted with probability $h(\mathbf{p}|y) I_{\mathcal{X}}(\mathbf{p}) / \prod_{j=1}^{K} \left(q_j / v_j \right)^{n_j}$.

We now return to assumptions **A** and **B**. Suppose that $\mathbf{M}$ is invertible but the elements of $\mathbf{v} = \left(\mathbf{M}^T \right)^{-1} \mathbf{1}$ are not all positive. In this case, let

$\alpha_k = \frac{1}{N} \sum_{n=1}^{N} a_{nk}, \quad \alpha = \max_k \{\alpha_k\}, \quad \mathbf{v} = (\alpha N)^{-1} \mathbf{n} \quad$ and $\quad \tilde{\mathbf{M}} = \alpha \mathbf{M} \Delta,$

where Δ is a diagonal matrix with diagonal elements $(1/\alpha_1, \dots, 1/\alpha_K)$. Note that $\mathbf{v} = \left(\tilde{\mathbf{M}}^T\right)^{-1}\mathbf{1} > 0$. Algorithm 6.1 and its proof can then be modified by replacing $\mathbf{M}$ by $\tilde{\mathbf{M}}$.

Suppose now that assumption **A** does not hold, i.e. $\mathbf{M}$ is not invertible. This can be remedied by adding positive quantities to the diagonal elements of $\mathbf{M}$. This also provides an alternative way of ensuring that the elements of $\mathbf{v}$ are positive.

A.2 Dirichlet priors and *pseudo* data

Suppose that the prior $\pi_0(\mathbf{p})$ is Dirichlet$(\alpha_1+1, \dots, \alpha_K+1)$, where $\alpha_i : i = 1, \dots, K$ are positive, integers and let $A = \sum_{j=1}^{K}\alpha_j$. The prior can be synthesized by introducing *pseudo* data, $\tilde{a}_{mk}, m = 1, \dots, A; k = 1, \dots, K$, defined as follows:

$$\tilde{a}_{mk} = \begin{cases} 1 & \text{if} \sum_{j=1}^{k-1}\alpha_j + 1 \le m \le \sum_{j=1}^{k}\alpha_j \\ 0 & \text{otherwise,} \end{cases} \tag{23}$$

since

$$\pi_0(\mathbf{p}) \propto \prod_{k=1}^{K} p_k^{\alpha_k} = \prod_{m=1}^{A}\left(\sum_{k=1}^{K}\tilde{a}_{mk}p_k\right). \tag{24}$$

With the Dirichlet prior, the posterior distribution given by Eq. (16) can be written as

$$f(\mathbf{p}|y) \propto \prod_{l=1}^{N+A}\left(\sum_{k=1}^{K}p_k\bar{a}_{lk}\right)I_{\mathcal{X}}(\mathbf{p}), \tag{25}$$

where $\{\bar{a}_{lk}, l = 1, \dots, N+A\}$ contains the real data $\{a_{nk}, n = 1, \dots, N\}$ and the pseudo data $\{\tilde{a}_{mk}, m = 1, \dots, A\}$.

The posterior distribution (26) has the same form as Eq. (17). Therefore GAM can be used to sample realizations from the posterior distribution (26).

A.3 Simulation results and discussion

We compare the running time of mixture models with sample sizes (N) and different number of components (K) in **Table 1**. The components have specified normal distributions with means $\mu = (\mu_1, \dots, \mu_K)$ and variances $\sigma^2 = \left(\sigma_1^2, \dots, \sigma_K^2\right)$. The prior on $\mathbf{p}$ is uniform. We sample 10,000 realizations from the posterior of the models.

When $K = 3, 4$, we simulate N observations from a three-component normal mixture with $\mu = (0, 0, 2)$, $\sigma^2 = (1, 4, 1)$ and mixture weight $\mathbf{p}_0 = (1/2, 1/3, 1/6)$. We then either sample from the posterior distribution of $\mathbf{p}$ using the same distributional components in the case $K = 3$ or sample from the posterior distribution of $\mathbf{p}$ with an additional component having mean $\mu_4 = 4$ and variance $\sigma_4^2 = 4$, in the case $K = 4$.

When $K = 5$, observations are simulated from the normal mixture distribution with components having means $\mu = (-2, 0, 4, 2, 3)$, variances $\sigma^2 = (1, 1, 4, 1, 4)$ and $\mathbf{p}_0 = (0.35, 0.3, 0.1, 0.2, 0.05)$. Samples from the posterior distribution of $\mathbf{p}$

are drawn assuming the same components. Similar calculations are carried out for $K = 6$, where $\mu = (0, 3, 2, -2, -4, 5)$, variances $\sigma^2 = (1, 1, 1, 1, 1, 4)$ and $\mathbf{p}_0 = (0.05, 0.3, 0.3, 0.1, 0.08, 0.17)$, again assuming that the component distributions are known.

