Bayesian Inference in the Presence of Intractable Normalizing Functions

Jaewoo Park and Murali Haran*
Department of Statistics, Pennsylvania State University

December 15, 2016

Abstract

Models with intractable normalizing functions arise frequently in statistics. Common examples of such models include exponential random graph models for social networks and Markov point processes for ecology and disease modeling. Inference for these models is complicated because the normalizing functions of their probability distributions include the parameters of interest. In Bayesian analysis they result in so-called doubly intractable posterior distributions which pose significant computational challenges. Several Monte Carlo methods have emerged in recent years to address Bayesian inference for such models. We provide a framework for understanding the algorithms and elucidate connections among them. Through multiple simulated and real data examples, we compare and contrast the computational and statistical efficiency of these algorithms and discuss their theoretical bases. Our study provides practical recommendations for practitioners along with directions for future research for MCMC methodologists.

Keywords: Markov Chain Monte Carlo, Doubly intractable distributions, Importance sampling, Markov random field models, Social network models, Point process

*Jaewoo Park is Ph.D. student (e-mail:jzp191@psu.edu) and Murali Haran is Professor, Department of Statistics, Pennsylvania State University, 326 Thomas Building, University Park, PA 16802, USA (e-mail:muh10@psu.edu). We are grateful to Anne-Marie Lyne, Ick Hoon Jin and Yves Atchade for helpful discussion and providing relevant example code. We give thanks to Faming Liang, Galin Jones and John Hughes for comments. MH was partially supported by the National Science Foundation through NSF-DMS-1418090.

1 Introduction

Markov chain Monte Carlo (MCMC) has been used to routinely carry out Bayesian inference for an enormous variety of complicated models (cf. Brooks et al., 2011). However, inference for Bayesian models with intractable normalizing functions, where the normalizing constant of the model is itself a function of parameters of interest, is still far from routine. There are many well known models that have intractable functions, for instance the class of exponential family random graph models and its variants (cf. Robins et al., 2007; Hunter and Handcock, 2012) which are popular models for social networks. Non-Gaussian Markov random field models in spatial statistics (cf. Besag, 1974; Hughes et al., 2011, for a review) also have intractable normalizing functions. In fact, it is worth noting that the Ising model (Lenz, 1920; Ising, 1925), which is a non-Gaussian Markov random field, appears in the landmark paper on the Metropolis algorithm (Metropolis et al., 1953). While the Metropolis algorithm provides an elegant way to simulate from this model for a given parameter value, the paper did not consider the more difficult problem of performing inference for this model.

Consider $h(\mathbf{x}|\theta)$, an unnormalized probability model for a random variable $\mathbf{x} \in \mathcal{X}$ given a parameter vector $\theta \in \Theta$, with a normalizing function $Z(\theta) = \int_{\mathcal{X}} h(\mathbf{x}|\theta)d\mathbf{x}$. Let $p(\theta)$ be the prior distribution for θ. The likelihood function, $L(\theta|\mathbf{x})$ is $h(\mathbf{x}|\theta)/Z(\theta)$ and the posterior distribution of θ is

$$\pi(\theta|\mathbf{x}) \propto p(\theta)\frac{h(\mathbf{x}|\theta)}{Z(\theta)}. \tag{1}$$

The problem in constructing an MCMC algorithm stems from the fact that $Z(\theta)$ cannot be easily evaluated and the acceptance probability at each step of the Metropolis-Hastings (MH) algorithm (Metropolis et al., 1953; Hastings, 1970) involves evaluating $Z(\theta)$ both at the current and proposed value of θ.

There have been several proposals to address the intractable normalizing function problem in a maximum likelihood context. Besag (1974) proposed the maximum pseudolikelihood estimate (MPLE) which maximizes the pseudolikelihood, a particular likelihood approximation that does not require $Z(\theta)$. MPLE does not, in general, define a likelihood function. Furthermore, MPLE may be a reasonable estimator when there is weak dependence among data points relative to the size of the data set but in other cases its performance is unsatisfactory. Younes (1988) proposed a stochastic gradient algorithm to solve normal equations to find the maximum likelihood estimate (MLE) for models with intractable normalizing functions. However the step size and starting point must be selected carefully, otherwise the algorithm becomes slow and does not converge (Ibáñez and Simó, 2003). Geyer and Thompson (1992) propose MCMC-MLE which is based on maximizing a Monte Carlo approximation to the likelihood; this approximation is based on an importance sampling approximation of $Z(\theta)$. This is an elegant and theoretically justified algorithm. In practice, MCMC-MLE suffers from some of the usual challenges faced by importance sampling approaches, namely that for an accurate approximation to MLE, the initial value for the algorithm should be reasonably close to the MLE. Some of these issues may be partially addressed by umbrella sampling (Torrie and Valleau, 1977; Geyer, 2011), which involves sampling from mixtures of importance sampling distributions. However approximating standard errors can also be difficult in situations where analytical gradients for the unnormalized likelihood are unavailable and using bootstrap techniques may be computationally infeasible (cf. Goldstein et al., 2015). In such cases and also in situations where there is an interest in incorporating prior information about the parameters or avoiding model degeneracies (cf. Handcock, 2003), Bayesian alternatives may be preferable. In this manuscript, we focus on Bayesian inference; we refer readers to Geyer (2011) for a general overview and Hunter et al.

(2012) for a review of recent MCMC-MLE methods for exponential random graph models.

Several MCMC algorithms have recently been proposed for Bayesian inference in the presence of intractable normalizing functions. These algorithms may be broadly classified into two general if somewhat overlapping categories: (1) algorithms where the introduction of a well chosen auxiliary variable results in the normalizing function (or a ratio of normalizing functions) canceling out in the Metropolis-Hastings acceptance probability, and (2) directly approximating the normalizing function (or a ratio of normalizing functions), and substituting the approximation into the acceptance probability. Here we will refer to the first as an *auxiliary variable approach* and the second as a *likelihood approximation approach* (see also a discussion in Liang et al., 2016). In addition, MCMC algorithms may also be classified as asymptotically exact or asymptotically inexact. For asymptotically exact algorithms the Markov chain's stationary distribution is exactly equal to the desired posterior distribution. On the other hand, asymptotically inexact or "noisy" algorithms generate Markov chains without this property; even asymptotically the samples generated only follow the target distribution approximately.

In what follows we discuss several MCMC algorithms. We provide an explanation of the ideas underpinning each algorithm along with figures that summarize them and make it easier to see how they are related. We also discuss theoretical justifications and practical implementation issues. We carry out a comparative study by using three different examples: an Ising model, a social network model, and a spatial point process. We provide some guidance about potential advantages and disadvantages of each algorithm along with connections among them, providing some future avenues for research. The remainder of this paper is organized as follows. In Section 2 we discuss several auxiliary variable algorithms. In Section 3 we cover several likelihood

approximation algorithms. In Section 4 we describe the application of the algorithms in the context of three different case studies and provide some insights based on our results. In particular, we discuss in detail the computational complexity of the algorithms in Section 5. We point out connections between the algorithms in Section 6 along with general guidelines and recommendations based on our study. We conclude with a summary in Section 7.

2 Auxiliary variable approaches

In this section, we review several auxiliary variable approaches. Here, the target distribution includes both the parameter of interest as well as an auxiliary variable. By a clever choice of the auxiliary variable proposal the intractable functions get cancelled in the acceptance probability of the Metropolis-Hastings algorithm. What distinguishes the different auxiliary variable algorithms from each other is how the auxiliary variable is sampled.

2.1 Auxiliary variable MCMC

Møller et al. (2006) introduce an auxiliary variable $\mathbf{y}$ with the conditional distribution $f(\mathbf{y}|\theta, \mathbf{x})$ so that the intractable terms are cancelled in the acceptance probability of the Metropolis-Hastings algorithm. Suppose the original target distribution is $\pi(\theta|\mathbf{x}) \propto p(\theta)h(\mathbf{x}|\theta)/Z(\theta)$. Then the augmented target distribution is $\pi(\theta, \mathbf{y}|\mathbf{x}) \propto f(\mathbf{y}|\theta, \mathbf{x})p(\theta)h(\mathbf{x}|\theta)/Z(\theta)$ whose marginal distribution becomes $\int_{\mathcal{X}} \pi(\theta, \mathbf{y}|\mathbf{x})d\mathbf{y} = \pi(\theta|\mathbf{x})$, the original target distribution. Now consider a joint proposal distribution for $\{\theta, \mathbf{y}\}$ updates which can be factorized as $q(\theta', \mathbf{y}'|\theta, \mathbf{y}) = q(\mathbf{y}'|\theta')q(\theta'|\theta)$. Møller et al. (2006) take the proposal for the auxiliary variable $q(\mathbf{y}'|\theta')$ as $h(\mathbf{y}'|\theta')/Z(\theta')$ which is the same as the model for the data $\mathbf{x}$ given the parameter value θ' proposed from $q(\theta'|\theta)$. The

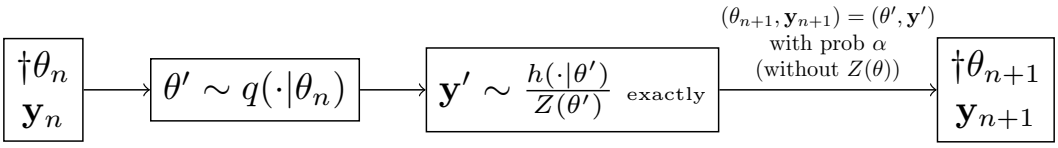

Figure 1: Illustration of the (n+1) step update for auxiliary variable MCMC. The dagger symbols indicate the parameter of interest.

resulting algorithm for the above joint distribution with this proposal distribution has Metropolis-Hastings acceptance probability

$$\alpha = \min\left\{1, \frac{f(\mathbf{y}'|\theta', \mathbf{x})p(\theta')h(\mathbf{x}|\theta')\cancel{Z(\theta)}h(\mathbf{y}|\theta)\cancel{Z(\theta')}q(\theta|\theta')}{f(\mathbf{y}|\theta, \mathbf{x})p(\theta)h(\mathbf{x}|\theta)\cancel{Z(\theta')}h(\mathbf{y}'|\theta')\cancel{Z(\theta)}q(\theta'|\theta)}\right\}. \tag{2}$$

The resulting Metropolis-Hastings acceptance probability (2) does not contain the normalizing functions because they get cancelled out. Since $\int_{\mathcal{X}} \pi(\theta, \mathbf{y}|\mathbf{x})d\mathbf{y} = \pi(\theta|\mathbf{x})$, the marginal θ samples follow the original target distribution. We will henceforth use AVM to refer to this auxiliary variable MCMC algorithm. We summarize the algorithm in Figure 1. We should note that updating $\mathbf{y}$ requires drawing a sample with exact distribution $h(\cdot|\theta')/Z(\theta')$. For some models this can be achieved via perfect sampling (Propp and Wilson, 1996), a clever method that uses bounding Markov chains to construct a sampler where the draws are exactly (not just asymptotically) from the target distribution. Although perfect sampling is possible for some models, for instance certain Markov random field (MRF) models, it is not trivial to construct a perfect sampler in general. This is a major practical limitation of the AVM.

Components to be tuned: Møller et al. (2006) report that the choice of $f(\mathbf{y}|\theta, \mathbf{x})$ impacts the mixing of the chain. Suppose $Z(\theta)$ is known so that we can choose $f(\mathbf{y}|\theta, \mathbf{x}) = h(\mathbf{y}|\theta)/Z(\theta)$. Then it is easily seen that (2) is identical to the

Algorithm 1 Auxiliary variable MCMC (AVM)

Given $\{\theta_n, \mathbf{y}_n\} \in \Theta \times \mathcal{X}$ at nth iteration.

1. Propose $\theta' \sim q(\cdot|\theta_n)$.
2. Propose the auxiliary variable exactly from probability model at θ':
   $\mathbf{y}' \sim \frac{h(\cdot|\theta')}{Z(\theta')}$ using perfect sampling.
3. Accept $\{\theta_{n+1}, \mathbf{y}_{n+1}\} = \{\theta', \mathbf{y}'\}$ with probability
   $\alpha = \min\left\{1, \frac{f(\mathbf{y}'|\theta', \mathbf{x})p(\theta')h(\mathbf{x}|\theta')h(\mathbf{y}_n|\theta_n)q(\theta_n|\theta')}{f(\mathbf{y}_n|\theta_n, \mathbf{x})p(\theta_n)h(\mathbf{x}|\theta_n)h(\mathbf{y}'|\theta')q(\theta'|\theta_n)}\right\}$, else reject (set $\{\theta_{n+1}, \mathbf{y}_{n+1}\} = \{\theta_n, \mathbf{y}_n\}$).

acceptance probability of the Metropolis-Hastings algorithm with the stationary distribution $\pi(\theta|\mathbf{x})$, which implies that this chain has the same convergence properties as the Markov chain where the normalizing function is known and the same proposal $q(\theta'|\theta)$ is used for θ. Therefore, a reasonable choice of conditional distribution for the auxiliary variable is $f(\mathbf{y}|\theta, \mathbf{x})$ that approximates $h(\mathbf{y}|\theta)/Z(\theta)$. A simple choice is $f(y|\theta, \mathbf{x}) = h(\mathbf{y}|\widehat{\theta})/Z(\widehat{\theta})$, where $\widehat{\theta}$ may be an approximation to the MLE. $\widehat{\theta}$ should be predetermined before implementing the AVM algorithm. For example, the maximum pseudolikelihood estimate (MPLE) proposed by Besag (1974) may be an option though for some problems the MPLE may be a poor approximation to the MLE. Then $f(\mathbf{y}'|\theta', \mathbf{x})/f(\mathbf{y}|\theta, \mathbf{x}) = h(\mathbf{y}'|\widehat{\theta})/h(\mathbf{y}|\widehat{\theta})$, which makes it possible to calculate (2) because there are no intractable terms. Another possible choice of $f(\mathbf{y}|\theta, \mathbf{x})$ is using a normalizable distribution without $Z(\theta)$. AVM mixes better as $f(\mathbf{y}|\theta, \mathbf{x})$ more closely resembles $h(\mathbf{y}|\theta)/Z(\theta)$.

Theoretical justification: Since the Markov Chain satisfies detailed balance with respect to the augmented target distribution, the marginal distribution of which is the posterior distribution of θ, AVM is an asymptotically exact MCMC algorithm. However, to achieve detailed balance condition, we need to sample $\mathbf{y}$ exactly from

the likelihood function $h(\mathbf{y}|\theta')/Z(\theta')$.

2.2 The Exchange Algorithm

Appearing almost simultaneously with Møller et al. (2006), the exchange algorithm (Murray et al., 2006) also constructs an augmented target distribution and updates the augmented state via the Metropolis-Hastings algorithm. Consider the auxiliary variable $\mathbf{y}$ which follows $h(\mathbf{y}|\theta')/Z(\theta')$ and conditional distribution of θ' for given θ as $q(\theta'|\theta)$ where θ is the parameter setting of data $\mathbf{x}$. Then the augmented joint distribution is

$$\pi(\theta, \theta', \mathbf{y}|\mathbf{x}) \propto p(\theta)L(\theta|\mathbf{x})q(\theta'|\theta)L(\theta'|\mathbf{y}) = p(\theta)\frac{h(\mathbf{x}|\theta)}{Z(\theta)}q(\theta'|\theta)\frac{h(\mathbf{y}|\theta')}{Z(\theta')}. \tag{3}$$

For this augmented distribution, $\{\theta', \mathbf{y}\}$ is updated through block-Gibbs samplers; θ' is generated from the proposal $q(\cdot|\theta)$ and the auxiliary variable $\mathbf{y}$ is generated from $h(\cdot|\theta')/Z(\theta')$. The first two arrows in Figure 2 correspond to the update of $\{\theta', \mathbf{y}\}$. Then θ is updated through exchanging parameter settings. Let $\{\theta, \theta'\}$ be the current parameter settings for $\{\mathbf{x}, \mathbf{y}\}$. Consider a swapping proposal $s(\{\theta^*, \theta'^*\}|\{\theta, \theta'\}) = \delta(\theta^* - \theta')\delta(\theta'^* - \theta)$, where δ denotes the Dirac delta function. After swapping is proposed, data $\mathbf{x}$ follows θ' instead of θ and the auxiliary variable $\mathbf{y}$ follows θ instead of θ'. The symmetric swapping proposal results in the acceptance probability (α),

$$\min\left\{1, \frac{\cancel{s(\{\theta, \theta'\}|\{\theta^*, \theta'^*\})}\,\pi(\theta', \theta, \mathbf{y}|\mathbf{x})}{\cancel{s(\{\theta^*, \theta'^*\}|\{\theta, \theta'\})}\,\pi(\theta, \theta', \mathbf{y}|\mathbf{x})}\right\} = \min\left\{1, \frac{p(\theta')h(\mathbf{x}|\theta')\cancel{Z(\theta)}h(\mathbf{y}|\theta)\cancel{Z(\theta')}q(\theta|\theta')}{p(\theta)h(\mathbf{x}|\theta)\cancel{Z(\theta')}h(\mathbf{y}|\theta')\cancel{Z(\theta)}q(\theta'|\theta)}\right\}, \tag{4}$$

with intractable terms cancelled. Since $\int_{\mathcal{X}}\int_{\Theta}\pi(\theta, \theta', \mathbf{y}|\mathbf{x})d\theta' d\mathbf{y} = \pi(\theta|\mathbf{x})$, the marginal θ samples follow the original target distribution, as shown in Figure 2. This idea of swapping parameter settings is connected to the parallel tempering

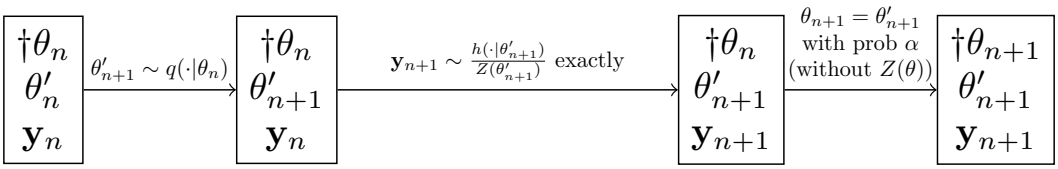

Figure 2: Illustration for the exchange algorithm. The dagger symbols indicate the parameter of interest.

algorithm (Geyer, 1991). Parallel tempering swaps the states while preserving joint distributions in the product space. The exchange algorithm also swaps parameter settings between $\mathbf{x}$ and $\mathbf{y}$, while preserving the augmented distribution. Since we do not need to keep $\{\theta', \mathbf{y}\}$ in the actual implementation of the algorithm, the exchange algorithm may be simply written as Algorithm 2.

Algorithm 2 Exchange algorithm

Given $\theta_n \in \Theta$ at nth iteration.

1. Propose $\theta' \sim q(\cdot|\theta_n)$.
2. Generate the auxiliary variable exactly from probability model at θ':
   $\mathbf{y} \sim \frac{h(\cdot|\theta')}{Z(\theta')}$ using perfect sampling.
3. Accept $\theta_{n+1} = \theta'$ with probability
   $\alpha = \min\left\{1, \frac{p(\theta')h(\mathbf{x}|\theta')h(\mathbf{y}|\theta_n)q(\theta_n|\theta')}{p(\theta_n)h(\mathbf{x}|\theta_n)h(\mathbf{y}|\theta')q(\theta'|\theta_n)}\right\}$, else reject (set $\{\theta_{n+1}, \mathbf{y}_{n+1}\} = \{\theta_n, \mathbf{y}_n\}$).

The exchange algorithm can also be extended through "bridging" (Murray et al., 2006), Algorithm 3. The idea may be summarized as follows. In the acceptance probability of Algorithm 2, one may think of the ratio $h(\mathbf{x}|\theta')/h(\mathbf{x}|\theta)$ as measuring whether the proposed θ' explains the data $\mathbf{x}$ better than current θ or not. Also, $h(\mathbf{y}|\theta)/h(\mathbf{y}|\theta')$ represents how well the auxiliary variable $\mathbf{y}$ is supported under θ versus θ. Therefore, even if θ' is a much better candidate than θ under the data $\mathbf{x}$,

swapping can be rejected if the auxiliary variable $\mathbf{y}$ is improbable under θ, which can lead to slow mixing of the chain. To improve this, annealed importance sampling (AIS) (Neal, 1996, 2001) can be combined with the exchange algorithm. Instead of sampling a single auxiliary variable, a series of auxiliary variables $\{\mathbf{y}_0, \mathbf{y}_1, ..., \mathbf{y}_K\}$ are sampled from the intermediate distributions between $h(\mathbf{y}|\theta')$ and $h(\mathbf{y}|\theta)$. Intermediate distributions can be constructed as

$$f_k(\mathbf{y}|\theta, \theta') = h(\mathbf{y}|\theta')^{1-\beta_k} h(\mathbf{y}|\theta)^{\beta_k}, \beta_k = \frac{k}{K+1}, \ k = 1, \ldots, K. \tag{5}$$

After proposing θ', generate $\mathbf{y}_0$ from $h(\mathbf{y}|\theta')/Z(\theta')$. Then generate each $\mathbf{y}_k$ from $T_k(\mathbf{y}_k|\mathbf{y}_{k-1})$, the Metropolis-Hastings (MH) transition kernel whose stationary distribution is $f_k(\mathbf{y}|\theta, \theta')$. This bridging step helps to generate $\mathbf{y}_K$ that is more probable under θ.

Algorithm 3 Exchange algorithm with bridging

Given $\theta_n \in \Theta$ at nth iteration.

1. Propose $\theta' \sim q(\cdot|\theta_n)$.
2. Generate the series of auxiliary variables $\{\mathbf{y}_0, \mathbf{y}_1, ..., \mathbf{y}_K\}$ from (5):
   $\mathbf{y}_0 \sim \frac{h(\cdot|\theta')}{Z(\theta')}$ using perfect sampling.
   $\mathbf{y}_k \sim T_k(\mathbf{y}_k|\mathbf{y}_{k-1})$ via 1-step MH update for $k = 1, ..., K$.
3. Accept $\theta_{n+1} = \theta'$ with bridging probability
   $\alpha = \min\left\{1, \frac{p(\theta')h(\mathbf{x}|\theta')q(\theta_n|\theta')}{p(\theta_n)h(\mathbf{x}|\theta_n)q(\theta'|\theta_n)} \prod_{k=0}^{K} \frac{f_{k+1}(\mathbf{y}_k|\theta_n,\theta')}{f_k(\mathbf{y}_k|\theta_n,\theta')}\right\}$,
   else reject (set $\{\theta_{n+1}, \mathbf{y}_{n+1}\} = \{\theta_n, \mathbf{y}_n\}$).

Components to be tuned: The basic exchange algorithm does not require any tuning besides the usual tuning of the proposal for θ. For the extended version, there are many options for bridging but even here (5) provides an automated

schedule. Larger K in (5) can lead to better mixing of the chain at the expense of computing time. Effective sample size (ESS) (Kass et al., 1998) provides a rough diagnostic for how well the chain is mixing. Therefore, effective sample size per unit time (ESS/T) can be used to determine K that provides a compromise between mixing and computational cost (Murray, 2007).

Theoretical justification: Both the exchange algorithm and the extended exchange algorithm with bridging are asymptotically exact MCMC in that constructed Markov chains satisfy detailed balance condition with respect to the target distribution. However just like AVM, the exchange algorithm is of limited applicability because it requires perfect sampling to generate the auxiliary variable.

2.3 The Double Metropolis-Hastings Sampler

Both AVM and the exchange algorithm require perfect sampling which can be very expensive or impossible for complicated probability models. Liang (2010) replaces perfect sampling for the auxiliary variable with a standard Metropolis-Hastings algorithm; the last state of the resulting Markov chain is then treated like a draw from the perfect sampler. This is called Double Metropolis-Hastings (DMH) because a Metropolis-Hastings algorithm is used within another Metropolis-Hastings algorithm. As in Figure 3, one is the "outer sampler" to generate θ draws while the other is the "inner sampler" used to generate the auxiliary variable $\mathbf{y}$.

Define $T^m_{\theta'}(\mathbf{y}|\mathbf{x})$ as m-MH updates from $\mathbf{x}$ to $\mathbf{y}$ under θ' whose stationary distribution is $h(\mathbf{y}|\theta')/Z(\theta')$; this satisfies the detailed balanced condition

$$h(\mathbf{x}|\theta')T^m_{\theta'}(\mathbf{y}|\mathbf{x}) = h(\mathbf{y}|\theta')T^m_{\theta'}(\mathbf{x}|\mathbf{y}). \tag{6}$$

Plugging this result into (4) yields

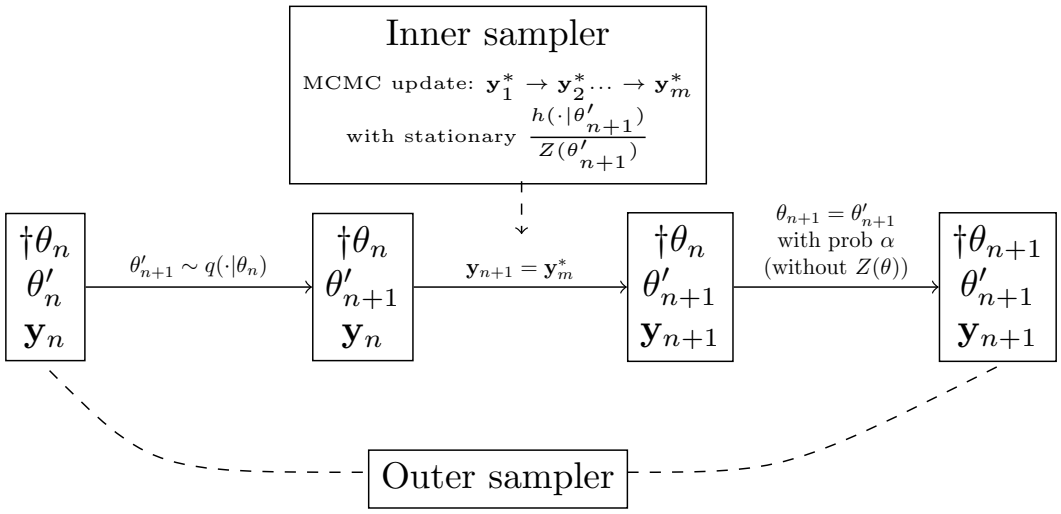

Figure 3: Illustration of the Double Metropolis-Hastings algorithm. The dagger symbols indicate the parameter of interest.

$$\alpha = \min\left\{1, \frac{p(\theta')h(\mathbf{y}|\theta)T_{\theta'}^m(\mathbf{x}|\mathbf{y})q(\theta|\theta')}{p(\theta)h(\mathbf{x}|\theta)T_{\theta'}^m(\mathbf{y}|\mathbf{x})q(\theta'|\theta)}\right\}, \tag{7}$$

which is the acceptance probability of the DMH algorithm. DMH approximates perfect sampling through m steps of an (inner) Metropolis-Hastings algorithm. A similar approximate approach which replaces perfect sampling with MH updates for the social network models is discussed in Caimo and Friel (2011). As we show later, DMH is simpler to implement and computationally more efficient relative to all other algorithms. However, its computational efficiency directly depends on the efficiency of the inner sampler. If the inner sampler update $T_{\theta'}^m(\mathbf{y}|\mathbf{x})$ is expensive or needs large m to get probable auxiliary variable $\mathbf{y}$, then DMH becomes computationally expensive. Since m is usually proportional to the dimension of the data $\mathbf{x}$, the inner sampler is the main computational bottleneck for large data.

Algorithm 4 Double Metropolis-Hastings algorithm

Given $\theta_n \in \Theta$ at nth iteration.

1. Propose $\theta' \sim q(\cdot|\theta_n)$.
2. Generate the auxiliary variable approximately from probability model at θ':
   $\mathbf{y} \sim T^m_{\theta'}(\cdot|\mathbf{x})$ using m-MH updates.
3. Accept $\theta_{n+1} = \theta'$ with probability
   $\alpha = \min\left\{1, \frac{p(\theta')h(\mathbf{x}|\theta')h(\mathbf{y}|\theta_n)q(\theta_n|\theta')}{p(\theta_n)h(\mathbf{x}|\theta_n)h(\mathbf{y}|\theta')q(\theta'|\theta_n)}\right\} = \min\left\{1, \frac{p(\theta')h(\mathbf{y}|\theta_n)T^m_{\theta'}(\mathbf{x}|\mathbf{y})q(\theta_n|\theta')}{p(\theta_n)h(\mathbf{x}|\theta_n)T^m_{\theta'}(\mathbf{y}|\mathbf{x})q(\theta'|\theta_n)}\right\}$,
   else reject (set $\{\theta_{n+1}, \mathbf{y}_{n+1}\} = \{\theta_n, \mathbf{y}_n\}$).

Components to be tuned: Users need to decide the number of MH updates m to generate $\mathbf{y}$. There are numerous stopping rules for MCMC if computing expectations is the goal (cf. Flegal et al., 2008; Gong and Flegal, 2015). However, here determining convergence to the stationary distribution is of interest (Jones and Hobert, 2001; Rosenthal, 1995), which is a very challenging problem in general. A simple heuristic for determining m is to set m to be proportional to the size of the data, for instance $m = 5n$, where n is the size of the data. We recommend that DMH should be run for larger m values, for instance $m = 10n$ to confirm that the results do not change much as m is increased.

Theoretical justification: DMH is an asymptotically inexact algorithm because the detailed balance condition does not hold for the outer sampler unless the inner sampler length (m) approaches infinity; in practice the inner sampler length is of course finite.

2.4 Adaptive Exchange Algorithm

In order to address the asymptotic inexactness of DMH, Liang et al. (2016) propose the adaptive exchange algorithm (AEX). AEX runs two chains simultaneously: the auxiliary chain and the target chain (see Figure 4). The auxiliary chain generates a sample from a family of distributions, $\{h(\mathbf{x}|\theta^{(1)})/Z(\theta^{(1)}), ..., h(\mathbf{x}|\theta^{(d)})/Z(\theta^{(d)})\}$, where $\{\theta^{(1)}, ..., \theta^{(d)}\}$ are pre-specified "particles" covering a parameter space. The generated sample is stored at each iteration. Then the target chain generates a posterior sample from the $\pi(\theta|\mathbf{x})$ via the exchange algorithm. The difference is that the auxiliary variable $\mathbf{y}$ is sampled from the cumulative samples in the auxiliary chain until current iteration (resampling). With increasing iterations, cumulative samples from the auxiliary chain grow, so that the resampling of $\mathbf{y}$ in the target chain becomes identical to exact sampling of $\mathbf{y}$. Then similar to the exchange algorithm, the marginal distribution of θ converges to the target $\pi(\theta|\mathbf{x})$. AEX is therefore an asymptotically exact algorithm that does not require perfect sampling. In order to make it easy for readers to understand AEX, we first provide an outline of the algorithm.

- For the nth iteration of AEX, we use the following notation.
  - Particles: $\{\theta^{(1)}, ..., \theta^{(d)}\}$, each $\theta^{(i)} \in \Theta$. These particles are fixed through the algorithm.
  - Particle index: $I_n \in \{1, ..., d\}$ returns index of a chosen particle at nth iteration.
  - Auxiliary data: $\mathbf{x}_n \in \mathcal{X}$ is an approximate sample from the probability model at selected particle $\theta^{(I_n)}$. That is, $\mathbf{x}_n \sim h(\cdot|\theta^{(I_n)})/Z(\theta^{(I_n)})$.
  - Normalizing function approximation at each particle: $\mathbf{w}_n = \{w_n^{(1)}, ..., w_n^{(d)}\} \in W$. For $i = 1, ..., d$, as n gets large $w_n^{(i)}$ converges to $Z(\theta^{(i)})$ via the stochastic approximation algorithm (Liang et al., 2007).

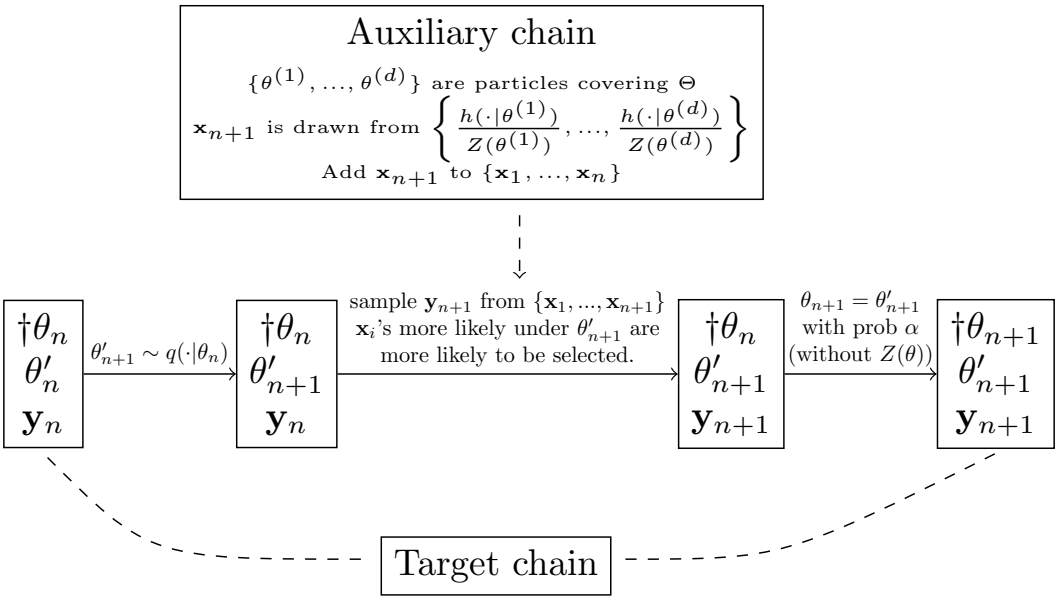

Figure 4: Illustration for the Adaptive Exchange algorithm. The dagger symbols indicate the parameter of interest.

- Cumulative information: $H_n = \cup_{j=1}^{n}\{I_j, \mathbf{x}_j, \mathbf{w}_j\}$ is necessary for constructing the algorithm. Therefore, $\{I_n, \mathbf{x}_n, \mathbf{w}_n\}$ should be stored at each iteration.
- Posterior sample: $\theta_n \in \Theta$

- **Sketch of Adaptive Exchange algorithm**: For the $(n+1)$st iteration,

  Auxiliary chain update

  - I_{n+1}: Consider k-nearest particles from $\theta^{(I_n)}$. The distance can be Euclidean distance. Using an acceptance probability (discussed below), we select among this set of nearest neighbors. I_{n+1} is the index of the selected particle.
  - $\mathbf{x}_{n+1}$: Using $\mathbf{x}_n$ as starting value, construct MH updates whose stationary distribution is $h(\mathbf{x}|\theta^{(I_{n+1})})/Z(\theta^{(I_{n+1})})$. $\mathbf{x}_{n+1}$ is the result of these updates.
  - $\mathbf{w}_{n+1}$: Obtain from I_{n+1} and $\mathbf{w}_n$ using stochastic approximation algorithm (Liang et al., 2007).
  - H_{n+1}: Cumulative information is updated. i.e. $H_{n+1} = H_n \cup \{I_{n+1}, \mathbf{x}_{n+1}, \mathbf{w}_{n+1}\}$.

  Target chain update

  - θ_{n+1}: Sampled from $\pi(\theta|\mathbf{x})$ via the exchange algorithm. The auxiliary variable $\mathbf{y}$ is sampled from the set $\{\mathbf{x}_1, ..., \mathbf{x}_{n+1}\}$ with probability (8). This avoids perfect sampling.

AEX updates a non-Markovian stochastic process $\{I_{n+1}, \mathbf{x}_{n+1}, \mathbf{w}_{n+1}, \theta_{n+1}\} \in \{1, ..., d\} \times \mathcal{X} \times R^d \times \Theta$ in the $(n+1)$st iteration. For each iteration, the auxiliary chain generates a sample $\mathbf{x}_{n+1}$ from the mixture distribution, $(1/d)\sum_{i=1}^{d} h(\mathbf{x}|\theta^{(i)})/Z(\theta^{(i)})$ through stochastic approximation Monte Carlo (SAMC) (Liang et al., 2007). Particles are

d-number of points in the grid over a parameter space Θ. Particles are chosen so that they can cover the important region of the parameter space and the sample spaces of neighboring distributions $\{h(\mathbf{x}|\theta^{(i)})/Z(\theta^{(i)})\}$ overlap each other. A clever choice of particles can improve mixing of the auxiliary chain and is important for AEX to be a practical algorithm. We refer the reader to the supplementary material for strategies for choosing particles. For the $(n+1)$st update of AEX, cumulative information H_{n+1} is constructed through the auxiliary chain as in the the sketch. For $(n+1)$st iteration of the algorithm, $\{I_{n+1}, \mathbf{x}_{n+1}, \mathbf{w}_{n+1}\}$ is updated for given $\{I_n, \mathbf{x}_n, \mathbf{w}_n\}$ using SAMC and then added to H_n to construct H_{n+1}. Hence, the accumulated information (H_{n+1}) becomes larger with each iteration of AEX. See Algorithm 5 for details about updating.

Then the target chain generates θ_{n+1} from the posterior using the exchange algorithm. For proposed θ', the auxiliary variable $\mathbf{y}$ is sampled from $\{\mathbf{x}_1, ..., \mathbf{x}_{n+1}\}$ through a dynamic importance sampling procedure. The resampling probability is

$$P(\mathbf{y} = \mathbf{x}_l|\theta') \propto \sum_{j=1}^{n+1} w_j^{(I_j)} \frac{h(\mathbf{x}_j|\theta')}{h(\mathbf{x}_j|\theta^{(I_j)})} 1\{\mathbf{x}_j = \mathbf{x}_l\}, \quad l = 1, ..., n+1. \tag{8}$$

In (8), $h(\mathbf{x}_j|\theta^{(I_j)})/w_j^{(I_j)}$ is the importance function at jth iteration. Since the importance function changes as I_j varies, it is called a dynamic importance function. Here, $\mathbf{x}_l$'s more likely under θ' are more likely to be sampled. Liang et al. (2016) show that the distribution of resampled $\mathbf{y}$ converges to $h(\mathbf{y}|\theta')/Z(\theta)$ as n increases. Since the proposal for $\mathbf{y}$ in (8) changes with each iteration, AEX appears at first to be an adaptive MCMC algorithm. However, unlike basic adaptive MCMC algorithms the target distribution also varies as H_{n+1} grows.

In Step 2(a) of Algorithm 5, $\{a_n\}$ is called the gain factor, which is the step size of update for the abundance factor $\mathbf{w}_n$. A large gain factor can lead to faster movement around all $\{h(\mathbf{x}|\theta^{(i)})/Z(\theta^{(i)})\}$. Since $a_n = n_0/\max(n_0, n)$ becomes smaller

as n increases, movement between particles becomes slower with increasing iterations. Liang et al. (2007) suggest larger n_0 for complicated problems so that makes it fast to move around all state space. Step 2(b) illustrates varying truncation (Jin and Liang, 2013), which can help the convergence of $\mathbf{w}_n$ regardless of starting point. For $\mathbf{w}_n \in W$, consider the sequence of set $\{K_s\} \in W$, which satisfies $\cup_{s\geq 0} K_s = W$ and $K_s \subset K_{s+1}^{\circ}$, where K_{s+1}° denotes the interior of set K_{s+1}. A truncation function T maps a point in $\mathcal{X} \times W$ to a random point in $\mathcal{X}_0 \times K_0$ for $\mathcal{X}_0 \subset \mathcal{X}$, and σ_n is the number of truncations that occurred by the nth iteration ($\sigma_0 = 0$). From such truncation, when an updated $\mathbf{w}_{n+1/2}$ is outside the interesting target region (K_s), it is reinitialized with a smaller gain factor and expanded target region ($K_{s+1} \supset K_s$). This varying truncation can help SAMC to find an appropriate step size $\{a_n\}$ and starting point of abundance factor automatically (Jin and Liang, 2013). The reader may understand such varying truncation method as a safeguard for when the abundance factor becomes degenerate.