From **Table 1**, we can see that the GAM algorithm, while using very little memory, is highly efficient in running time. The last row of the table is the estimated acceptance probability of the GAM algorithm. The algorithm is very efficient when the component densities are known. We can see this not only by simulation but also from theoretical considerations, as follows.

A.3.1 Explanation of efficiency

When $\mathbf{v} = \left(\mathbf{M}^T\right)^{-1}\mathbf{1} > 0$, we are able to use $\mathbf{M}$ directly without modification to construct the hat function, thereby speeding up the calculations. In the simulations of the previous section, this was always found to be the case. Now we explain why this should be so.

If the maximum likelihood estimator of $\mathbf{p}$ is consistent, then when $N \to \infty$,

$$\frac{1}{n_j}\sum_{n \in A_j}\left|\frac{L_{nk}}{\sum_{k=1}^K p_k^* L_{nk}} - \frac{L_{nk}}{\sum_{k=1}^K p_k L_{nk}}\right| \xrightarrow{p} 0. \tag{26}$$

Assuming sufficient regularity, we also have

$$m_{jk} = \frac{\sum_{n \in A_j} a_{nk}}{n_j} \tag{27}$$

$$= \frac{1}{n_j}\sum_{n \in A_j}\frac{L_{nk}}{\sum_{k=1}^K p_k^* L_{nk}} \tag{28}$$

$$\xrightarrow{p} E\left(\frac{f_k(Y)}{f(Y)}|\mathcal{L}_j\right) \tag{29}$$

$$= \frac{\int_{\mathcal{L}_j} f_k(y)dy}{\gamma_j}, \qquad \text{as} \quad N \to \infty, \tag{30}$$

where Y has density $f(y) = \sum_{k=1}^K p_k f_k(y)$, $\mathcal{L}_j = \left\{y | f_j(y) \geq f_k(y), k = 1, \dots, K\right\}$ and $\gamma_j = \int_{\mathcal{L}_j} f(y)dy$.

K	3	3	4	4	6	6
N	400	1000	400	1000	400	1000
Fearnhead's	242 s	3610 s	*	*	*	*
Leydold's	≤1 s	3.6 s	*	*	*	*
RoU	16:11 s	28:16 s	31:18 s	68:33 s	88:60 s	152:76 s
GAM	4 s	9 s	11 s	16 s	6 s	11 s
GAM AP	0.7472	0.7509	0.2433	0.3088	0.5325	0.5505

*Fearnhead's algorithm, Leydold's algorithm and ratio-of-uniform. GAM method. GAM acceptance probability. The * indicates that Fearnhead's method, and Leydold's method will not run on a standard desktop when $K = 4$ and $K = 5$.*

Table 1.
Running times (in s).

Therefore

$$\mathbf{M}^T \xrightarrow{p} \mathbf{W}^T, \tag{31}$$

where

$$\mathbf{W} = \{w_{jk}\}, w_{jk} = \frac{\int_{\mathcal{L}_j} f_k(y)dy}{\gamma_j}. \tag{32}$$

Let $\overline{\mathbf{v}} = (\gamma_1, \dots, \gamma_K)$, then

$$\mathbf{W}^T\overline{\mathbf{v}} = \begin{bmatrix} \sum_{j=1}^K \int_{\mathcal{L}_j} f_1(y)dy \\ \vdots \\ \sum_{j=1}^K \int_{\mathcal{L}_j} f_K(y)dy \end{bmatrix} = \begin{bmatrix} \int f_1(y)dy \\ \vdots \\ \int f_K(y)dy \end{bmatrix} = \mathbf{1}, \tag{33}$$

where the second equal sign is because $\cup_{j=1}^K \mathcal{L}_j = (-\infty, \infty)$. So, there exists $\overline{\mathbf{v}} > 0$, satisfying $\mathbf{W}^T\overline{\mathbf{v}} = \mathbf{1}$. Using Eq. (32), we can conclude that when N is large enough, there also exists $\mathbf{v} \approx \overline{\mathbf{v}} > 0$, satisfying $\mathbf{M}^T\mathbf{v} = \mathbf{1}$.