In practice, Liang et al. (2016) suggest that only the auxiliary chain is implemented N_1 iterations at first to construct the database, and then the auxiliary and target chain can be run simultaneously. In detail, through N_1 preliminary iterations of the auxiliary chain, information: $\cup_{j=1}^{N_1}\{I_j, \mathbf{x}_j, \mathbf{w}_j\}$ is constructed. Using this information as initial $H_0 = \cup_{j=1}^{N_1}\{I_j, \mathbf{x}_j, \mathbf{w}_j\}$, we can implement the entire Algorithm 5. In this case, the size of cumulative information become $|H_0| = N_1$, and $|H_{n+1}| = N_1 + n + 1$ (without preliminary step, $|H_0| = 0$, and $|H_{n+1}| = n + 1$). The reason of conducting preliminary step is because the early performance of the target chain can be affected by initial starting values of the auxiliary chain. This preliminary step can construct an initial database H_0 and improves the mixing of the target chain.

Algorithm 5 Adaptive Exchange algorithm (details)

Given $\{I_n, \mathbf{x}_n, \mathbf{w}_n, \theta_n\} \in \{1, ..., d\} \times \mathcal{X} \times W \times \Theta$ at nth iteration.

Auxiliary Chain: Constructing cumulative information H_{n+1}.

1. Update I_n or $\mathbf{x}_n$.

(a) With probability 0.5, update I_n at $\mathbf{x}_n$:

Propose $\theta^{(I')}$ from the k-nearest particles of $\theta^{(I_n)}$ with equal probability:

Accept $\{I_{n+1}, \mathbf{x}_{n+1}\} = \{I', \mathbf{x}_n\}$ with probability
$\alpha_1 = \min\left\{1, \frac{w_n^{(I_n)} h(\mathbf{x}_n|\theta^{(I')})}{w_n^{(I')} h(\mathbf{x}_n|\theta^{(I_n)})}\right\}$.

(b) With probability 0.5, update $\mathbf{x}_n$ at $\theta^{(I_n)}$:

Propose $\mathbf{x}'$ by using m-MH updates from $h(\cdot|\theta^{(I_n)})/Z(\theta^{(I_n)})$: $\mathbf{x}' \sim T^m(\cdot|\mathbf{x}_n)$.

Accept $\{I_{n+1}, \mathbf{x}_{n+1}\} = \{I_n, \mathbf{x}'\}$ with probability
$\alpha_2 = \min\left\{1, \frac{h(\mathbf{x}'|\theta^{(I_n)}) T(\mathbf{x}_n|\mathbf{x}')}{h(\mathbf{x}_n|\theta^{(I_n)}) T(\mathbf{x}'|\mathbf{x}_n)}\right\}$.

2. Update approximation of $Z(\theta^{(i)})$ at $i = 1, ..., d$:

(a) $\log(w_{n+1/2}^{(i)}) = \log(w_n^{(i)}) + a_{n+1}(\ 1\{I_{n+1} = i\} - 1/d\ ), \quad a_n = n_0/\max(n_0, n)$.

(b) Stochastic truncation (T) is implemented if $\mathbf{w}_{n+1/2}$ is outside of the target region (K_{σ_n}), and records the number of truncation (σ_n):

$$\{\mathbf{w}_{n+1}, \mathbf{x}_{n+1}\} = \begin{cases} \mathrm{T}(\{\mathbf{w}_n, \mathbf{x}_n\}),\ \sigma_{n+1} = \sigma_n + 1 & \mathbf{w}_{n+1/2} \notin K_{\sigma_n} \\ \{\mathbf{w}_{n+1/2}, \mathbf{x}_{n+1}\},\ \sigma_{n+1} = \sigma_n & o.w. \end{cases}$$

3. Cumulative information is updated: $H_{n+1} = H_n \cup \{I_{n+1}, \mathbf{x}_{n+1}, w_{n+1}^{(I_{n+1})}\}$.

Target Chain: Exchange algorithm with resampling y

4. Propose θ': $\theta' \sim q(\cdot|\theta_n)$.

5. Sample the auxiliary variable from collected dataset in auxiliary chain:

$\mathbf{y} \sim \{\mathbf{x}_1, ..., \mathbf{x}_{n+1}\}$, with resampling probabilities as
$P(\mathbf{y} = \mathbf{x}_l|\theta') \propto \sum_{j=1}^{|H_{n+1}|} w_j^{(I_j)} \frac{h(\mathbf{x}_j|\theta')}{h(\mathbf{x}_j|\theta^{(I_j)})} 1\{\mathbf{x}_j = \mathbf{x}_l\}, \quad l = 1, ..., |H_{n+1}|$.

6. Accept $\theta_{n+1} = \theta'$ with probability
$\alpha_3 = \min\left\{1, \frac{p(\theta')h(\mathbf{x}|\theta')h(\mathbf{y}|\theta_n)q(\theta_n|\theta')}{p(\theta_n)h(\mathbf{x}|\theta_n)h(\mathbf{y}|\theta')q(\theta'|\theta_n)}\right\}$, else reject (set $\{\theta_{n+1}, \mathbf{y}_{n+1}\} = \{\theta_n, \mathbf{y}_n\}$).

Although we classify AEX as an auxiliary variable approach, AEX also shares characteristics with likelihood approximation approaches. The abundance factor $\{w_j^{(i)}\}$ converges to $\{Z(\theta^{(i)})\}$ for each particle, which is guaranteed by SAMC. This implies that likelihood approximation approaches are applied to estimate $Z(\theta^{(i)})$ directly. Then the auxiliary variable is resampled to cancel out $Z(\theta)$ in the target chain. A strength of AEX is its broad applicability, while still remaining asymptotically exact. It is reasonably efficient computationally. However because of its adaptive structure, there can be serious memory issues. When there are low-dimensional sufficient statistics, as is typically the case with exponential family models, only the sufficient statistics of $\mathbf{x}_{n+1}$ need to be stored in Step 3 of the Algorithm 5. However, in the absence of such sufficient statistics, $\mathbf{x}_{n+1}$ itself need to be stored at each step. Furthermore, without sufficient statistics, resampling probability calculations in Step 5 becomes expensive because $h(\mathbf{x}_j|\theta')$ should be recalculated; when there are sufficient statistics, one can simply take the product of θ' and the sufficient statistic.

Components to be tuned: We provide details about choosing particles (number of particles, strategies for choosing particles) in the supplementary material. In the auxiliary chain, the number of neighbors for updating $\theta^{(I_n)}$, the number of preliminary runs for the auxiliary chain (N_1), the number of MH updates (m), gain factor component (n_0), and the target region (K_s) of abundance factor should be tuned. For particles, neighbor is defined using the distance from each other. We can define the closest 20 points as neighboring particles, and this choice appears to work well both in simulated examples in this manuscript as well as the examples in Liang et al. (2016). For updating $\mathbf{x}_n$, $m = 1$ cycle of MH updates are enough, because we don't need to generate independent samples. For complicated problems Liang et al. (2016) recommend larger values for N_1 and n_0. For the simulation examples, Liang et al.

(2016) use around $N_1 = 10,000d$, $n_0 = 25,000$ where d is the number of particles. We used similar values in our simulated examples. If the preliminary auxiliary chain does not appear to converge, larger n_0 and N_1 should be used. The convergence of the auxiliary chain can be checked by comparing sampling frequencies $v(i)/N_1$ with the target probabilities $1/d$, where $v(i)$ is the number of visitations at each particle. If $v(i)/N_1 \approx 1/d$ for each particle, then the preliminary run of auxiliary chain can be diagnosed as having converged. For the choice of compact subset of W, Liang et al. (2016) set $K_s = [0, 10^{100+10s}]^d$ and $\mathcal{X}_0 = \mathcal{X}$ in their examples. Detailed simulation settings about the social network model also can be checked in Jin et al. (2013).

Theoretical justification: The AEX is an asymptotically exact algorithm and the ergodic average satisfies the Weak Law of Large Numbers. Since AEX is adaptive in both proposal and target, conventional theory for adaptive MCMC does not directly apply. Liang et al. (2016) extend a result in Roberts and Rosenthal (2007)[Theorem 1], which shows the ergodicity of the adaptive chain with the same target distribution. Although the original proof (Liang et al., 2016) assumes compactness of the parameter space Θ, Jin et al. (2013) extend the results without compactness assumption for Θ. The details of the assumptions for the proof(s) are provided across several results in Jin et al. (2013)[Lemma 3.1, 3.2 and Theorem 3.1]. In order to make it easy for readers to understand the assumptions from a practical point of view, we have attempted to distill the key assumptions as follows: (1) both $\mathcal{X}$ and W are compact, (2) $h(\mathbf{x}|\theta)$ is bounded away from 0 and ∞, (3) $\lim_{n\to\infty} a_n = 0$, $\sum_{n=1}^{\infty} a_n = \infty$, $\sum_{n=1}^{\infty} a_n^{\eta} < \infty$ for some $\eta \in (1, 2]$, (4) Doeblin condition holds for kernel of auxiliary chain. Some of these conditions are easily verified in practice. The sample space $\mathcal{X}$ for data is compact for finite lattice problem and point process with finite number of points on bounded domain. Sample space W for

abundance factor is also compact, if we set K_s as suggested in the components to be tuned above. The second assumption is also satisfied for realistic parameter settings. The third is a technical assumption for SAMC, and is satisfied if we construct $a_n = n_0/\max(n_0, n)$. The sufficient condition for the last assumption is local positiveness of m-MH updates in Step 1(b) of Algorithm 5, which means $\exists\ \delta > 0, \epsilon > 0$ such that, $\forall \mathbf{x} \in \mathcal{X}, |\mathbf{x} - \mathbf{y}| \leq \delta$ implies $T(\mathbf{y}|\mathbf{x}) \geq \epsilon$. This is satisfied for a simple Gibbs sampler, birth-death sampler in point process, and tie-no-tie sampler in social network models. Therefore, broadly speaking, the assumptions hold in most problems. We point the readers to the manuscript in order to find the complete set of assumptions for these results.

3 likelihood approximation approaches

While the auxiliary variable approaches we have discussed so far avoid approximating $Z(\theta)$, the likelihood approximation approaches we discuss here directly approximate $Z(\theta)$ through Monte Carlo and the approximated normalizing functions are substituted into the acceptance probability of MCMC.

3.1 Atchade, Lartillot and Robert's Adaptive MCMC

Atchade et al. (2008) provide an adaptive MCMC algorithm which approximates $Z(\theta)$ through importance sampling in the acceptance probability. We will henceforth refer to this algorithm as the ALR (Atchade, Lartillot and Robert) algorithm. ALR updates non-Markovian stochastic process and approximates $Z(\theta)$ adaptively using the entire path of the updated process with each iteration. With increasing iterations, $Z(\theta)$ is approximated using larger number of samples; this can lead to more accurate approximation. ALR is an asymptotically exact algorithm that does not require

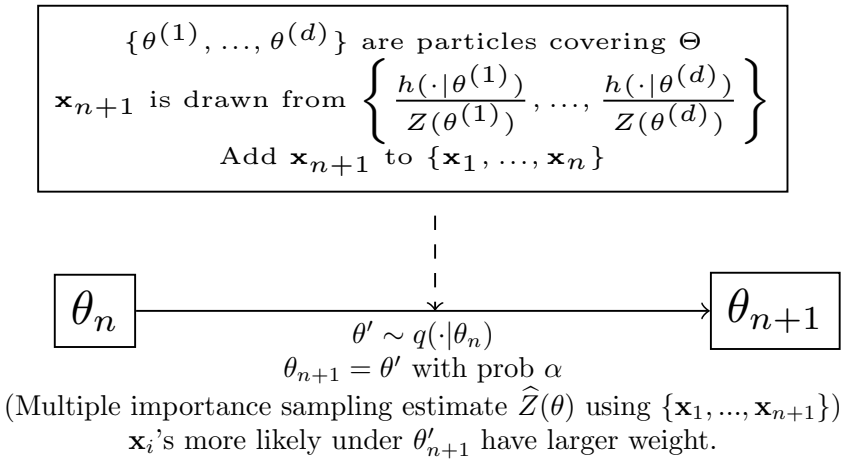

Figure 5: Illustration for Atchade, Lartillot and Robert's algorithm.

perfect sampling.

The ALR algorithm is based on ideas developed in the MCMC-MLE (Geyer and Thompson, 1992). Consider $\{\mathbf{x}_1, ..., \mathbf{x}_n\}$, samples from $h(\mathbf{x}|\theta^{(0)})/Z(\theta^{(0)})$ for some $\theta^{(0)}$. Then a Monte Carlo approximation of $Z(\theta)/Z(\theta^{(0)})$ can be obtained by importance sampling. However the approximation might be poor if θ is far from $\theta^{(0)}$. Atchade et al. (2008) extend the idea of MCMC-MLE by introducing multiple particles, $\{\theta^{(1)}, ..., \theta^{(d)}\}$. A linear combination of importance sampling estimates for $\{Z(\theta)/Z(\theta^{(1)}), ..., Z(\theta)/Z(\theta^{(d)})\}$ approximates $Z(\theta)$. This is called a multiple importance sampling estimate. This idea of sampling from mixture distribution is connected to the umbrella sampling (Torrie and Valleau, 1977). Larger weights are assigned to the estimate of $Z(\theta)/Z(\theta^{(i)})$ when θ and $\theta^{(i)}$ are closer to each other. This is a more robust approximation than a single point importance sampling estimate, as long as one of the $\theta^{(i)}$ is close to θ. Here we provide a sketch of ALR as follows. We also summarize the algorithm in Figure 5.

- For nth iteration of ALR, we describe notations as follows.
  - Particles: $\{\theta^{(1)}, ..., \theta^{(d)}\}$, each $\theta^{(i)} \in \Theta$. These particles are fixed through the algorithm.
  - Particle index: $I_n \in \{1, ..., d\}$ returns index of a chosen particle at nth iteration.
  - Auxiliary data: $\mathbf{x}_n \in \mathcal{X}$ is an approximate sample from probability model at selected particle $\theta^{(I_n)}$. i.e. $\mathbf{x}_n \sim h(\cdot|\theta^{(I_n)})/Z(\theta^{(I_n)})$.
  - Normalizing function approximation at each particle: $\mathbf{c}_n = \{c_n^{(1)}, ..., c_n^{(d)}\} \in R^d$. For $i = 1, ..., d$, as n gets large $c_n^{(i)}$ converges to $\log Z(\theta^{(i)})$ by stochastic approximation (Wang and Landau, 2001; Atchade and Liu, 2004).
  - Cumulative information: $H_n = \cup_{j=1}^{n}\{I_j, \mathbf{x}_j\}$ is necessary for constructing the algorithm. Therefore, $\{I_n, \mathbf{x}_n\}$ should be stored at each iteration.
  - Posterior sample: $\theta_n \in \Theta$

- **Sketch of ALR algorithm**: Part 2 is the entire algorithm. Part 1 is preliminary steps finding initial values for Part 2; this saves computational costs.

  Part 1: Construct reasonable approximation of $\{\log Z(\theta^{(i)})\}$.
  Updates are repeated until $c_{n+1}^{(i)}$ converges to $\{\log Z(\theta^{(i)})\}$ for $i = 1, ..., d$. For $(n+1)$st iteration,
  - $\mathbf{x}_{n+1}$: Using $\mathbf{x}_n$ as starting value, construct MH updates whose stationary distribution is $h(\mathbf{x}|\theta^{(I_{n+1})})/Z(\theta^{(I_{n+1})})$. $\mathbf{x}_{n+1}$ is result of this updates.
  - I_{n+1}: Sampled among $\{1, ..., d\}$. Sampling probabilities are calculated using $\mathbf{x}_{n+1}$ and $\mathbf{c}_n$. I_{n+1} is the index of the selected particle.
  - $\mathbf{c}_{n+1}$: Obtain from $\mathbf{x}_{n+1}$ and $\mathbf{c}_n$ using stochastic approximation (Wang and Landau, 2001; Atchade and Liu, 2004).

Return $\{\mathbf{x}_\tau, I_\tau, \mathbf{c}_\tau\}$, where τ is the total number of updates in Part 1.

Part 2: Update the entire process.

Using $\{\mathbf{x}_\tau, I_\tau, \mathbf{c}_\tau\}$ in Part 1 as initial $\{\mathbf{x}_0, I_0, \mathbf{c}_0\}$, for $(n+1)$st iteration, the entire process $\{\mathbf{x}_{n+1}, I_{n+1}, \mathbf{c}_{n+1}, \theta_{n+1}\}$ is updated as follows.

- Update $\{\mathbf{x}_{n+1}, I_{n+1}, \mathbf{c}_{n+1}\}$ as in Part 1.
- H_{n+1}: Cumulative information is updated. i.e. $H_{n+1} = H_n \cup \{I_{n+1}, \mathbf{x}_{n+1}\}$.
- θ_{n+1}: sampled from $\pi(\theta|\mathbf{x})$ through regular MCMC step; $Z(\theta)$ is approximated using H_{n+1} and $\mathbf{c}_{n+1}$ as in (10) and plugged into the acceptance probability.

By extending a result in Wang and Landau (2001); Atchade and Liu (2004), the ALR algorithm updates a non-Markovian stochastic process $\{\mathbf{x}_{n+1}, I_{n+1}, \mathbf{c}_{n+1}, \theta_{n+1}\} \in \mathcal{X} \times \{1, ..., d\} \times R^d \times \Theta$ in the $(n+1)$st iteration. See Algorithm 6 for details about updating. Then $Z(\theta)$ is approximated through multiple importance sampling and plugged into the acceptance probability to generate posterior samples from $\pi(\theta|\mathbf{x})$. The decomposition of $Z(\theta)$ is

$$\begin{aligned} Z(\theta) &= \int_{\mathcal{X}} h(\mathbf{x}|\theta) d\mathbf{x} \\ &= \sum_{i=1}^{d} k(\theta, \theta^{(i)}) \int_{\mathcal{X}} \frac{h(\mathbf{x}|\theta)}{h(\mathbf{x}|\theta^{(i)})} h(\mathbf{x}|\theta^{(i)}) d\mathbf{x} \\ &= \sum_{i=1}^{d} k(\theta, \theta^{(i)}) Z(\theta^{(i)}) \int_{\mathcal{X}} \frac{h(\mathbf{x}|\theta)}{h(\mathbf{x}|\theta^{(i)})} \frac{h(\mathbf{x}|\theta^{(i)})}{Z(\theta^{(i)})} d\mathbf{x}. \end{aligned} \tag{9}$$

This suggests following approximation of $Z(\theta)$,

$$\widehat{Z}_{n+1}(\theta) = \sum_{i=1}^{d} k(\theta, \theta^{(i)}) \exp(c_{n+1}^{(i)}) \frac{\sum_{k=1}^{n+1} \frac{h(\mathbf{x}_k|\theta)}{h(\mathbf{x}_k|\theta^{(i)})} 1\{I_k = i\}}{\sum_{k=1}^{n+1} 1\{I_k = i\}}. \tag{10}$$

In (10), $Z(\theta)$ is approximated with the weighted sum of the importance sampling estimate $Z(\theta^{(i)})$ at each particle. A "similarity kernel" $k(\theta, \theta^{(i)})$ measures the distance between θ and $\theta^{(i)}$ and assigns larger weights to particles closer to θ. Similar to AEX, particles are pre-specified before starting the algorithm and are chosen so that they can cover parameter space. In the examples in Atchade et al. (2008), particles are collected from the stochastic approximation (Younes, 1988). Also, particles can be chosen as in the AEX (Algorithm 9). Algorithm 6 and Algorithm 7 are detailed descriptions of ALR.

We can skip Algorithm 6 (Part 1) and update the entire process $\{\mathbf{x}_n, I_n, \mathbf{c}_n, \theta_n\}$ with arbitrary initial point $\{\mathbf{x}_0, I_0, \mathbf{c}_0, \theta_0\}$, by only using Algorithm 7 (Part 2). However in practice, only $\{\mathbf{x}_n, I_n, \mathbf{c}_n\}$ can be updated first until $\mathbf{c}_n$ converges using Algorithm 6, and then the entire process $\{\mathbf{x}_n, I_n, \mathbf{c}_n, \theta_n\}$ can be updated through Algorithm 7 using the last value of updated process $\{\mathbf{x}_\tau, I_\tau, \mathbf{c}_\tau\}$ in Algorithm 6 as an initial value of Algorithm 7. This can reduce memory costs as well as computational costs. To update θ_n, we need to calculate the approximation $\widehat{Z}_{n+1}(\theta)$ as in (10), which requires cumulative information H_{n+1}. However it is obvious that if $\mathbf{c}_{n+1}$ does not converge well, $\widehat{Z}_{n+1}(\theta)$ will also be a poor approximation, which can lead to the unreliable generation of posterior samples of θ_{n+1}. Considering that usually it takes lots of updates for the convergence of $\mathbf{c}_{n+1}$, we have to store large H_{n+1} to obtain reasonable posterior samples which is a waste of memory. On the other hand, for updating $\{\mathbf{x}_n, I_n, \mathbf{c}_n\}$ in Algorithm 6, not only we don't need to store previous samples H_{n+1} but also calculation of $\widehat{Z}_{n+1}(\theta)$ is not necessary. Therefore, updating the entire process $\{\mathbf{x}_n, I_n, \mathbf{c}_n, \theta_n\}$ after the convergence of $\mathbf{c}_n$ is guaranteed through Algorithm 6 can reduce both memory and computational costs extensively.

Algorithm 6 ALR algorithm (Part 1): Construct approximation of $\{\log Z(\theta^{(i)})\}$.

Given $\{\mathbf{x}_n, I_n, \mathbf{c}_n\} \in \mathcal{X} \times \{1, ..., d\} \times R^d$ at nth iteration.

Algorithm is repeated until each particle has been visited equally:

while $\gamma_{n+1} > \epsilon_1$ **do**

Reset $\mathbf{v} = \mathbf{0} \in R^d$, $v(i)$ counts the number of visitations at each particle $\theta^{(i)}$.

while $\max_i |v(i) - \bar{v}| > \epsilon_2 \bar{v}$ **do**

For $i = 1, ..., d$, set $v(i) = v(i) + 1_i(I_{n+1})$.

1. Update $\mathbf{x}_{n+1}$ through m-MH step for given particle:
   $\mathbf{x}_{n+1} \sim T_{I_n}^m(\cdot|\mathbf{x}_n)$ whose stationary distribution is $\frac{h(\mathbf{x}|\theta^{(I_n)})}{Z(\theta^{(I_n)})}$.
2. Decide where to visit in the next iteration:
   $I_{n+1} \sim \{1, ..., d\}$ with probabilities $h(\mathbf{x}_{n+1}|\theta^{(i)}) \exp(-c_n^{(i)})$ for $i = 1, ..., d$.
3. Update approximation of $\log Z(\theta^{(i)})$:
   $c_{n+1}^{(i)} = c_n^{(i)} + \gamma_n \frac{h(\mathbf{x}_{n+1}|\theta^{(i)}) \exp(-c_n^{(i)})}{\sum_{j=1}^d h(\mathbf{x}_{n+1}|\theta^{(j)}) \exp(-c_n^{(j)})}$ for $i = 1, ..., d$, and $\gamma_{n+1} = \gamma_n$.

end while

Step size become smaller: $\gamma_{n+1} = \gamma_n / 2$

end while

Return $\{\mathbf{x}_\tau, I_\tau, \mathbf{c}_\tau\}$, where τ is the total number of iterations for Algorithm 6.

Algorithm 7 ALR algorithm (Part 2): Update the entire process.

Use $\{\mathbf{x}_\tau, I_\tau, \mathbf{c}_\tau\}$ in Algorithm 6 as an initial value $\{\mathbf{x}_0, I_0, \mathbf{c}_0\}$ for Algorithm 7.

Update $\{\mathbf{x}_n, I_n, \mathbf{c}_n, \theta_n\} \in \mathcal{X} \times \{1, ..., d\} \times R^d \times \Theta$ as follows.

Update $\{\mathbf{x}_{n+1}, I_{n+1}, \mathbf{c}_{n+1}\}$ using Step 1-3 in the Algorithm 6 with deterministic step size, $\gamma_{n+1} = \epsilon_1/(n+1)^{0.7}$. Then append dataset: $H_{n+1} = H_n \cup \{\mathbf{x}_{n+1}, I_{n+1}\}$.

4. Approximate $Z(\theta)$ adaptively using all previous samples (H_{n+1}) and $\mathbf{c}_{n+1}$:
   $\widehat{Z}_{n+1}(\theta) = \sum_{i=1}^{d} k(\theta, \theta^{(i)}) \exp(c_{n+1}^{(i)}) \frac{\sum_{k=1}^{n+1} \frac{h(\mathbf{x}_k|\theta)}{h(\mathbf{x}_k|\theta^{(i)})} 1\{I_k=i\}}{\sum_{k=1}^{n+1} 1\{I_k=i\}}$.
5. Propose $\theta' \sim q(\cdot|\theta)$ and accept $\theta_{n+1} = \theta'$ with probability
   $\alpha = \min\left\{1, \frac{p(\theta')h(\mathbf{x}|\theta')\widehat{Z}_{n+1}(\theta_n)q(\theta_n|\theta')}{p(\theta_n)h(\mathbf{x}|\theta_n)\widehat{Z}_{n+1}(\theta')q(\theta'|\theta_n)}\right\}$, else reject (set $\{\theta_{n+1}, \mathbf{y}_{n+1}\} = \{\theta_n, \mathbf{y}_n\}$).

The purpose of Part 1 of the Algorithm 6 is to obtain a reasonable normalizing function estimate, $\mathbf{c}_n$. The step size $\{\gamma_n\}$ plays an important role in the convergence of $\mathbf{c}_n$ (similar to gain factor a_n is step size for abundance factor $\mathbf{w}_n$ in AEX). If γ_n is too small, it takes a long time for convergence. If γ_n is too large it can lead to large variance of $\mathbf{c}_n$. Therefore in the beginning of the algorithm, γ_n is set to be a large value so that it can move around the space fast; it slowly becomes smaller after moving around the state space. In Algorithm 6, number of visitations at each particle is recorded through $\mathbf{v}$. If $\mathbf{v}$ is close to uniform distribution ($\{I_n\}$ has visited $\{1, ..., d\}$ about equally), γ_n becomes smaller. If step size γ_n is smaller than convergence criteria ϵ_1, $\mathbf{c}_n$ can be diagnosed as having converged.

The strengths and weakness of ALR are similar to those of AEX. A major advantage of ALR is that it is an asymptotically exact sampler without requiring perfect sampling. However there can be serious memory issues even with the practical implementation of algorithm. Without low-dimensional sufficient statistics, the entire

$\mathbf{x}_n$ needs to be stored with each iteration to calculate $\widehat{Z}_{n+1}(\theta)$ in Step 4 of the Algorithm 7. Furthermore, without sufficient statistics, calculations in Step 3 and Step 4 become expensive, because $h(\mathbf{x}_k|\theta), h(\mathbf{x}_k|\theta^{(i)})$ need to be recalculated; with sufficient statistics, one can simply take the product of θ or $\theta^{(i)}$ and the sufficient statistic of $\mathbf{x}_k$.

Components to be tuned: In this algorithm, step size (γ_n), the convergence checking components (ϵ_1, ϵ_2), number of MH updates (m), number of particles (d) and kernel $k(\theta, \theta^{(i)})$ need to be tuned. Atchade et al. (2008) set initial γ_0 as 1, $\epsilon_1 = 0.001, \epsilon_2 = 0.2$. Under these settings, consider the sequence of bounded stopping times $0 = \tau_0 < \tau_1 < ... < \tau_{10} = \tau$, where τ is the total number of iteration in Algorithm 6. Until τ_1, initial γ_0 is used as step size. For given this step size, τ_1 is stopping time until each particle has been visited equally (i.e. $\max_{i=1,...,d} |v(i) - \bar{v}| \leq \epsilon_2 \bar{v}$). Then γ_{τ_1+1} becomes $\gamma_0/2 = 1/2$ and is kept until τ_2. This is repeated until τ_{10} where $\gamma_{\tau_{10}+1} = 1/2^{10} < 0.001 = \epsilon_1$. Likewise, step size is controlled to guarantee successful convergence of $\mathbf{c}_n$. Once $\mathbf{c}_n$ has converged well, Algorithm 7 is implemented which updates the entire process. γ_{n+1} is updated as the deterministic sequence $0.001/(n+1)^{0.7}$ in Algorithm 7. Similar to AEX, $m = 1$ cycle of MH updates are enough in practice. For the number of particles, $d = 100p$ appears to work well in practice to cover the p-dimensional parameter space. Although various choice of kernel $k(\theta, \theta^{(i)})$ is possible, uniform kernel with bandwidth h is used in this manuscript. This kernel gives $1/h$ weights for the h closest particles and gives 0 weights for others. Bandwidth should be determined by trials and errors and we used $h = 20$.

Theoretical justification: The ALR is an asymptotically exact algorithm and the ergodic average from the chain satisfies the Strong Law of Large Numbers. Atchade et al. (2008) provide a proof that $\{\theta_n\}$ from the generated process con-

verges to the target distribution exactly. Three assumptions are required: (1) $h(\mathbf{x}|\theta)$ is bounded away from 0 and ∞, (2) $q^{n_0}(\theta'|\theta) \geq \epsilon$ for all $\theta, \theta' \in \Theta$ where q is proposal density and $n_0 \geq 1$ is an integer, (3) $\{\gamma_n\}$ is random and adaptively updated based on the previous generated process, which satisfies $\sum \gamma_n = \infty$ and $\sum \gamma_n^2 < \infty$ almost surely. The first assumption holds for finite $\mathcal{X}$ and realistic parameter settings. The second assumption also holds for most symmetric kernels. The third assumption is technical, and taken from Wang and Landau (2001). Because we can typically control the sequence $\{\gamma_n\}$, this assumption is also generally easy to satisfy. Therefore, the assumptions hold in most cases.

3.2 Pseudo-marginal MCMC

When the likelihood function $L(\theta|\mathbf{x})$ is intractable, we can substitute the Monte Carlo approximation $\widehat{L}(\theta|\mathbf{x})$ in the acceptance probability of MCMC as an alternative. If $\widehat{L}(\theta|\mathbf{x})$ is positive and an unbiased estimate for $L(\theta|\mathbf{x})$, the stationary distribution of the Markov chain is exactly equal to the desired target posterior. This is called pseudo-marginal MCMC (Beaumont, 2003; Andrieu and Roberts, 2009).

The pseudo-marginal framework requires an unbiased approximation for $1/Z(\theta)$. An unbiased approximation of $Z(\theta)$ can be obtained using importance sampling estimate $\widehat{Z}(\theta)$ via MCMC samples from the likelihood. However, $1/\widehat{Z}(\theta)$ is a consistent but biased approximation for $1/Z(\theta)$. Lyne et al. (2015) address this bias through geometric series correction. Let $\widetilde{Z}(\theta)$ be the upper bound of $Z(\theta)$ or any estimate of $Z(\theta)$. Then likelihood is represented through multiplication with infinite geometric series as

$$L(\theta|\mathbf{x}) = \frac{h(\mathbf{x}|\theta)}{\widetilde{Z}(\theta)} c(\theta) \left[1 + \sum_{n=1}^{\infty} k(\theta)^n \right], \quad k(\theta) = 1 - c(\theta) \frac{Z(\theta)}{\widetilde{Z}(\theta)}, \tag{11}$$

where $c(\theta)$ controls $|k(\theta)| < 1$. Therefore with an infinite supply of independent

unbiased estimates $\widehat{Z}_i(\theta)$ for $Z(\theta)$, unbiased approximation of the likelihood can be constructed as

$$\widehat{L}(\theta|\mathbf{x}) = \frac{h(\mathbf{x}|\theta)}{\widetilde{Z}(\theta)} c(\theta) \left[1 + \sum_{n=1}^{\infty} \prod_{i=1}^{n} \widehat{k}_i(\theta)\right], \quad \widehat{k}_i(\theta) = 1 - c(\theta) \frac{\widehat{Z}_i(\theta)}{\widetilde{Z}(\theta)}. \tag{12}$$

Another possible choice is using multiplication correction with a Maclaurin series expansion of the exponential function (Lyne et al., 2015). However both the geometric and Maclaurin series corrections require an infinite number of unbiased estimates $\widehat{Z}_i(\theta)$, which makes them impractical. To address this problem, Lyne et al. (2015) propose stochastic truncation of the infinite series called the Russian Roulette, originally used in physics (Carter and Cashwell, 1975). Using simulated finite random stopping time τ, only a τ number of $\widehat{Z}_i(\theta)$ are required to construct an unbiased likelihood approximation. This can lead to an asymptotically exact algorithm.

Let the random stopping time be $\tau \geq 1$ with $p_n = P(\tau \geq n)$ for all $n \geq 1$, and $p_1 = 1$. The stopping time can be defined as, $\tau = \inf\{m \geq 1 : U_j \geq q_j\}$ where $U_j \sim$ i.i.d. $u(0,1), q_j \in (0,1]$, which makes $p_n = \prod_{j=1}^{n-1} q_j$. Then the partial sum $S_\tau = 1 + \sum_{n=1}^{\tau-1} \prod_{i=1}^{n} \widehat{k}_i(\theta)/p_n$ with $S_1 = 1$ is called the Russian Roulette estimate. Lyne et al. (2015) show that S_τ is the unbiased estimate of infinite series as

$$E[S_\tau] = \sum_{j=1}^{\infty} S_j P(\tau = j) = 1 + \sum_{n=1}^{\infty} \prod_{i=1}^{n} \widehat{k}_i(\theta). \tag{13}$$

This indicates that the infinite series can be represented as the expectation of finite series via the Russian Roulette truncation. In this way, unbiased approximation of the likelihood is achieved as

$$\widehat{L}_\tau(\theta|\mathbf{x}) = \frac{h(\mathbf{x}|\theta)}{\widetilde{Z}(\theta)} c(\theta) S_\tau, \quad S_\tau = \left[1 + \sum_{n=1}^{\tau-1} \frac{\prod_{i=1}^{n} \widehat{k}_i(\theta)}{\prod_{j=1}^{n-1} q_j}\right], \tag{14}$$

which makes expectation of $\widehat{L}_\tau(\theta|\mathbf{x})$ is equal to (12). $\widehat{L}_\tau(\theta|\mathbf{x})$ will be plugged

into the acceptance probability of the Metropolis-Hastings algorithm as in Figure 6. This finite series truncation makes the algorithm practical, since the finite τ number of $\widehat{Z}_i(\theta)$ is used to construct unbiased approximation. The choice of q_j directly has an impact on both variance of likelihood approximation and the computing speed of the algorithm. Smaller q_j can lead to smaller τ which is the required number of the $\widehat{Z}_i(\theta)$ for constructing $\widehat{L}_\tau(\theta|\mathbf{x})$; smaller τ can lead to faster run of the algorithm. However Lyne et al. (2015) provide a proof that if q_j is close to zero, variance of likelihood approximation become large or even infinite. Lyne et al. (2015) suggest that $q_j = \prod_{i=1}^{j} \widehat{k}_i(\theta)$ appears to work well in practice.

Algorithm 8 summarizes the Russian Roulette algorithm. Although the Russian Roulette algorithm is an asymptotically exact MCMC and applicable for general forms of $h(\mathbf{x}|\theta)$, the algorithm is computationally expensive. It requires multiple supplies (τ number) of $\widehat{Z}_i(\theta)$ with each iteration (Step 3(b)). Each $\widehat{Z}_i(\theta)$ approximation requires multiple MCMC samples from the likelihood, making it computationally expensive.

Components to be tuned: When a geometric correction is used for the Russian Roulette algorithm, five components of the algorithm need to be tuned: raw estimate $\widetilde{Z}(\theta)$, $c(\theta)$ which controls $|k(\theta)| < 1$, (N) the number of samples for constructing the importance sampling estimate at each iteration, (m) length of inner sampler to generate MCMC samples, and importance function $h(\mathbf{x}|\theta^{(0)})/Z(\theta^{(0)})$. If a tight upper bound of $Z(\theta)$ can be derived, it is a computationally efficient choice for $\widetilde{Z}(\theta)$, because it can lead to small $\widehat{k}_i(\theta)$ and small q_j, which implies small τ. However, in most situation it is difficult to get a tight upper bound of $Z(\theta)$. Therefore, importance sampling estimate can be used for $\widetilde{Z}(\theta)$. $c(\theta)$ and N should be determined through trials and errors to make $|\widehat{k}_i(\theta)| < 1$. In our simulation examples, we set

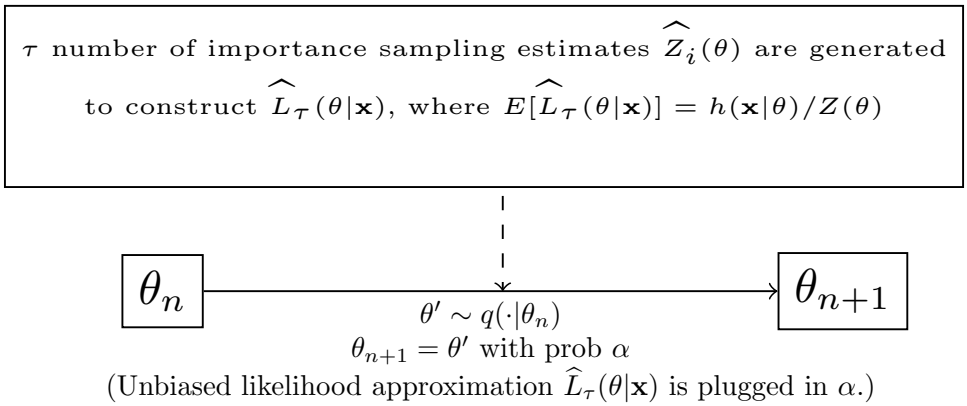

Figure 6: Illustration for the Russian Roulette algorithm.

Algorithm 8 Russian Roulette algorithm

Given $\theta_n \in \Theta$ at nth iteration.

1. Propose $\theta' \sim q(\cdot|\theta_n)$.

2. Calculate raw estimate $\widetilde{Z}(\theta')$ (e.g. importance sampling estimate).

3. Construct unbiased likelihood estimate: start with $i = j = 0$, $U_0 = 0, q_0 = 1$.

while $q_j > U_j$ **do**

(a) $i = i + 1$, $j = j + 1$, and $U_j \sim$ i.i.d. $u(0, 1)$.

(b) Construct a $\widehat{Z}_i(\theta')$ via importance sampling:

$\widehat{Z}_i(\theta') = \frac{1}{N}\sum_{i=1}^{N} h(\mathbf{x}|\theta)/h(\mathbf{x}|\theta^{(0)})$.

$\{\mathbf{x}_1, ..., \mathbf{x}_N\} \sim T^m_{\theta^{(0)}}(\cdot|\mathbf{x})$ whose stationary distribution is $h(\mathbf{x}|\theta^{(0)})/Z(\theta^{(0)})$, which is the importance function.