Since $\gamma_j = \int_{\mathcal{L}_j} f(y)dy$ and $n_j = \#\{A_j\}$, we have $n_j/N \xrightarrow{p} \gamma_j$. When each $n_j, j = 1, \dots, K$ is large, if a random sample $\mathbf{q}$ is drawn from a Dirichlet distribution with parameter $\mathbf{n} + \mathbf{1}$, then each q_j is approximately equal to $n_j/N \approx \gamma_j$. Furthermore, $v_j \approx \gamma_j$, so $\mathbf{q}$ satisfies

$$\mathbf{V}^{-1}\mathbf{q} \approx \mathbf{1}, \tag{34}$$

and then,

$$\mathbf{p} = \mathbf{M}^{-1}\mathbf{V}^{-1}\mathbf{q} \approx \mathbf{M}^{-1}\mathbf{1} = \mathbf{p}^*. \tag{35}$$

If $\mathbf{p}$ is approximately equal to the mode $\mathbf{p}^*$, the two sides of the inequality,

$$h(\mathbf{p}|y) = \prod_{n=1}^N \left(\sum_{k=1}^K p_k a_{nk} \right) \le \left[\prod_{j=1}^K v_j^{-n_j} \right] \prod_{j=1}^K q_j^{n_j}, \tag{36}$$

are approximately equal as well. Thus, the closer the sampled realization $\mathbf{p}$ is to $\mathbf{p}^*$, the larger the acceptance probability is. So the algorithm runs very rapidly, since the sampled values of $\mathbf{p}$ are always around the mode $\mathbf{p}^*$.

This algorithm requires calculating the MLE, which can be performed very quickly since the likelihood function is log-concave. In fact an approximate *guess* for $\mathbf{p}^*$ will suffice. The more accurate the guess is, the more efficient the algorithm will be.

The method performs well when the component densities are correctly specified, as explained in the previous section. For these same reasons, we would expect the algorithm to perform poorly under misspecification. Details of robustness to misspecification can be found in [11].

Author details

Hongsheng Dai
University of Essex, Colchester, UK

*Address all correspondence to: hdaia@essex.ac.uk

References

[1] Agarwal A, Duchi JC. Distributed delayed stochastic optimization. In: 51st IEEE Conference on Decision and Control; Maui Hawaii, USA. 2012

[2] Aldous DJ. The random walk construction of uniform spanning trees and uniform labelled trees. SIAM Journal on Discrete Mathematics. 1990;**3**:450-465

[3] Beskos A, Papaspiliopoulos O, Roberts GO, Fearnhead P. Exact and computationally efficient likelihood-based estimation for discretely observed diffusion processess. Journal of Royal Statistical Society, B. 2006;**68**:333-382

[4] Beskos A, Roberts GO. Exact simulation on diffusions. The Annals of Applied Probability. 2005;**15**:2422-2444

[5] Beskos A, Papaspiliopoulos O, Roberts GO. A factorisation of diffusion measure and finite sample path constructions. Methodology and Computing in Applied Probability. 2008;**10**:85-104

[6] Cai H. Exact sampling using auxiliary variables. In: Statistical Computing Section of ASA Proceedings. 1999

[7] Casella G, Mengersen KL, Robert CP, Titterington DM. Perfect samplers for mixtures of distributions. Journal of the Royal Statistical Society B. 2002;**64**: 777-790

[8] Connor SB, Kendal WS. Perfect simulation of M/G/c queues. Advances in Applied Probability. 2015;**47**(4): 1039-1063

[9] Corcoran JN, Tweedie RL. Perfect sampling from independent Metropolis-Hastings chains. Journal of Statistical Planning and Inference. 2000;**104**(2): 297-314

[10] Cowles MK, Carlin BP. Markov chain Monte Carlo convergence diagnostics: A comparative review. Journal of the American Statistical Association. 1996;**91**:883-904

[11] Dai H. Perfect simulation methods for Bayesian applications [PhD thesis], University of Oxford. 2007

[12] Dai H. Exact Monte Carlo simulation for fork-join networks. Advances in Applied Probability. 2011;**43**(2):484-503

[13] Dai H. Exact simulation for diffusion bridges: An adaptive approach. Journal of Applied Probability. 2014;**51**(2):346-358

[14] Dai H. Exact simulation for fork-join networks with heterogeneous service. International Journal of Statistics and Probability. 2015;**4**(1):19-32

[15] Dai H. A new rejection sampling method without using hat function. Bernoulli. 2017;**23**:2434-2465

[16] Dai H, Pollock M, Roberts G. Monte Carlo fusion. Journal of Applied Probability. 2019;**56**:174-191

[17] Djuric PM, Huang Y, Ghirmai T. Perfect sampling: A review and applications to signal processing. IEEE Transactions on Signal Processing. 2002; **50**(2):345-356

[18] Fearnhead P. Perfect simulation from population genetic models with selection. Theoretical Population Biology. 2001;**59**(4):263-279

[19] Fearnhead P. Direct simulation for discrete mixture distributions. Statistics and Computing. 2005;**15**(2): 125-133

[20] Fill JA. An interruptible algorithm for perfect sampling via Markov chains. The Annals of Applied Probability. 1998;**8**:131-162

[21] Fill JA, Machida M, Murdoch DJ, Rosenthal JS. Extension of Fill's perfect rejection sampling algorithm to general chains. Random Structure and Algorithms. 2000;**17**:290-316