(c) Calculate $q_j = \prod_{i=1}^{j} \widehat{k}_i(\theta')$, where $\widehat{k}_i(\theta') = 1 - c(\theta')\widehat{Z}_i(\theta')/\widetilde{Z}(\theta')$.

end while

Set $\tau' = j$ and $\widehat{L}_{\tau'}(\theta'|\mathbf{x}) = \frac{h(\mathbf{x}|\theta')}{\widetilde{Z}(\theta')} c(\theta') \left[1 + \sum_{n=1}^{\tau'-1} \frac{\prod_{i=1}^{n} \widehat{k}_i(\theta')}{\prod_{j=1}^{n-1} q_j}\right]$.

4. Accept $\theta_{n+1} = \theta'$ with probability

$\alpha = \min\left\{1, \frac{p(\theta')\widehat{L}_{\tau'}(\theta'|\mathbf{x})q(\theta_n|\theta')}{p(\theta_n)\widehat{L}_\tau(\theta_n|\mathbf{x})q(\theta'|\theta_n)}\right\}$, else reject (set $\{\theta_{n+1}, \mathbf{y}_{n+1}\} = \{\theta_n, \mathbf{y}_n\}$).

N between 1000 and 2000. $c(\theta)$=0.4 is used for every examples. Length of inner sampler m should be larger as there is strong dependence among data to generate more reliable samples. A simple choice for importance function is using maximum pseudolikelihood estimate as $\theta^{(0)}$. The other types of importance sampling estimate (e.g. annealed importance sampling estimate) can also be used to construct $\widehat{Z}_i(\theta)$. To reduce computing time, we can construct independent $\widehat{Z}_i(\theta)$ in parallel.

Theoretical justification: The Russian Roulette algorithm is based on the pseudo-marginal framework. If $h(\mathbf{x}|\theta)$ is bounded, we can attain positive unbiased approximation of the likelihood and implement algorithm without any theoretical difficulties. Even when $h(\mathbf{x}|\theta)$ is unbounded, $|\widehat{L}_\tau(\theta|\mathbf{x})|$ can be substituted instead of $\widehat{L}_\tau(\theta|\mathbf{x})$ in the Algorithm 8 and weighted average of $f(\theta_i)$ with sign of $\widehat{L}_\tau(\theta|\mathbf{x})$ is used to estimate $E[f(\theta)]$ without losing exactness. See "sign problem" in Lyne et al. (2015) for details. Therefore, this algorithm is generally applicable.

3.3 Noisy MCMC

If the Markov chain with transition kernel P satisfies detailed balance with respect to the target π, it is called an asymptotically exact MCMC. However, when P is approximated by $\widehat{P}$, the samples generated may only approximately follow the target $\widehat{\pi}$. A general term for such algorithm is noisy MCMC. Although it is asymptotically inexact, the approximated chain can have statistical or computational efficiency. Alquier et al. (2016) describe a broad class of noisy MCMC algorithms, and use total variational distance to quantify the distance between the asymptotically exact and inexact chain.

In the exchange algorithm, a single auxiliary variable $\mathbf{y}$ is generated from $h(\mathbf{y}|\theta')/Z(\theta')$. Therefore the $h(\mathbf{y}|\theta)/h(\mathbf{y}|\theta')$ term in the (4) may be thought of as a one-sample un-

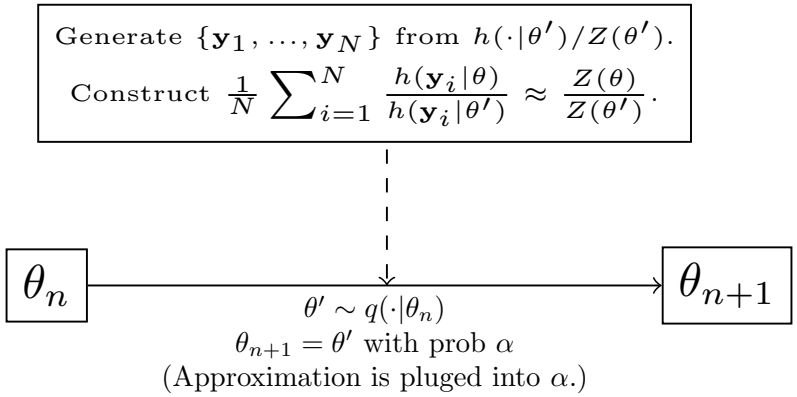

Figure 7: Illustration for the Noisy MCMC. Auxiliary variables can be generated exactly (noisy exchange) or approximately (noisy DMH).

biased importance sampling estimate of $Z(\theta)/Z(\theta')$. Instead of using a single $\mathbf{y}$, if multiple auxiliary variables $\{\mathbf{y}_1, ..., \mathbf{y}_N\}$ are generated from the $h(\mathbf{y}|\theta')/Z(\theta')$ and used to construct importance sampling estimate, this estimate will have smaller variance than before. This is called the noisy exchange algorithm with the acceptance probability

$$\alpha = \min\left\{1, \frac{p(\theta')h(\mathbf{x}|\theta')q(\theta|\theta')}{p(\theta)h(\mathbf{x}|\theta)q(\theta'|\theta)} \frac{1}{N}\sum_{i=1}^{N} \frac{h(\mathbf{y}_i|\theta)}{h(\mathbf{y}_i|\theta')}\right\}. \tag{15}$$

We summarize the algorithm in Figure 7. For the $1 < N < \infty$, the algorithm is asymptotically inexact because the detailed balance condition does not hold. Liang and Jin (2013) also propose similar approach called Monte Carlo MH (MCMH). Alquier et al. (2016) report that noisy exchange algorithm shows better mixing than the exchange algorithm. Also, the estimate for the ergodic average has smaller bias than the exchange algorithm in empirical studies. As trivial extension, we will refer to DMH with generating multiple auxiliary variables as noisy DMH.

Although we do not study details in this manuscript, for completeness we briefly discuss a noisy version of Metropolis adjusted Langevin (MALA) exchange algorithm (Alquier et al., 2016). MALA is constructed based on the Stochastic Differential Equations (SDE) for the Langevin diffusion which describes the dynamics of the process according to the gradient of the target distribution. Noisy MALA exchange algorithm uses such gradient information in the proposal for θ; which is the only difference from the noisy exchange algorithm. Noisy MALA exchange algorithm is two-stage approximation from (1) first order Euler approximation to obtain solution for SDE and (2) Monte Carlo approximation for gradient of the intractable target distribution. Alquier et al. (2016) report that noisy MALA exchange algorithm shows better mixing compared to the standard exchange algorithm.

Components to be tuned: N is an additional component to be tuned compared to the exchange algorithm or DMH. The choice of N totally depends on the computational capabilities for the problems. As N becomes larger, the constructed chain can have better mixing and estimate from the chain can have lower variance, at the expense of computing time. As the Russian Roulette algorithm, we can take advantage of the parallel computing for sampling N number of $\mathbf{y}_i$, and evaluating $h(\mathbf{y}_i|\theta)/h(\mathbf{y}_i|\theta')$ independently.

Theoretical justification: Let P be the transition kernel for the exchange algorithm with target π, and $\widehat{P}$ be the kernel for the noisy exchange algorithm with the approximate target $\widehat{\pi}$. Alquier et al. (2016) provide an upper bound for total variation norm distance between P and $\widehat{P}$ based on a result in Mitrophanov (2005)[Corollary 3.1], which requires uniform ergodicity of P. Two assumptions are used in this derivation: (1) $p(\theta)$ is bounded away from 0 and ∞, (2) $q(\theta'|\theta)$ is bounded

away from 0 and ∞; both assumptions are typically satisfied.

4 Simulated and real data examples

We have reviewed algorithms that fall under two different classes, auxiliary variable methods and likelihood approximation approaches. Here we study the algorithms above in the context of applications to three different examples: (1) Ising model, (2) a social network model, and (3) an attraction-repulsion point process model. Examples are implemented in `R` and `C++`, using the Rcpp and RcppArmadillo packages (Eddelbuettel et al., 2011). We use the examples to compare the efficiency of the algorithms in practice.

4.1 Ising model example

The Ising model (Lenz, 1920; Ising, 1925), a specific version of a MRF, models spatial dependence on a lattice. Consider the m by n lattice of a one-dimensional Ising model. The one-dimensional Ising model is as

$$L(\theta|\mathbf{x}) = \frac{1}{Z(\theta)} \exp\left\{\theta S(\mathbf{x})\right\}, \quad S(\mathbf{x}) = \sum_{i=1}^{m}\sum_{j=1}^{n-1} x_{i,j}x_{i,j+1} + \sum_{i=1}^{m-1}\sum_{j=1}^{n} x_{i,j}x_{i+1,j}. \tag{16}$$

The m by n observed lattice $\mathbf{x}$ can have binary values $x_{i,j} = \{-1, 1\}$, where i, j denotes row and column location in the lattice. The larger the θ becomes, the stronger the interactions between the lattice point. Summation over all 2^{mn} possible configurations of this model is required for the calculation of the normalizing function, which is computationally expensive even for lattice of moderate size. The simulations are conducted on a 10×10 lattice with uniform prior [0,1] with two different parameter settings: (1) $\theta = 0.2$, representing moderate dependence, and (2)

$\theta = 0.43$, corresponding to strong dependence. Data $\mathbf{x}$ is generated through perfect sampling with two different parameter settings.

Each of the algorithm is tuned according to the components to be tuned part in the previous sections. $h(\mathbf{y}|\widehat{\theta})/Z(\widehat{\theta})$ is chosen as the conditional density of the auxiliary variable $\mathbf{y}$ for the AVM algorithm, where $\widehat{\theta}$ is MPLE. The auxiliary variable is generated by 10 cycles of Gibbs updates in DMH. For AEX, 100 particles are selected among 5,000 samples from fractional DMH, as described in Algorithm 9. Then the preliminary run of the auxiliary chain is implemented for 420,000 iterations with the first 20,000 iterations discarded for burn-in. The resulting samples are thinned (at equally spaced intervals of 20) to obtain 20,000 samples. In the auxiliary chain, we set $n_0 = 20,000$, $K_s = [0, 100^{100+10s}]^{100}$, and $\mathbf{x}_n$ is updated by a single cycle of Gibbs updates. For ALR algorithm, 100 particles are selected from uniform prior and $\mathbf{x}_n$ is updated by a single cycle of Gibbs updates. The kernel giving equal weights for $h = 10$ nearest particles and 0 for others is used. Simulation settings in the Russian Roulette algorithm follow Lyne et al. (2015). Annealed importance sampling (AIS) estimate of $Z(\theta)$ is constructed using 1000 samples, and each sample is generated with 2000 temperature from an uniform lattice using the schedule as in (5). Geometric series correction with $c(\theta) = 0.4$ is used to construct unbiased estimate of likelihood. In noisy DMH, 100 samples are used to produce importance sampling estimate and same inner sampler is constructed as DMH. We note that in order to make the computations feasible, we used parallel computing to obtain importance sampling estimates for both noisy DMH and the Russian Roulette algorithms. The parallel computing was implemented through OpenMP with the samples generated in parallel across 8 processors. We treated a run from the exchange algorithm as our gold standard; it was run for 1,010,000 iterations with 10,000 discarded for burn-in and 10,000 thinned samples are obtained from the remaining 1,000,000. Same simu-

$\theta = 0.2$	Mean	95%HPD	ESS	Acc	Time(second)	ESS/Time
AVM	0.20	(0.08,0.32)	1527.88	0.37	12.46	122.62
Exchange	0.20	(0.08,0.33)	2061.23	0.49	15.29	134.81
DMH	0.21	(0.08,0.34)	2068.29	0.48	1.78	1161.96
AEX	0.21	(0.07,0.35)	1778.15	0.47	23.72	74.96
ALR	0.20	(0.08,0.34)	3647.34	0.59	9.33	390.93
RussianR	0.20	(0.06,0.33)	2650.83	0.50	13849.10	0.19
NoisyDMH	0.20	(0.06,0.33)	3347.76	0.59	71.02	47.14
Gold	0.20	(0.08,0.33)	9845.89			
$\theta = 0.43$	Mean	95%HPD	ESS	Acc	Time(second)	ESS/Time
AVM	0.44	(0.35,0.57)	1020.12	0.32	10395.91	0.10
Exchange	0.43	(0.33,0.54)	2146.05	0.40	4889.32	0.44
DMH	0.47	(0.34,0.61)	2029.24	0.48	1.72	1179.79
AEX	0.43	(0.32,0.53)	2048.01	0.41	24.96	82.05
ALR	0.43	(0.32,0.53)	4125.59	0.51	9.88	417.57
RussianR	0.44	(0.33,0.54)	2708.55	0.45	30247.70	0.09
NoisyDMH	0.47	(0.32,0.61)	3486.99	0.62	75.17	46.39
Gold	0.43	(0.33,0.54)	10000.00			

Table 1: Inference results for an Ising model on a 10×10 lattice. 20,000 MCMC samples are generated through all algorithms. The highest posterior density (HPD) is calculated by using coda package in `R`. The calculation of Effective Sample Size (ESS) follows Kass et al. (1998); Robert and Casella (2013). Acc represents acceptance probability.

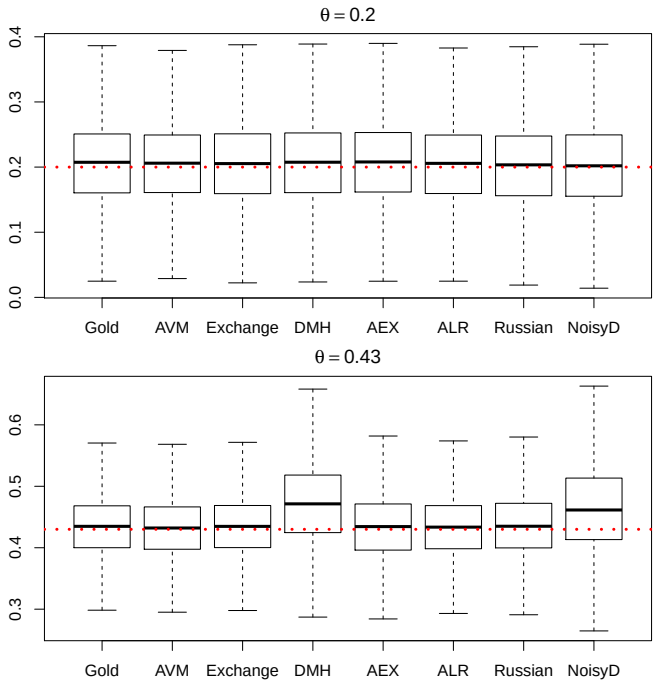

Figure 8: Posterior densities for different θ. Dotted line indicates true value.

lation settings are used for $\theta = 0.43$ case except that 4000 temperatures are used in the Russian Roulette algorithm. Since perfect sampling takes long time for $\theta = 0.43$ case, we use ALR algorithm as the gold standard. All algorithms were run until the Monte Carlo standard error calculated by batch means (Jones et al., 2006; Flegal et al., 2008) is below 0.01.

Table 1 shows that the estimates from different algorithms are well matched to the gold standard when $\theta = 0.2$. Posterior densities in the Figure 8 also indicate that posterior samples agree well. Likelihood approximation approaches show larger ESS than auxiliary variable approaches. On the other hand, auxiliary variable approaches have smaller computational costs than likelihood approximation approaches. In particular, DMH shows the shortest computing time. For this parameter setting, infer-

ence results from both asymptotically exact and inexact algorithms are accurate.

However when $\theta = 0.43$, inference results are biased for asymptotically inexact algorithms. In Table 1 and Figure 8, it is observed that the inference results from DMH and Noisy DMH do not match the gold standard. Due to the strong dependence at this parameter setting, mixing of the inner sampler to generate samples from the $h(\mathbf{x}|\theta)/Z(\theta)$ is slower than for the $\theta = 0.2$ case. Therefore a large number of Gibbs updates are necessary for accurate inference. Although Noisy DMH provides a closer estimate to the true value than DMH because it uses multiple samples for estimating $Z(\theta)/Z(\theta')$, it is still biased. The number of iterations for the inner sampler in the Russian Roulette algorithm also has to be increased due to the slow mixing. We have increased temperature schedules from 2000 to 4000 for $\theta = 0.43$ case, which makes the algorithm more expensive. We can also observe that AVM and the exchange algorithm take several hours, because perfect sampling takes longer to achieve coalescence. On the other hand, there are no extra costs for AEX and ALR in the presence of strong dependence. For both algorithms, still a single cycle of Gibbs update is enough for updating $\mathbf{x}_n$. This is because both algorithms require updating not sampling $\mathbf{x}_n$ from the probability model.

Summary: In the case where the Ising model has only moderate dependence, all the algorithms work reasonably well. Here, DMH may be preferable because it is the easiest to construct and the fastest to run. On the other hand, in the context of Ising models that exhibit strong dependence, the computational problem becomes more challenging. For the asymptotically inexact algorithms like DMH, in order to obtain accurate results in the presence of strong dependence in the Ising model, the number of iterations for the inner sampler needs to be very large. This results in a much larger computational burden. AVM and the exchange algorithms also become

impractical because perfect sampling takes too long. AEX and ALR can be useful for spatial autologistic models because both algorithms can attain accurate estimates with moderate computing speed even in the presence of strong dependence.

4.2 Social network model example

Exponential random graph model (ERGM) (Robins et al., 2007; Hunter et al., 2008) provides an approach for modeling relationships among nodes of a network. Consider the undirected ERGM with n nodes, where the probability model is as

$$L(\theta|\mathbf{x}) = \frac{1}{Z(\theta)} \exp\left\{\theta_1 S_1(\mathbf{x}) + \theta_2 S_2(\mathbf{x}) + \theta_3 S_3(\mathbf{x}) + \theta_4 S_4(\mathbf{x})\right\}, \tag{17}$$

$$S_l(\mathbf{x}) = \sum_{i=1}^{n} \binom{x_{i+}}{l}, l = 1, 2, 3; \qquad S_4(\mathbf{x}) = \sum_{i<j<k} x_{i,j} x_{j,k} x_{k,i}.$$

For all $i \neq j$, $x_{i,j} = 1$ if the ith node and jth node are connected, otherwise $x_{i,j} = 0$ and $x_{i,i}$ is defined as 0. This forms n by n adjacency matrix, $\mathbf{x}$. Sufficient statistics $S(\mathbf{x}) = \{S_1(\mathbf{x}), S_2(\mathbf{x}), S_3(\mathbf{x}), S_4(\mathbf{x})\}$ represents the number of edges, two-stars, three-stars and triangles respectively, where x_{i+} indicates row sum of adjacency matrix. k-star indicates the number of nodes which exactly have k relationships. Triangle represents the number of cyclic relationships. Calculation of the normalizing function requires summation over all $2^{n(n-1)/2}$ configuration, which is infeasible. In this example, we study the Florentine business dataset (Padgett, 1994), which describes the business networks among 16 Florentine families.

We have not implemented AVM and the exchange algorithms because perfect samplers are possible only for a few special cases (Butts, 2012). Each of the algorithm is tuned according to the components to be tuned part in the previous sections. The auxiliary variable is generated by 20 cycles of Gibbs updates in DMH. In AEX,

θ_2	Mean	95%HPD	ESS	Acc	Time(second)	ESS/Time
DMH	1.24	(0.02,2.57)	1026.67	0.24	10.45	98.25
AEX	1.24	(0.17,2.58)	991.87	0.20	126.46	7.84
ALR	1.25	(0.16,2.52)	1456.57	0.33	2500.37	0.58
RussianR	1.27	(0.03,2.68)	1433.60	0.31	33534.96	0.04
NoisyDMH	1.28	(0.03,2.59)	1600.90	0.32	297.35	5.38
Gold	1.27	(0.08,2.50)	9655.90			

Table 2: Inference results for 2-star in ERGM for Florentine business dataset. 30,000 MCMC samples are generated through all algorithms. The highest posterior density (HPD) is calculated by using coda package in `R`. The calculation of Effective Sample Size (ESS) follows Kass et al. (1998); Robert and Casella (2013). Acc represents acceptance probability.

200 particles are selected among 5,000 samples from fractional DMH, as described in Algorithm 9. Then the preliminary run of the auxiliary chain is implemented for 630,000 iterations with the first 30,000 iterations discarded for burn-in. The resulting samples are thinned (at equally spaced intervals of 20) to obtain 30,000 samples. In the auxiliary chain, we set $n_0 = 20,000$, $K_s = [0, 100^{100+10s}]^{200}$, and $\mathbf{x}_n$ is updated by a single cycle of Gibbs updates. For ALR, 400 particles are chosen from the short run of DMH, and a single cycle of Gibbs updates are used to update $\mathbf{x}_n$. Same kernel with $h = 20$ is used as previous example. In the Russian Roulette algorithm, 2000 samples are generated through 20 cycles of Gibbs updates and used for importance sampling estimate whose importance function is $h(\mathbf{x}|\theta^{(0)})/Z(\theta^{(0)})$ where $\theta^{(0)}$ is MPLE. And then, geometric series correction with $c(\theta) = 0.4$ is used to construct unbiased estimate of likelihood. 100 samples are used for importance sampling estimate in noisy DMH. Importance sampling estimate in the Russian Roulette and

noisy DMH are attained through parallel running as in the previous example. To obtain a gold standard for comparisons, we run the AEX algorithm 10 times independently. Each run consists 101,000 iterations with 1,000 iterations discarded as burn-in. Then 10,000 samples are obtained by thinning from the remaining 10 sets of 100,000 samples. All algorithms were run until the Monte Carlo standard error is at or below 0.02.

Here we only provide the inference results regarding θ_2 because, similar results are observed for the other parameters. Table 2 indicates that the estimates from the different algorithms are similar to the those of the gold standard. This is because 20 cycles of Gibbs updates are enough to generate auxiliary samples from the likelihood in this example. However, it is observed that for smaller number of iteration, such as 2 or 5 cycles, inference results from the asymptotically inexact algorithms are biased compare to asymptotically exact algorithms. Similar to the Ising model example, less correlated samples can be generated from likelihood approximation approaches, at the expense of computing time. Both ALR and AEX are computationally efficient compare to the Russian Roulette algorithm. This is because we can effectively store the previous samples using sufficient statistics in the likelihood. However, different from one-dimensional Ising model example, ALR is relatively expensive, because it takes longer to visit all state space equally in Algorithm 6 as the dimension grows. On the other hand, the performance of AEX is relatively robust in the multidimensional case and is the fastest among asymptotically exact algorithms if particles are cleverly chosen.

Summary: AEX algorithm is the most reliable approach because it is asymptotically exact while at the same time retaining some computational efficiency because this model has low-dimensional statistics. On the other hand, DMH is simple and

computationally efficient and as long as the length of the inner sampler is reasonable, it also provides accurate inference in the context of ERGM.

4.3 Spatial point process example

A spatial point process $\mathbf{x} = \{x_1, ..., x_n\}$ is a realisation of random points in a bounded plane $S \subset R^2$. By introducing interaction function $\phi(D_{ij})$ which is the function of distance between the coordinates of x_i and x_j, a probability model may be used to describe spatial patterns among the point. Extending the Strauss process (Strauss, 1975) which explains repulsion patterns among point, Goldstein et al. (2015) develop the point process model to explain both attraction and repulsion patterns of the cells infected with human respiratory syncytial virus (RSV). The interaction function is

$$\phi(D) = \begin{cases} 0 & 0 \leq D \leq R \\ \theta_1 - \left(\frac{\sqrt{\theta_1}}{\theta_2 - R}(D - \theta_2)\right)^2 & R < D \leq D_1 \\ 1 + \frac{1}{(\theta_3(D - D_2))^2} & D > D_1 \end{cases} \tag{18}$$

and the probability model is as,

$$L(\theta|\mathbf{x}) = \frac{\lambda^n \left[\prod_{i=1}^n \exp\left\{\min\left(\sum_{i \neq j} \log\left(\phi(D_{i,j})\right), 1.2\right)\right\}\right]}{Z(\theta)}, \quad \theta = \{\lambda, \theta_1, \theta_2, \theta_3\}. \tag{19}$$

There are four parameters $\{\lambda, \theta_1, \theta_2, \theta_3\}$ in the model. λ controls intensity of the point process and $\{\theta_1, \theta_2, \theta_3\}$ are the parameters regarding interaction function. θ_1 is the peak value of ϕ, θ_2 is value of D at the peak of ϕ and θ_3 represents descent rate after the peak. When the points are too close to each other, ϕ value in (18) is less than 1 which means points have a tendency to remain apart. However as the

distance between points is larger, ϕ value become increased which means that points clump together. Attraction patterns become smaller as the distance between the points become larger. Likewise, this model can capture attraction repulsion spatial association among infected cells. Calculation of the normalizing function requires integration over continuous domain S, which is intractable.

Goldstein et al. (2015) implement DMH for three independent replicates of 3,200 points in a well with radius 1,350 pixels, which is about 10,000 points. Inference took roughly 12 hours using C language, which is computationally expensive. This is because the number of point n is large, and for each iteration of the DMH, thousands of birth-death MCMC is required to generate the auxiliary variable. To allow for a comparison with other algorithms, which are computationally more expensive than DMH, we work with a smaller point pattern; however, this pattern is still computationally challenging enough to serve as a good testbed for the various algorithms we consider. Simulations are conducted on a well with 337.5 radius pixels and number of points $n \approx 200$ without replicates. Point process $\mathbf{x}$ is generated through the long run of birth-death MCMC. We follow RSV-B simulation settings in Goldstein et al. (2015), where true parameter is $\{\lambda \times 10^4, \theta_1, \theta_2, \theta_3\} = \{4, 1.2, 15, 0.3\}$ with uniform prior. Since the number of data points is small, descent rate parameter θ_3 is not recovered well. Therefore in this experiment, θ_3 is fixed as 0.3 and inference regarding other three parameters is conducted.

Implementing AVM or the exchange algorithm is infeasible because perfect sampling is impossible for this example. Although both AEX and ALR can be implemented theoretically, it is not practical because there are no summary statistics. Without summary statistics, we need to store distance matrix of cumulative point process samples with varying dimension around 200 by 200, which is expensive in the memory aspects. Therefore, DMH, noisy DMH and the Russian Roulette algo-

θ_1	Mean	95%HPD	ESS	Acc	Time(minute)	ESS/Time
DMH	1.20	(1.03,1.33)	1343.11	0.27	3.93	341.35
RussianR	1.19	(1.04,1.35)	1867.54	0.32	3299.76	0.57
NoisyDMH	1.20	(1.04,1.34)	2676.49	0.42	77.73	34.44
Gold	1.19	(1.03,1.34)	4397.39			

Table 3: Inference results for outputs for θ_1 in attraction repulsion point process model ($n \approx 200$). 40,000 MCMC samples are generated through all algorithms. The highest posterior density (HPD) is calculated by using coda package in `R`. The calculation of Effective Sample Size (ESS) follows Kass et al. (1998); Robert and Casella (2013). Acc represents acceptance probability.

rithms are conducted for this example. Each of the algorithm is tuned according to the components to be tuned part in the previous sections. For all the algorithms, samples are generated from the likelihood through 2,000 iterations of birth-death MCMC. For noisy DMH, 100 number of samples are used to construct importance sampling estimate with each iteration. Geometric series correction with $c(\theta) = 0.4$ is used for the Russian Roulette algorithm. 1,000 samples are used to construct unbiased importance sampling estimate $\widehat{Z}_i(\theta)$ and $\widetilde{Z}(\theta)$. As an importance function, $h(\mathbf{x}|\theta^{(0)})/Z(\theta^{(0)})$ is used where $\theta^{(0)}$ is sample mean from short run of DMH. Like previous examples, importance sampling estimates in noisy DMH and the Russian Roulette are evaluated through parallel run. We use the Russian Roulette algorithm as the gold standard; it was run for 101,000 iterations with 1,000 samples discarded for burn-in and 10,000 thinned samples are obtained from the remaining samples. All algorithms were run until the Monte Carlo standard error is at or below 0.01.

Here we only provide the inference results regarding θ_1 because, similar results are observed for the other parameters. Table 3 shows that inference results from the

different algorithms are well matched to the gold standard and the highest posterior density (HPD) intervals cover true values. As in the previous examples, likelihood approximation approaches can generate less correlated samples with higher acceptance rate.

Summary: DMH has huge advantages in computational efficiency. Compared to the Russian Roulette algorithm which takes about 55 hours, DMH only takes several minutes to be implemented. This is because both calculation of the $h(\mathbf{x}|\theta)$ and the inner sampler to generate auxiliary variables are expensive, compare to previous examples. Furthermore, considering ESS/T, DMH can generate effective sample within the shortest time, while simulated samples are close to the gold standard. This fact demonstrates that for computationally expensive problem, especially the model without low-dimensional sufficient statistics, we recommend DMH as a practical approach, though asymptotically exact inference is not guaranteed theoretically. Considering that the example is 16 times smaller problem than original works, it is infeasible to implement other algorithms for the original problem.

5 Computational complexity

For precise comparison, we investigated the computational complexity of the algorithms; how the algorithms scale with an increase in data points. It is challenging to calculate complexity exactly, because some of algorithms contain random quantity (stopping time), or have components to be tuned specifically for problems. Here we provide our findings based on heuristic evaluation of complexity and memory cost for each algorithm. Algorithms requiring perfect sampling will not be considered, because perfect sampling can only be constructed for limited cases. Let $L(\cdot)$ be the

complexity of evaluating $h(\mathbf{x}|\theta)$, $G(\cdot)$ be the complexity of inner sampler, and n be the number of points in data $\mathbf{x}$. Computational caveats used for simplicity of calculations are as follows.

Computational caveats

1. Dimension of data $\mathbf{x}$ and the auxiliary variable $\mathbf{y}$ are assumed to be same as n. Although this is not true for point process example, where the auxiliary variable is generated through birth-death MCMC with varying dimensions, their dimensions are approximately similar to original data's dimension n.

2. In point process example, we can store data $\mathbf{x}$ itself or distance matrix of $\mathbf{x}$ in the adaptive algorithms (AEX, ALR) to evaluate $h(\mathbf{x}|\theta)$. The latter can avoid re-computations at the expense of memory cost. Here we assume that distance matrix of $\mathbf{x}$ will be stored.

3. We assumed that the length of the inner sampler is proportional to n. It is true when the inner sampler is used for update (AEX, ALR) which requires a single cycle of update (proportional to n). However, it may not be correct if the purpose of the inner sampler is sampling (DMH, Russian Roulette, Noisy DMH) from the probability model. The mixing of the inner sampler can be different depending on the parameter settings. However, several cycles of update (proportional to n) appears to work in practice.

All the algorithms require sampling (DMH, Noisy DMH, Russian Roultte) or updating (AEX, ALR) from the probability model and evaluating $h(\mathbf{x}|\theta)$. In exponential family models such as Ising model or ERGM, complexity of $L(\cdot)$ is of order n, because number of calculations for $S(\mathbf{x})$ is proportional to n. For inner sampler, when a single point is proposed to be changed, only neighboring points (fixed number

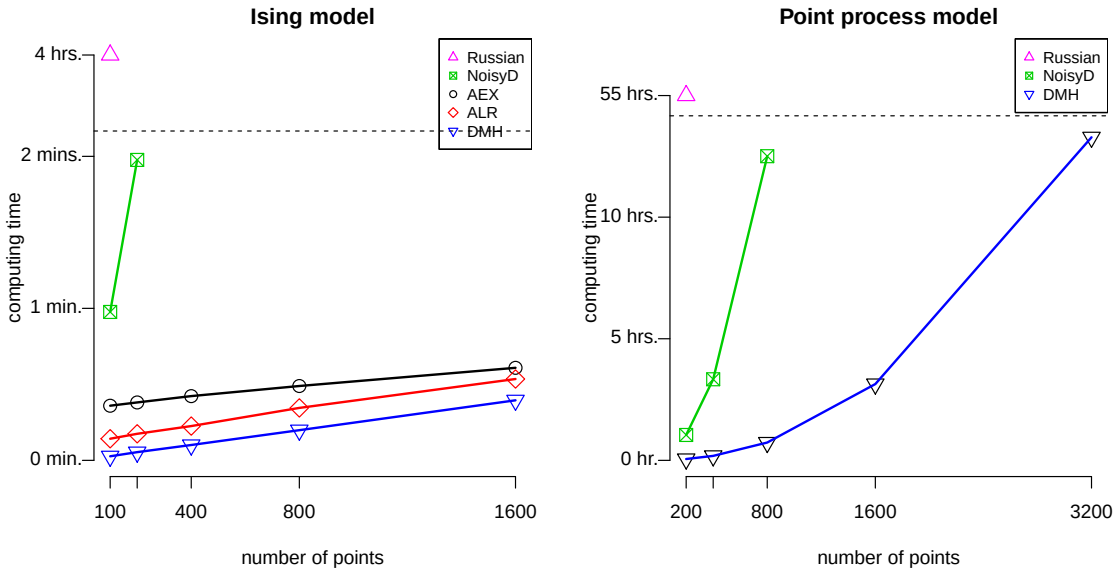

Figure 9: Illustration of the observed computing time for algorithms. For Ising model, time is measured for $\theta = 0.2$ with 20,000 iterations. For point process model, time is measured for RSV-B simulation settings in Goldstein et al. (2015) with 40,000 iterations.

regardless of n) are affected because $S(\mathbf{x})$ is Markovian statistics. Since the length of the inner sampler is assumed to be increased with n scale, $G(\cdot)$ is also of order n. On the other hand, complexity of $L(\cdot)$ for point process model is of order n^2, because evaluation of the $h(\mathbf{x}|\theta)$ requires calculating n by n distance matrix for n data points, and evaluating interaction function $\phi(\cdot)$ on the corresponding distance matrix. For inner sampler, when a single point is proposed to be added (birth) or deleted (death), the distance of a proposed point from other n points should be calculated, and then $\phi(\cdot)$ should be evaluated at each point. Since the length of the inner sampler is assumed to be proportional to n, $G(\cdot)$ is of order n^2.

There is a big difference in calculating $h(\mathbf{x}|\theta)$ for exponential family and point process model. For the exponential family model, once we evaluate $S(\mathbf{x})$, we can simply take the product of θ and $S(\mathbf{x})$ for evaluation of $h(\mathbf{x}|\theta)$ in different θ. On the other hand for point process model, $h(\mathbf{x}|\theta)$ should be recalculated; $\phi(\cdot)$ should be evaluated at the distance matrix of $\mathbf{x}$ with different parameters. This fact differs calculations in Table 6 in the supplementary material. Here we provide our major observations from the Table 6.

Major observations

1. Complexity of exponential family is $\mathcal{O}(n)$ and $\mathcal{O}(n^2)$ in point process for all the algorithms.

2. Although algorithms have same complexity, the amount of calculations with each iteration has big differences.

    - The amount of calculations per each iteration is the largest for the Russian Roulette algorithm for both models.

- The amount of calculations per each iteration of AEX, ALR and DMH are similar for exponential family models.
- Considering fixed complexity and memory cost together, DMH is the cheapest algorithm for both models.

3. For memory consumption, adaptive algorithms (AEX and ALR) are expensive.
   - With increasing iterations, both algorithms become slower because algorithms use cumulative samples H_{n+1} in calculations per each iteration (AEX: Step 5 of the Algorithm 5, ALR: Step 4 of the Algorithm 7).
   - Memory consumption may not be issues for exponential family model. However for model without low-dimensional summary statistics, it becomes severe; making both algorithms infeasible.

Figure 9 is the observed computing time for several algorithms with different scales in both models. We only include parts of results for Noisy DMH and the Russian Roulette algorithm. This is because both algorithms are too expensive to compare with other algorithms for large scales. For different scales, mixing of each algorithm is observed as similar based on effective sample size, once we use appropriate step size (covariance) for proposal. Step size can be tuned to achieve similar acceptance rate for different scales; the larger data size becomes, the smaller step is used. We can also adaptively update step size (Atchadé, 2006; Atchade et al., 2008). Figure 9 supports our calculations about $\mathcal{O}(n)$ complexity for exponential family and $\mathcal{O}(n^2)$ complexity for point process model. Also in the exponential family model, slopes of AEX, ALR and DMH are similar to each other. However DMH is the fastest because of memory consumption of both adaptive algorithms. These facts are consistent with our calculations.

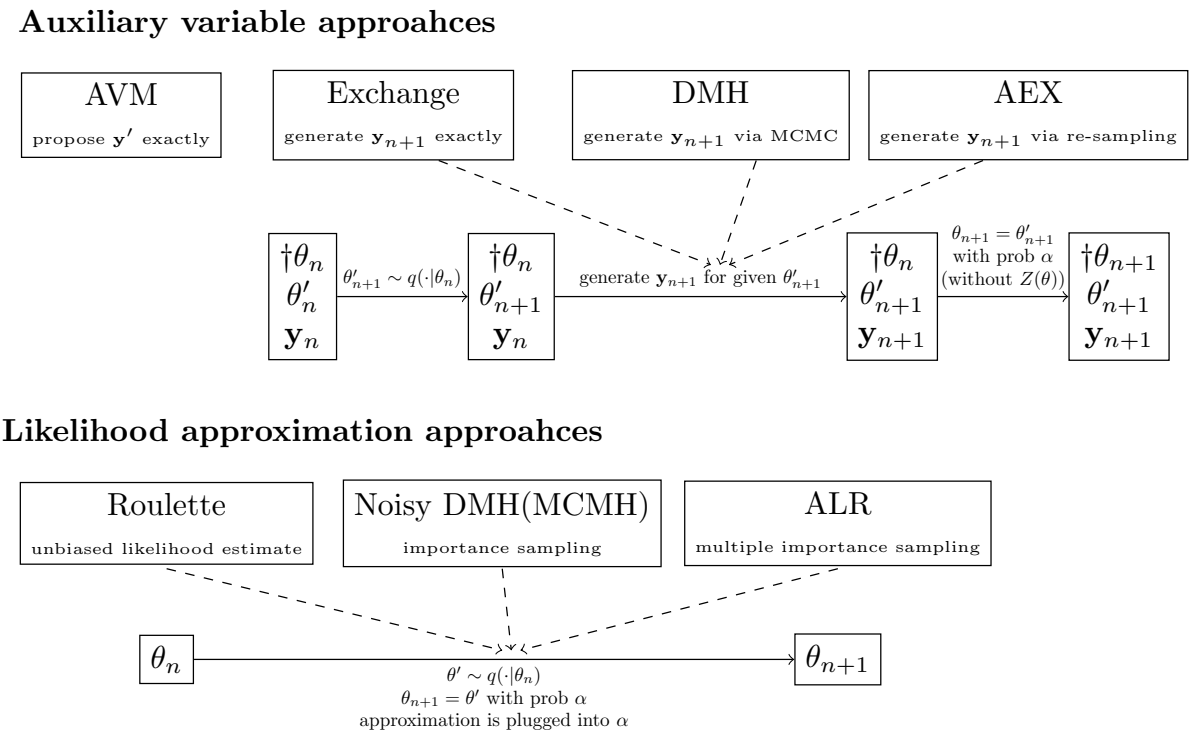

Figure 10: Connections between the algorithms. The dagger symbols indicate the parameter of interest in auxiliary variable approaches.

6 Summary of results

Here, we point out connections between the algorithms with some insights. Based on our study, we provide general guidance about advantages and disadvantages of each algorithm, along with summaries.