[22] Gilks WR, Wild P. Adaptive rejection sampling for Gibbs sampling. Applied Statistics. 1992;**41**:337-348

[23] Green PJ, Murdoch DJ. Exact sampling for Bayesian inference: Towards general purpose algorithms. Bayesian Statistics. 1998;**6**:301-321

[24] Hansen NR. Geometric ergodicity of discrete-time approximations to multivariate diffusions. Bernoulli. 2003; **9**(4):725-743

[25] Hobert J, Robert C, Titterington D. On perfect simulation for some mixtures of distributions. Statistics and Computing. 1999;**9**:287-298

[26] Hörmann W, Leydold J. Improved perfect slice sampling. Technique Report. 2003

[27] Huber M. Perfect sampling using bounding chains. The Annals of Applied Probability. 2004;**14**:734-753

[28] Kendall WS, Moller J. Perfect simulation using dominating processes on ordered spaces, with application to locally stable point processes. Advances in Applied Probability. 2000;**32**(3):844-865

[29] Leydold J. A rejection technique for sampling from log-concave multivariate distributions. Modeling and Computer Simulation. 1998;**8**(3):254-280. Available from: citeseer.nj.nec.com/leydold98rejection.html

[30] Li C, Srivastava S, Dunson DB. Simple, scalable and accurate posterior interval estimation. Biometrika. 2017; **104**(3):665-680

[31] Minsker S, Srivastava S, Lin L, Dunson DB. Scalable and robust Bayesian inference via the median posterior. In: Proceedings of the 31st International Conference on Machine Learning (ICML-14). 2014. pp. 1656-1664

[32] Mira A, Møller J, Roberts GO. Perfect slice samplers. Journal of the Royal Statistical Society B. 2001;**63**: 593-606

[33] Møller J. Perfect simulation of conditionally specified models. Journal of the Royal Statistical Society, B. 1999; **61**:251-264

[34] Murdoch DJ, Green PJ. Exact sampling from a continuous state space. Scandinavian Journal of Statistics. 1998; **25**:483-502

[35] Neiswanger W, Wang C, Xing E. Asymptotically exact, embarrassingly parallel MCMC. In: Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence (2014). 2014. pp. 623-632

[36] Phillipe A, Robert CP. Perfect simulation of positive Gaussian distributions. Statistics and Computing. 2003;**13**:179-186

[37] Propp JG, Wilson DB. Exact sampling with coupled Markov chains and applications to statistical mechanics. Random Structures and Algorithms. 1996;**9**:223-252

[38] Propp JG, Wilson DB. How to get an exact sample from a generic Markov chain and sample a random spanning tree from a directed graph, both within the cover time. Journal of Algorithms. 1998;**27**:170-217

[39] Robert C, Casella G. Monte Carlo Statistical Methods. Springer Science & Business Media; 2013

[40] Scott SL, Blocker AW, Bonassi FV, Chipman HA, George EI, McCulloch RE. Bayes and big data: The

consensus Monte Carlo algorithm. International Journal of Management Science and Engineering Management. 2016;**11**(2):78-88

[41] Srivastava S, Cevher V, Tan-Dinh Q, Dunson DB. WASP: Scalable Bayes via barycenters of subset posteriors. In: Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2016). 2016. pp. 912-920

[42] Stamatakis A, Aberer AJ. Novel parallelization schemes for large-scale likelihood-based phylogenetic inference. In: 2013 IEEE 27th International Symposium on Parallel and Distributed Processing. 2013. DOI: 10.1109/IPDPS.2013.70

[43] Tan A, Doss H, Hobert JP. Honest importance sampling with multiple Markov chains. Journal of Computational and Graphical Statistics. 2015;**24**(3):792-826

[44] Uney M, Clark DE, Julier SJ. Distributed fusion of PHD filters via exponential mixture densities. IEEE Journal of Selected Topics in Signal Processing. 2013;**7**(3):521-531

[45] Wakefield JC, Gelfand AE, Smith AFM. Efficient generation of random variates via the ratio-of-uniforms method. Statistics and Computing. 1991;**1**:129-133

[46] Wang X, Dunson. Parallelizing MCMC via Weierstrass Sampler. arXiv preprint arXiv:1312.4605. 2013

[47] Wilson DB. Generating random spanning trees more quickly than the cover time. In: Annual ACM Symposium on Theory of Computing Archive, Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing. 1996. pp. 296-303

[48] Wilson DB. How to couple from the past using a read-once source of randomness. Random Structures and Algorithms. 2000;**16**:85-113

[49] Wilson DB. Layered multishift coupling for use in perfect sampling algorithms (with a primer on CFTP). In: Madras N, editor. Monte Carlo Methods. Vol. 26. Fields Institute Communications. American Mathematical Society; 2000. pp. 141-176