6.1 Connections and Observations

All the algorithms require sampling from the probability model either approximately (MCMC or particle methods) or exactly (perfect sampling). These samples, in turn,

are used in some form of an importance sampling estimate. This is clear in the likelihood approximation approaches; for the auxiliary variable approaches one may think of the acceptance probability of the Metropolis-Hastings algorithm as containing a single sample importance sampling approximation of $Z(\theta)/Z(\theta')$. Let the conditional density of the auxiliary variable in AVM be $f(\mathbf{y}|\theta, \mathbf{x}) = h(\mathbf{y}|\widehat{\theta})/Z(\widehat{\theta})$, where $\widehat{\theta}$ is the MPLE. The component of the acceptance probability that is only related to the auxiliary variable is therefore as

$$\frac{f(\mathbf{y}'|\theta', \mathbf{x})h(\mathbf{y}|\theta)}{h(\mathbf{y}'|\theta')f(\mathbf{y}|\theta, \mathbf{x})} = \frac{h(\mathbf{y}'|\widehat{\theta})h(\mathbf{y}|\theta)}{h(\mathbf{y}'|\theta')h(\mathbf{y}|\widehat{\theta})}. \tag{20}$$

Murray et al. (2006) point out that since $\mathbf{y} \sim h(\cdot|\theta)/Z(\theta)$ and $\mathbf{y}' \sim h(\cdot|\theta')/Z(\theta')$, $h(\mathbf{y}'|\widehat{\theta})/h(\mathbf{y}'|\theta')$ and $h(\mathbf{y}|\widehat{\theta})/h(\mathbf{y}|\theta)$ may be thought of as one-sample unbiased importance sampling estimate of $Z(\widehat{\theta})/Z(\theta')$ and $Z(\widehat{\theta})/Z(\theta)$ respectively. Therefore, (20) is the ratio of two unbiased estimates which is a biased estimate of $Z(\theta)/Z(\theta')$. Murray et al. (2006) explain that compare to AVM, the exchange algorithm is more direct because $h(\mathbf{y}|\theta)/h(\mathbf{y}|\theta')$ in the (4) is one-sample unbiased importance sampling estimate for $Z(\theta)/Z(\theta')$ where $\mathbf{y} \sim h(\cdot|\theta')/Z(\theta')$. Though sampling schemes for the auxiliary variable are different, same logic is applied to DMH and AEX. The only difference is that DMH generates $\mathbf{y}$ via inner sampler, and AEX generate $\mathbf{y}$ via resampling; dynmaic importance sampling from the mixture distribution. In brief, although theoretical backgrounds are different, both classes of algorithms are connected through importance sampling. We summarize connections between algorithms in Figure 10.

As briefly mentioned, AEX lies at the intersection of both approaches. In the auxiliary chain, $w_n^{(i)}$ approximates $Z(\theta^{(i)})$ at each particle up to constant as likelihood approximation approaches. Intractable functions are cancelled in the target chain as auxiliary variable approaches. Furthermore, AEX is closely connected to ALR. Both algorithms approximate $Z(\theta^{(i)})$ at each particle; $w_n^{(i)}$ in AEX and $c_n^{(i)}$ in ALR. Then

using these approximations, both collect a sample from a family of distributions $\{h(\mathbf{x}|\theta^{(1)})/Z(\theta^{(1)}), ..., h(\mathbf{x}|\theta^{(d)})/Z(\theta^{(d)})\}$ via (different) stochastic approximation and keep a sample with each iteration. With increasing iterations, collected dataset H_{n+1} grows, which make both algorithms as asymptotically exact; resampling distribution of $\mathbf{y}$ becomes close to $h(\mathbf{y}|\theta')/Z(\theta')$ in AEX and approximation $\widehat{Z}_{n+1}(\theta)$ becomes more accurate in ALR. The difference is that AEX cancels out $Z(\theta)$ while ALR plugs-in $\widehat{Z}_{n+1}(\theta)$ into the acceptance probability.

In general, likelihood approximation approaches have better mixing than auxiliary variable approaches based on effective sample size. However auxiliary variable approaches are less expensive per iteration; higher effective sample size per second. Hence, roughly speaking, auxiliary variable approaches are faster than likelihood approximation approaches. However the efficiency of algorithms can be changed depending on the underlying system of observed data (parameter settings), and decisions about tuning components in each algorithm. As there are strong dependence among data, it takes longer time to generate samples from the probability model either approximately or exactly. Perfect sampling takes longer to achieve coalescence, which can cause computational expense in AVM and the exchange algorithm. Also, MCMC sampling requires larger number of iterations, because the mixing of the inner sampler become slow in the presence of strong dependence. This can cause increased computing time for DMH and the Russian Roulette algorithm, which generate samples from the likelihood via MCMC. In this case, AEX and ALR show relatively robust performance, because both algorithms require updating $\mathbf{x}_n$, not sampling $\mathbf{x}_n$ from the probability model. However for both algorithms, the choice of particles are crucial in both statistical and computational efficiency. If θ has more dimensions, it becomes more likely that there are fewer particles close to θ. Then approximation of $Z(\theta)$ might be inaccurate (ALR) or resampled auxiliary variable might be

improbable (AEX); which can lead to poor mixing. Likewise the number of particles should be larger as the dimension grows. However without clever strategies, both algorithms require unnecessary large number of particles; which is computationally expensive. Therefore, careful choice about particles is necessary as in Algorithm 9 especially for multidimensional parameter cases.

6.2 Guidelines and Recommendations

6.2.1 Criteria for comparison of algorithms.

Comparing asymptotically exact and inexact algorithms is challenging in general because we have to simultaneously account for two different aspects of the algorithms: (1) mixing of the Markov chain, which affects the rate at which sample-based approximations converge to the true values, and (2) the quality of the target approximation in the case of asymptotically inexact algorithms. In order to provide reasonable comparisons, we always check the final results in each case using a variety of diagnostics, for example plots of marginal distributions as the Monte Carlo sample size increases, in order to convince ourselves that the approximation we obtain finally is close enough to the truth so that the above issues are avoided. Once this is the case, we can compare the efficiency of the algorithms via effective sample size (ESS) and effective samples per time (ESS/T). We have used ESS as an practical criteria of comparing mixing of the different algorithms; ESS/T can measure the computational efficiency as well.

6.2.2 Comparison and Guideline.

Based on the criteria above, we provide general guidance on the algorithms. Table 4 summarizes algorithms depending on several criteria. The criteria of comparison are

Method	Class	Exact or inexact	Adaptation	Requirement	Ease of use
AVM	A	Exact	No	Perfect sampling	3
Exchange	A	Exact	No	Perfect sampling	2
DMH	A	Inexact	No	Nothing	1
AEX	A/L	Exact	Yes	Sufficient statistics	4
ALR	L	Exact	Yes	Sufficient statistics	4
Russian Roulette	L	Exact	No	Nothing	4
Noisy DMH	L	Inexact	No	Nothing	2

Table 4: Comparison between algorithms

as follows.

Criteria

- Class: which class the each algorithm falls in. A,L indicates auxiliary variable approaches and likelihood approximation approaches respectively and A/L indicates both.
- Exact or Inexact: whether the stationary distribution of the Markov chain is exactly equal to the target posterior in the long run or not.
- Adaptation: whether the algorithm is adaptive or not.
- Requirement: practical factors necessary to implement algorithm.
- Ease of use (rank): how much users need to make decision to run the algorithm. The smallest number indicates the simplest one.

AVM and the exchange algorithm propose novel idea of cancelling out $Z(\theta)$ and give inspiration to recent algorithms. However it is difficult to implement both algorithms in practice, because the algorithms require perfect sampling. Even if we can implement perfect sampling, it still has limited applicability, because perfect sampling takes lots of time, either in the presence of strong dependence or for large scale data.

In more general cases, AEX, ALR, and the Russian Roulette can construct asymptotically exact chain without perfect sampling. Especially with low-dimensional summary statistics, AEX algorithm is the fastest. Compare to ALR, AEX is faster for multidimensional θ cases, because AEX doesn't have random stopping time τ. For ALR it is observed that τ in Algorithm 6 can be large for multidimensional θ problems, because it becomes difficult to move around state space equally. This makes ALR slower than AEX for the complex parameter space. Therefore we recommend AEX for the model with low-dimensional summary statistics such as spatial autologistic model and ERGM.

Without low-dimensional summary statistics, AEX is expensive in memory consumption. In principle, the Russian Roulette algorithm can be applicable for the general form of the likelihood without sufficient statistics. Compare to AEX or ALR, the Russian Roulette algorithm doesn't require summary statistics or assumption that $h(\mathbf{x}|\theta)$ should be bounded. However, it may not be practical because of the computational weakness. This is because, constructing unbiased likelihood estimate requires τ number of unbiased estimates $\widehat{Z}_i(\theta)$ per every iteration. If $\widehat{Z}_i(\theta)$ requires N samples, τN samples need to be generated from the likelihood using inner sampler per each iteration. For this reason, the Russian Roulette algorithm is computationally expensive compare to other methods.

We recommend DMH for the computationally expensive problems without low-

dimensional summary statistics. Since only a single sample from the likelihood is required per each iteration, DMH is computationally cheaper than any other algorithms. Furthermore, the algorithm is widely applicable without requiring any assumptions. Once, we choose the appropriate number of the inner sampler updates, we can obtain plausible inference results in practice as well as the effective samples within the shortest time (ESS/T). However users need to remind that this is only a practical approach, and exactness is not guaranteed in DMH theoretically. At last, if we can afford to pay additional computational cost, noisy DMH can be useful. The mixing of the chain becomes better, instead of generating multiple auxiliary samples per each iteration.

7 Discussion

MCMC algorithms for the likelihood with intractable normalizing functions have been investigated in this paper for both theoretical and practical aspects. We classified algorithms into two categories: (1) auxiliary variable approaches which cancel out $Z(\theta)$ and (2) likelihood approximation approaches which directly approximate $Z(\theta)$. In general, auxiliary variable approaches are faster than likelihood approximation approaches based on effective sample size per time. From both heuristic calculation of complexity and different application examples, we observed that the Russian Roulette is the most expensive and DMH is the cheapest algorithm. Furthermore, we need to care about memory issues for AEX and ALR, especially for models without low-dimensional summary statistics. All the algorithms are connected through importance sampling. Especially AEX and ALR are closely connected each other because of using particle methods to generate samples, which will grow with increasing iterations. At last, we recommended using AEX for models with low-dimensional

summary statistics and using DMH for models without summary statistics.

For users, practical implementation issue is crucial as much as theoretical justification of algorithms. Chopin and Ridgway (2015) also point out ease of use (less components to be tuned) and generic of code as important criteria for comparing Bayesian computation algorithms. From this perspective, AEX and ALR might be difficult choice for practitioners though both algorithms are asymptotically exact and moderately fast. Users have to manually tune lots of components for different problems. In specific, we recommended using AEX for social network example (models with low-dimensional sufficient statistics). However in the exponential random graph models, there are well known degeneracy problems (Handcock, 2003; Schweinberger, 2011); for some parameter region, networks are likely to be fully connected or entirely unconnected. Once particles methods (AEX, ALR) are stuck in this degeneracy region, it might be difficult for moving around state space fast. To avoid these problems, Jin et al. (2013) tuned AEX using priors avoiding degeneracy region, which requires problem specific works. Beyond ERGM example, concerning generic use of algorithms, we advocate DMH as a practical and the easiest method. One approach that we can suggest is to start inference with DMH; the fastest and generic way to investigate parameter region. Then implement AEX for asymptotically exact inference. Preliminary run of DMH can be helpful for both tuning of AEX (e.g. selecting particles) and debugging of AEX.

Thanks to all previous clever works, lots of intractable normalizing function problems can be addressed. However, there are still remaining issues that we need to solve. There can be two directions for future research. First is constructing computationally feasible asymptotically exact algorithm for general form of $h(\mathbf{x}|\theta)$. Current developed asymptotically exact algorithms are infeasible for models without low-dimensional summary statistics, because algorithm itself is too expensive (Russian Roulette) or

algorithms require large memory costs (AEX, ALR). Currently, DMH is the only feasible approach with potential bias. The other issue is developing extremely fast approximation method for large scale $\mathbf{x}$. Although DMH is the fastest, because of its nested structure, inference takes long time for large scale $\mathbf{x}$ (Goldstein et al., 2015). Compare to past, larger datasets pose significant computational challenges. Thus even developed algorithm is only an approximated approach (asymptotically inexact), if it is faster than current existing methods, the algorithm will be valuable in the era of the big data. Anticipating both asymptotically exact and extremely fast might be too ambitious.

References

Alquier, P., Friel, N., Everitt, R., and Boland, A. (2016). Noisy Monte Carlo: Convergence of Markov chains with approximate transition kernels. *Statistics and Computing*, 26(1-2):29–47.

Andrieu, C. and Roberts, G. O. (2009). The pseudo-marginal approach for efficient Monte Carlo computations. *The Annals of Statistics*, 37(2):697–725.

Atchade, Y., Lartillot, N., and Robert, C. P. (2008). Bayesian computation for statistical models with intractable normalizing constants. *Brazilian Journal of Probability and Statistics*, 27:416–436.

Atchadé, Y. F. (2006). An adaptive version for the Metropolis adjusted Langevin algorithm with a truncated drift. *Methodology and Computing in applied Probability*, 8(2):235–254.

Atchade, Y. F. and Liu, J. S. (2004). The Wang-Landau algorithm for Monte Carlo computation in general state spaces. *Statistica Sinica*, 20:209–33.

Beaumont, M. A. (2003). Estimation of population growth or decline in genetically monitored populations. *Genetics*, 164(3):1139–1160.

Beaumont, M. A., Zhang, W., and Balding, D. J. (2002). Approximate Bayesian computation in population genetics. *Genetics*, 162(4):2025–2035.

Besag, J. (1974). Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, 36:192–236.

Brooks, S., Gelman, A., Jones, G., and Meng, X.-L. (2011). *Handbook of Markov Chain Monte Carlo*. CRC press.

Butts, C. T. (2012). A perfect sampling method for exponential random graph models. *Irvine, CA: Department of Sociology, University of California*.

Caimo, A. and Friel, N. (2011). Bayesian inference for exponential random graph models. *Social Networks*, 33(1):41–55.

Carter, L. L. and Cashwell, E. D. (1975). Particle-transport simulation with the Monte Carlo method. *Technical report, Los Alamos Scientific Lab., N. Mex.(USA)*.

Chopin, N. and Ridgway, J. (2015). Leave pima indians alone: binary regression as a benchmark for Bayesian computation. *arXiv preprint arXiv:1506.08640*.

Doucet, A., Pitt, M., Deligiannidis, G., and Kohn, R. (2015). Efficient implementation of Markov chain Monte Carlo when using an unbiased likelihood estimator. *Biometrika*, page asu075.

Eddelbuettel, D., François, R., Allaire, J., Chambers, J., Bates, D., and Ushey, K. (2011). Rcpp: Seamless r and c++ integration. *Journal of Statistical Software*, 40(8):1–18.

Flegal, J. M., Haran, M., and Jones, G. L. (2008). Markov chain Monte carlo: Can we trust the third significant figure? *Statistical Science*, 23:250–260.

Geyer, C. J. (1991). Markov chain Monte Carlo maximum likelihood. *Computing Science and Statistics: Proceedings of the 23rd Symposium on the interface*, pages 156–163.

Geyer, C. J. (2011). Introduction to Markov chain Monte Carlo. In Brooks, S., Gelman, A., Meng, X.-L., and Jones, G. L., editors, *Handbook of Markov Chain Monte Carlo*. Chapman & Hall, Boca Raton.

Geyer, C. J. and Thompson, E. A. (1992). Constrained Monte Carlo maximum likelihood for dependent data. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 657–699.

Goldstein, J., Haran, M., Simeonov, I., Fricks, J., and Chiaromonte, F. (2015). An attraction-repulsion point process model for respiratory syncytial virus infections. *Biometrics*, 71(2):376–385.

Gong, L. and Flegal, J. M. (2015). A practical sequential stopping rule for high-dimensional Markov chain Monte Carlo. *Journal of Computational and Graphical Statistics*, 25(3):684–700.

Handcock, M. S. (2003). *Statistical models for social networks: Inference and degeneracy*, volume 126. National Academies Press.

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109.

Hughes, J., Haran, M., and Caragea, P. (2011). Autologistic models for binary data on a lattice. *Environmetrics*, 22(7):857–871.

Hunter, D. R. and Handcock, M. S. (2012). Inference in curved exponential family models for networks. *Journal of Computational and Graphical Statistics*.

Hunter, D. R., Handcock, M. S., Butts, C. T., Goodreau, S. M., and Morris, M. (2008). ergm: A package to fit, simulate and diagnose exponential-family models for networks. *Journal of statistical software*, 24(3).

Hunter, D. R., Krivitsky, P. N., and Schweinberger, M. (2012). Computational statistical methods for social network models. *Journal of Computational and Graphical Statistics*, 21(4):856–882.

Ibáñez, M. V. and Simó, A. (2003). Parameter estimation in Markov random field image modeling with imperfect observations. a comparative study. *Pattern recognition letters*, 24(14):2377–2389.

Ising, E. (1925). Beitrag zur theorie des ferromagnetismus. *Zeitschrift für Physik A Hadrons and Nuclei*, 31(1):253–258.

Jin, I. H. and Liang, F. (2013). Fitting social network models using varying truncation stochastic approximation MCMC algorithm. *Journal of Computational and Graphical Statistics*, 22(4):927–952.

Jin, I. H., Yuan, Y., and Liang, F. (2013). Bayesian analysis for exponential random graph models using the adaptive exchange sampler. *Statistics and its interface*, 6(4):559.

Jones, G. L., Haran, M., Caffo, B. S., and Neath, R. (2006). Fixed-width output analysis for Markov chain Monte Carlo. *Journal of the American Statistical Association*, 101(476):1537–1547.

Jones, G. L. and Hobert, J. P. (2001). Honest exploration of intractable probability distributions via Markov chain Monte Carlo. *Statistical Science*, pages 312–334.

Kass, R. E., Carlin, B. P., Gelman, A., and Neal, R. M. (1998). Markov chain Monte Carlo in practice: a roundtable discussion. *The American Statistician*, 52(2):93–100.

Lenz, W. (1920). Beitrag zum verständnis der magnetischen erscheinungen in festen körpern. *Physikalische Zeitschrift*, 21:613–615.

Liang, F. (2010). A double Metropolis–Hastings sampler for spatial models with intractable normalizing constants. *Journal of Statistical Computation and Simulation*, 80(9):1007–1022.

Liang, F. and Jin, I.-H. (2013). A Monte Carlo Metropolis-Hastings algorithm for sampling from distributions with intractable normalizing constants. *Neural computation*, 25(8):2199–2234.

Liang, F., Jin, I. H., Song, Q., and Liu, J. S. (2016). An adaptive exchange algorithm for sampling from distributions with intractable normalizing constants. *Journal of the American Statistical Association*, 111:377–393.

Liang, F., Liu, C., and Carroll, R. J. (2007). Stochastic approximation in Monte Carlo computation. *Journal of the American Statistical Association*, 102(477):305–320.

Lyne, A.-M., Girolami, M., Atchade, Y., Strathmann, H., Simpson, D., et al. (2015). On Russian roulette estimates for Bayesian inference with doubly-intractable likelihoods. *Statistical science*, 30(4):443–467.

Marin, J.-M., Pudlo, P., Robert, C. P., and Ryder, R. J. (2012). Approximate Bayesian computational methods. *Statistics and Computing*, 22(6):1167–1180.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092.

Mitrophanov, A. Y. (2005). Sensitivity and convergence of uniformly ergodic Markov chains. *Journal of Applied Probability*, 42:1003–1014.

Møller, J., Pettitt, A. N., Reeves, R., and Berthelsen, K. K. (2006). An efficient Markov chain Monte Carlo method for distributions with intractable normalising constants. *Biometrika*, 93(2):451–458.

Murray, I., Ghahramani, Z., and MacKay, D. J. C. (2006). MCMC for doubly-intractable distributions. In *Proceedings of the 22nd Annual Conference on Uncertainty in Artificial Intelligence (UAI-06)*, pages 359–366. AUAI Press.

Murray, I. A. (2007). *Advances in Markov chain Monte Carlo methods*. University of London.

Neal, R. M. (1996). Sampling from multimodal distributions using tempered transitions. *Statistics and computing*, 6(4):353–366.

Neal, R. M. (2001). Annealed importance sampling. *Statistics and Computing*, 11(2):125–139.

Padgett, J. F. (1994). Marriage and elite structure in renaissance florence, 12821500. *Paper delivered to the Social Science History Association*.

Propp, J. G. and Wilson, D. B. (1996). Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random structures and Algorithms*, 9(1-2):223–252.

Robert, C. and Casella, G. (2013). *Monte Carlo statistical methods*. Springer Science & Business Media.

Roberts, G. O. and Rosenthal, J. S. (2007). Coupling and ergodicity of adaptive Markov chain Monte Carlo algorithms. *Journal of applied probability*, 44(2):458–475.

Robins, G., Pattison, P., Kalish, Y., and Lusher, D. (2007). An introduction to exponential random graph (p*) models for social networks. *Social networks*, 29(2):173–191.

Rosenthal, J. S. (1995). Minorization conditions and convergence rates for Markov chain Monte Carlo. *Journal of the American Statistical Association*, 90(430):558–566.

Schweinberger, M. (2011). Instability, sensitivity, and degeneracy of discrete exponential families. *Journal of the American Statistical Association*, 106(496):1361–1370.

Strauss, D. J. (1975). A model for clustering. *Biometrika*, 62(2):467–475.

Torrie, G. M. and Valleau, J. P. (1977). Nonphysical sampling distributions in Monte Carlo free-energy estimation: Umbrella sampling. *Journal of Computational Physics*, 23(2):187–199.

Wang, F. and Landau, D. (2001). Efficient, multiple-range random walk algorithm to calculate the density of states. *Physical review letters*, 86(10):2050–2053.

Younes, L. (1988). Estimation and annealing for gibbsian fields. In *Annales de l'IHP Probabilités et statistiques*, volume 24, pages 269–294.

Supplementary Material for Bayesian Inference in the Presence of Intractable Normalizing Functions

A Appendix: Choice of particles in AEX

Algorithm 9 Choosing particles in the AEX

1. Generate $\{\theta_1, ..., \theta_{N_0}\}$ from the fractional DMH.

2. Choose d particles from N_0 candidates through max-min procedure.

(a) Standardize each dimension of parameter to the interval $[0, 1]$.

(b) Calculate distance matrix $[D_{st}]_{N_0 \times N_0}$ of standardized parameters.

(c) Randomly choose one particle, $\theta^{(1)} = \theta_k$ among N_0 samples in the Step 1 and define $A = \{k\}$, $A^c = \{1, 2, ..., k-1, k+1, ..., N_0\}$.

(d) Select the next particle $\theta^{(m)}$ as the farthest one, among the closest candidates from the previously chosen particles:
$\theta^{(m)} = \theta_l$ satisfying, $l \in A^c$ and $\exists\ l' \in A$ such that $D_{l,l'} = \max_{s \in A^c} \min_{t \in A} D_{st}$.
Then set $A = A \cup \{l\}$ and $A^c = A^c \setminus \{l\}$. Repeat this for $m = 2, ..., d$.

3. Return standardized particles in Step 2(a) to the original scale.

Liang et al. (2016) suggest an approach for choosing particles. Here, we provide the algorithm in details with components to be tuned for choosing particles. In Step 1 of Algorithm 9, N_0 number of candidate particles are generated through fractional DMH. Fractional DMH is a DMH with larger acceptance probability. By using an acceptance probability α^ζ with $0 < \zeta < 1$ instead of α in (7), we can draw samples from a wider region than the actual parameter space; samples can cover the support of the target posterior. Then a max-min procedure (see Steps 2 and 3

of Algorithm 9) is conducted to choose d particles among N_0 candidates. Through this procedure, the d selected particles have almost the same coverage in parameter space as N_0 candidates. We may use other strategies for choosing particles. For social network model example, Jin et al. (2013) select particles through Approximate Bayesian Computation (ABC) (Beaumont et al., 2002; Marin et al., 2012) which are approximate sampling schemes using summary statistics, without evaluating the likelihood.

Components to be tuned: To select particles, we have to decide the number of candidate particles (N_0), and the number of particles (d) in Algorithm 9. As the dimension of θ increases, we recommend using larger number of d to move around parameter space smoothly in the auxiliary chain. In this manuscript, we use $d = 100$ for a one-dimension case, and $d = 200$ for the four-dimension case. In both cases, a few thousand N_0 appear to be enough, such as $N_0 = 5,000$.

B Appendix: Complexity calculations

Here we provide details about calculating complexity and memory costs for the algorithms. Table 6 summarizes these calculations. We recommend the reader to review this section with each algorithm. In Table 6, complexity indicates computational cost for a single run of the algorithms. Memory represents the magnitude of the quantities to be saved per each iteration to avoid re-computations. Fixed complexity measures all computational costs and fixed memory represents amount of quantities to be saved, before implementing the entire algorithm. We provide notations as in Table 5. All the other notations follow previous sections.

Symbol	Definition
n	the number of data points
m	current iteration index in MCMC
N	the number of samples used to construct importance sampling estimate
N_0	the number of preliminary runs to attain particles (AEX, ALR)
N_1	the number of preliminary runs for the auxiliary chain in AEX
b	the thinning interval for MCMC
c	the number of possible cores for parallel computing
τ	random stopping time (ALR, Russian Roulette)
$\widetilde{\tau}^R$	expected stopping time for geometric Russian Roulette algorithm
p	dimension of the parameter
d	the number of particles
$S(\cdot)$	sufficient statistics
$L(\cdot)$	complexity function of evaluating $h(\cdot\|\theta)$
$G(\cdot)$	complexity function of inner sampler

Table 5: Summary of notations used for complexity calculations.

B.1 Complexity of DMH

In Algorithm 4

1. Exponential family

   - Complexity

     Step 2: A single auxiliary variable $\mathbf{y}$ is generated.

     Step 3: $S(\mathbf{y})$ is calculated once.

     Total: $G(n) + L(n)$

   - Fixed memory

Data: p-dimensional $S(\mathbf{x})$ should be stored.

Total: p

2. Point process

   - Complexity

     Step 2: A single auxiliary variable $\mathbf{y}$ is generated.

     Step 3: $h(\mathbf{x}|\theta'), h(\mathbf{y}|\theta), h(\mathbf{y}|\theta')$ are calculated.

     Total: $G(n^2) + 3L(n^2)$

   - Fixed memory

     Data: n by n distance matrix of $\mathbf{x}$ should be stored.

     Total: n^2

B.2 Complexity of Noisy DMH

Algorithm 4 with generating N auxiliary variables and acceptance probability (15).

1. Exponential family

   - Complexity

     Step 2: N number of auxiliary variables $\mathbf{y}_j$ are generated.

     Step 3: N number of corresponding $S(\mathbf{y}_j)$ are calculated.

     Total: Steps can be parallelized, $N[G(n) + L(n)]/c$

   - Fixed memory

     Data: p-dimensional $S(\mathbf{x})$ should be stored.

     Total: p

2. Point process

- Complexity

  Step 2: N number of auxiliary variables $\mathbf{y}_j$ are generated.

  Step 3: N number of $h(\mathbf{y}_j|\theta), h(\mathbf{y}_j|\theta')$ and a single $h(\mathbf{x}|\theta')$ are calculated.

  Total: Generating N number of $\mathbf{y}_j$ and evaluating N number of $h(\mathbf{y}_j|\theta), h(\mathbf{y}_j|\theta')$ can be parallelized, $N[G(n^2) + 2L(n^2)]/c + L(n^2)$

- Fixed memory

  Data: n by n distance matrix of $\mathbf{x}$ should be stored.

  Total: n^2.

B.3 Complexity of AEX

In Algorithm 5

1. Exponential family

   - Complexity

     Step 1(b): With 0.5 probability, one MH update of $\mathbf{x}'$ and $S(\mathbf{x}')$ is calculated once.

     Total: $0.5[G(n) + L(n)]$

   - Memory

     p-dimensional $S(\mathbf{x}_{m+1})$ and scalar $w_{m+1}^{(I_{m+1})}$, I_{m+1} are stored.

     Total: $p + 2$

   - Fixed complexity

     Choosing particles: N_0 iterations of fractional DMH, $N_0[G(n) + L(n)]$

     Preliminary run for auxiliary chain: $0.5N_1[G(n) + L(n)]$

     Total: $[N_0 + 0.5N_1][G(n) + L(n)]$

   - Fixed memory

     Data: p-dimensional $S(\mathbf{x})$ should be stored.

Particles: d number of p-dimensional particles, and d by d distance matrix of particles to define neighbors should be stored.

Preliminary run for auxiliary chain: p-dimensional $S(\mathbf{x}_{m+1})$ and scalar $w_{m+1}^{(I_{m+1})}$, I_{m+1} are stored for N_1 iterations, and can be stored with b thinning interval.

Total: $p + dp + d^2 + (p+2)N_1/b$

2. Point process

- Complexity

  Step 1(a): With 0.5 probability, $h(\mathbf{x}_m|\theta^{(I')})$ is calculated once.

  Step 1(b): With 0.5 probability, one MH update of $\mathbf{x}'$ and $h(\mathbf{x}'|\theta^{(I_m)})$ is calculated once.

  Step 5: $|H_{m+1}|$ number of $h(\mathbf{x}_j|\theta')$ should be calculated. With N_1 number of preliminary run and b thinning interval for the auxiliary chain, $|H_{m+1}|$ is equal to $N_1/b + m + 1$.

  Step 6: $h(\mathbf{x}|\theta'), h(\mathbf{y}|\theta), h(\mathbf{y}|\theta')$ are calculated.

  Total: $0.5G(n^2) + [N_1/b + m + 5]L(n^2)$

- Memory

  n by n distance matrix of $\mathbf{x}_{m+1}$, scalar $w_{m+1}^{(I_{m+1})}$, I_{m+1} and evaluated $h(\mathbf{x}_{m+1}|\theta^{(I_{m+1})})$ are stored.

  Total: $n^2 + 3$

- Fixed complexity

  Choosing particles: N_0 iterations of fractional DMH, $N_0[G(n^2) + 3L(n^2)]$

  Preliminary run for auxiliary chain: $N_1[0.5L(n^2) + 0.5[G(n^2) + L(n^2)]]$

  Total: $[N_0 + 0.5N_1]G(n^2) + [3N_0 + N_1]L(n^2)$

- Fixed memory

Data: n by n distance matrix of $\mathbf{x}$ should be stored.

Particles: d number of p-dimensional particles, and d by d distance matrix of particles to define neighbors should be stored.

Preliminary run for auxiliary chain: n by n distance matrix of $\mathbf{x}_{m+1}$, scalar $w_{m+1}^{(I_{m+1})}$, I_{m+1} and evaluated $h(\mathbf{x}_{m+1}|\theta^{(I_{m+1})})$ are stored for N_1 iterations, and can be stored with b thinning interval.

Total: $n^2 + dp + d^2 + (n^2 + 3)N_1/b$

In practice, AEX consists of three parts: (1) choosing particles, (2) preliminary run for the auxiliary chain, and (3) implementing the entire algorithm. Fixed complexity and fixed memory are evaluated from the first two parts.

B.4 Complexity of ALR

In Algorithm 6 and Algorithm 7

1. Exponential family

   - Complexity

     Step 1: $\mathbf{x}_{m+1}$ is generated once.

     Step 2: $S(\mathbf{x}_{m+1})$ is calculated once.

     Total: $G(n) + L(n)$

   - Memory

     p-dimensional $S(\mathbf{x}_{m+1})$ and scalar I_{m+1} is stored.

     Total: $p + 1$

   - Fixed complexity

     Choosing particles: N_0 iterations of DMH, $N_0[G(n) + L(n)]$

     Stopping time: Step 1-3 are repeated until τ, $\tau[G(n) + L(n)]$

     Total: $(N_0 + \tau)[G(n) + L(n)]$

- Fixed memory

  Data: p-dimensional $S(\mathbf{x})$ should be stored.

  Particles: d number of p-dimensional particles are stored.

  Stopping time: d-dimensional $\mathbf{c}_m$ is stored.

  Total: $p + dp + d$

2. Point process

- Complexity

  Step 1: $\mathbf{x}_{m+1}$ is generated once.

  Step 2: d number of $h(\mathbf{x}_{m+1}|\theta^{(i)})$ are calculated, which can be parallelized.

  Step 4: $h(\mathbf{x}_j|\theta'), h(\mathbf{x}_j|\theta)$ are evaluated for $m+1$ number of cumulative $\mathbf{x}_j$, which can be parallelized.

  Step 5: $h(\mathbf{x}|\theta')$ is evaluated once.

  Total: $G(n^2) + [(d + 2m + 2)/c + 1]L(n^2)$

- Memory

  n by n distance matrix of $\mathbf{x}_{m+1}$, scalar I_{m+1} and d number of $h(\mathbf{x}_{m+1}|\theta^{(i)})$ should be stored.

  Total: $n^2 + d + 1$

- Fixed complexity

  Choosing particles: N_0 iterations of DMH, $N_0[G(n^2) + 3L(n^2)]$

  Stopping time: Step 1-3 are repeated until τ, $\tau[G(n^2) + dL(n^2)/c]$

  Total: $[N_0 + \tau]G(n^2) + [3N_0 + \tau d/c]L(n^2)$

- Fixed memory

  Data: n by n distance matrix of $\mathbf{x}$ should be stored.

  Particles: d number of p-dimensional particles are stored.

  Stopping time: d-dimensional $\mathbf{c}_m$ is stored.

Total: $n^2 + dp + d$

Similar to AEX, the algorithm consists of three parts: (1) choosing particles, (2) repeating step 1-3 in Algorithm 6, until $\mathbf{c}_m$ converges, and (3) updating the entire process in Algorithm 7. Fixed complexity and fixed memory are evaluated from the first two parts.

It is observed that random stopping time τ is affected by complexity of parameter space and choice of particles rather than n. Intuitively, as the particles cover important region of the parameter space, it is easier to move around state space quickly, compare to particles simply collected from the entire parameter space. Therefore, proper choice of particles can lead smaller τ. It is observed that if we choose particles covering interesting region of parameter space as in Algorithm 9, τ is almost fixed even with increasing n.

B.5 Complexity of the Russian Roulette algorithm

In Algorithm 8

1. Exponential family

   - Complexity

     Step 2: Importance sampling estimate, $\widetilde{Z}(\theta')$ is constructed, which requires generating N number of $\mathbf{y}_j$ and calculating $S(\mathbf{y}_j)$.

     Step 3: τ number of $\widehat{Z}_i(\theta')$ are constructed, which require generating N number of $\mathbf{y}_j$ and calculating $S(\mathbf{y}_j)$ per each $\widehat{Z}_i(\theta')$.

     Total: Steps can be parallelized, $N(\tau + 1)[G(n) + L(n)]/c$

   - Fixed Memory

     Data: p-dimensional $S(\mathbf{x})$ should be stored.

     Total: p

2. Point process

- Complexity

  Step 2: Importance sampling estimate, $\widetilde{Z}(\theta')$ is constructed, which requires generating N number of $\mathbf{y}_j$ and calculating $h(\mathbf{y}_j|\theta'), h(\mathbf{y}_j|\theta^{(0)})$.

  Step 3: τ number of $\widehat{Z}_i(\theta')$ are constructed, which require generating N number of $\mathbf{y}_j$ and calculating $h(\mathbf{y}_j|\theta'), h(\mathbf{y}_j|\theta^{(0)})$ per each $\widehat{Z}_i(\theta')$. Then $h(\mathbf{x}|\theta')$ is calculated.

  Total: Step 2-3 can be parallelized, $N(\tau+1)[G(n^2)+2L(n^2)]/c + L(n^2)$

- Fixed Memory

  Data: n by n distance matrix of $\mathbf{x}$ should be stored.

  Total: n^2.

$$E[\tau] = \sum_{x=1}^{\infty} xp(\tau = x) = \sum_{x=1}^{\infty} xk^{x(x-1)/2}(1-k^x) = \sum_{x=1}^{\infty} k^{x(x-1)/2} = \widetilde{\tau}^R \tag{21}$$

Because τ in the complexity is random variable, we propose heuristic evaluation of expected stopping time. Consider q_j, where $p(\tau \geq x) = \prod_{j=1}^{x-1} q_j$ as we defined in the previous section. We can use $q_j = \prod_{i=1}^{j} \widehat{k}_i(\theta)$, where $\widehat{k}_i(\theta) = 1 - c(\theta)\widehat{Z}_i(\theta)/\widetilde{Z}(\theta)$ as suggested in Lyne et al. (2015). For large enough importance sampling size N, $\widehat{Z}(\theta)_i/\widetilde{Z}(\theta)$ will be close to 1 when $\widetilde{Z}(\theta)$ is importance sampling estimate as well. In this case, $\widehat{k}_i(\theta) \approx k = 0.6$ is almost constant because $c(\theta) = 0.4$ is used for our examples. Therefore, $q_j \approx k^j$ and expected stopping time can be represented as (21). For $c(\theta) = 0.4$, $\widetilde{\tau}^R \approx 1.86$.

Then remaining issue is how large N should be depending on size of data n. N should be tuned to make $|\widehat{k}_i(\theta)| < 1$ for convergence of geometric series. As N increases, it is oberved that the mixing of the chain become better (higher effective sample size) and ergodic average with lower Monte Carlo standard error can be

attained at the expense of increased computational costs. How to tune N efficiently is still debatable. In pseudo-marginal MCMC, Doucet et al. (2015) propose the rule of thumb to choose N so that standard deviation of the log-likelihood estimate is around 1.2. Although this can be a one guideline to tune N, it is found that suggested N is too large in our point process example which can lead unnecessary costs. Also, it is found that using Annealed importance sampling estimate can achieve lower variance in many problems than using simple importance sampling estimate. Considering all possibilities, it is challenging to generalize how N should be tuned depending on n, because it can be problem specific.

DMH	Exponential family	Point process
C	$G(n) + L(n)$	$G(n^2) + 3L(n^2)$
FM	p	n^2
NoisyDMH	Exponential family	Point process
C	$N[G(n) + L(n)]/c$	$NG(n^2)/c + [2N/c + 1]L(n^2)$
FM	p	n^2
$*$AEX	Exponential family	Point process
C	$0.5[G(n) + L(n)]$	$0.5G(n^2) + [N_1/b + m + 5]L(n^2)$
M	$p + 2$	$n^2 + 3$
FC	$[N_0 + 0.5N_1][G(n) + L(n)]$	$[N_0 + 0.5N_1]G(n^2) + [3N_0 + N_1]L(n^2)$
FM	$p + dp + d^2 + (p + 2)N_1/b$	$n^2 + dp + d^2 + (n^2 + 3)N_1/b$
$*$ALR	Exponential family	Point process
C	$G(n) + L(n)$	$G(n^2) + [(d + 2m + 2)/c + 1]L(n^2)$
M	$p + 1$	$n^2 + d + 1$
FC	$(N_0 + \tau)[G(n) + L(n)]$	$[N_0 + \tau]G(n^2) + [3N_0 + \tau d/c]L(n^2)$
FM	$p + dp + d$	$n^2 + dp + d$
$*$Roulette	Exponential family	Point process
C	$N(\widetilde{\tau}^R + 1)[G(n) + L(n)]/c$	$N(\widetilde{\tau}^R + 1)G(n^2)/c + [2N(\widetilde{\tau}^R + 1)/c + 1]L(n^2)$
FM	p	n^2

Table 6: Computational complexity and memory size of the algorithms in the simulation examples. If costs are negligible, we didn't include in the table. C, M, FC, FM denote complexity, memory, fixed complexity, fixed memory respectively. Asterisk symbols indicate calculated complexity is expected number.