Journal of Machine Learning Research 22 (2021) 1-41 Submitted 9/18; Revised 5/21; Published 5/21

Analysis of high-dimensional Continuous Time Markov Chains using the Local Bouncy Particle Sampler

Tingting Zhao ZHAOTT0416@GMAIL.COM
College of Information and Computer Sciences
University of Massachusetts Amherst
Amherst, MA, 01003, USA

Alexandre Bouchard-Côté BOUCHARD@STAT.UBC.CA
Department of Statistics
University of British Columbia
Vancouver, BC, V6T 1N4, Canada

Editor: Zhihua Zhang

Abstract

Sampling the parameters of high-dimensional Continuous Time Markov Chains (CTMCs) is a challenging problem with important applications in many fields of applied statistics. In this work a recently proposed type of non-reversible rejection-free Markov Chain Monte Carlo (MCMC) sampler, the Bouncy Particle Sampler (BPS), is brought to bear to this problem. BPS has demonstrated its favourable computational efficiency compared with state-of-the-art MCMC algorithms, however to date applications to real-data scenario were scarce. An important aspect of practical implementation of BPS is the simulation of event times. Default implementations use conservative thinning bounds. Such bounds can slow down the algorithm and limit the computational performance. Our paper develops an algorithm with exact analytical solution to the random event times in the context of CTMCs. Our local version of BPS algorithm takes advantage of the sparse structure in the target factor graph and we also provide a graph-theoretic tool for assessing the computational complexity of local BPS algorithms.

Keywords: CTMCs, Hamiltonian Monte Carlo (HMC), Piecewise-deterministic Markov Process (PDMP), BPS, Local Bouncy Particle Sampler (LBPS), Generalized Linear Models (GLM)

1. Introduction

CTMCs have widespread applications ranging from chronic multi-state disease progression (Chen et al., 1996; Combescure et al., 2003; Saeedi and Bouchard-Côté, 2011; Liu et al., 2015) to phylogenetics (Yin and Zhang, 2012). However, the estimation of parameters in CTMCs is a challenging problem when incomplete data observations are only available at a finite number of time points. This is the case in a wide range of applications, for analyzing censored survival data (Kay, 1977), for describing panel data under Markov assumptions (Kalbfleisch and Lawless, 1985), for characterizing multi-state disease progression (Jackson et al., 2003), and for inferring evolutionary processes (Jukes and Cantor, 1969; Zhao et al., 2016) using biological sequences.

A (homogeneous) CTMC is a continuous-time stochastic process taking values on a finite or countable set. The parameters involved in a CTMC are used to characterize the transitions between states and the distributions of the intervals between two consecutive transitions. The parameters are organized into a rate matrix. If the sample paths have been completely observed continuously over a finite time interval, statistical inference is straightforward. However, a more typical situation is that only partial observations of the states on a finite number of time points are available. For high dimensional rate matrices, efficient posterior inference is challenging. In particular, despite several algorithmic advances (Moler and Van Loan, 2003), matrix exponentiation, which is required to compute the marginal distributions of CTMCs, is still computationally expensive.

In the following, we will make use of a flexible framework to parameterize rate matrices (Zhao et al., 2016), which subsumes much of the earlier parameterizations (Kimura, 1980; Hasegawa et al., 1985). This previous work used off-the-shelf Adaptive Hamiltonian Monte Carlo (AHMC) methods and did not exploit the sparsity often found in the parameterization of high-dimensional rate matrices.

In order to exploit sparsity, we make use of recently developed Monte Carlo schemes based on non-reversible PDMPs. In particular, we build our samplers based on the non-reversible rejection-free dynamics proposed in Peters et al. (2012) in the physics literature and later developed for statistical applications in Bouchard-Côté et al. (2018). It has been shown by Neal (2004); Sun et al. (2010); Chen and Hwang (2013); Bierkens (2016) that non-reversible MCMC algorithms can outperform reversible MCMC in terms of mixing rate and asymptotic performance. Related sampling schemes have been developed including continuous-time Monte Carlo algorithms and continuous-time Sequential Monte Carlo algorithms (Pakman et al., 2017; Fearnhead et al., 2018). Bierkens et al. (2019) have developed a super-efficient sampling algorithm based on the zig-zag process under a big data context. But we focus on proof-of-concept applications of the BPS algorithm in this work.

In the BPS algorithm, the posterior samples of a variable of interest are continuously indexed by the position of a particle that moves along piecewise linear trajectories. When encountering a high energy barrier (low posterior density value), the particle is never rejected but instead the direction of its path is changed after a Newtonian collision. A key algorithmic requirement is to efficiently determine the bouncing time of the particle. Most existing work (Vanetti et al., 2017; Pakman et al., 2017; Fearnhead et al., 2018) use conservative bounds from the thinning algorithm of an inhomogeneous Poisson Process (PP) to sample the collision time. Conservative bounds can lead to computational inefficiency of the algorithm. Bouchard-Côté et al. (2018) have obtained analytical solutions to the collision times under certain simple scenarios such as Gaussian distribution. Thus, the BPS can be highly efficient but does require application-specific work since the derivation of the collision time is case-specific. Therefore, we focus on demonstrating a concrete case study of how this can be achieved in the context of CTMCs and what potential benefits can be obtained.

The key contributions of this paper are as follows:

- Efficient algorithms to simulate the bouncing time for each factor of the factorized posterior density of CTMCs to boost the computational efficiency.

- A novel sampler combining HMC and LBPS to efficiently sample from CTMCs. This sampler is also of interest for sampling in other sparse factor graphs.
- A graph-theoretic tool to help select optimal sets of variables to be updated with HMC and which ones to be updated with LBPS, both for the special case of CTMCs but also more generally in sparse factor graphs.
- A proof-of-concept application on protein evolution to demonstrate on real data the computational efficiency of LBPS compared to state-of-the-art HMC algorithms.

The remaining of the paper is organized as follows. Section 2 provides background on CTMCs and on the BPS algorithm. Section 3 describes a novel model for CTMCs used to demonstrate our computational methodology. Sections 4 and 5 describe our key theoretical and methodological contributions. In Section 6, we compare our novel sampler to state-of-the-art methods. Finally, in Section 7, we provide results on protein data, and we discuss potential extensions of our work in Section 8.

2. Problem setup and notations

2.1 CTMCs notation

We first introduce some notation for CTMCs. More background on CTMCs can be found in Norris (1998); Guttorp and Minin (2018). We use the same notation as Zhao et al. (2016). A (homogeneous) CTMC is a continuous-time stochastic process $\{X(t) : t \geqslant 0\}$ taking values on a finite or countable set $\mathcal{X}$. Throughout the paper, we assume $\mathcal{X}$ is finite and $\mathcal{X} = \{1, 2, \ldots, |\mathcal{X}|\}$. Denote $\{X_n, n \geqslant 0\}$ as the sequence of states visited in the continuous time path $\{X(t)\}$, and let A_n be the corresponding times when the state changes. A rate matrix Q indexed by $\mathcal{X}$ is used to describe the instantaneous transition rate for each pair of distinct states in $\mathcal{X}$. For example, $q_{x,x'}$ represents the instantaneous rate between x and x', where $x \in \mathcal{X}, x' \in \mathcal{X}, x \neq x'$. The diagonal elements of Q are negative and enforce the constraint that each row sums to zero. The absolute values of the diagonal elements represent the rate parameter of an exponential distribution, used to characterize the waiting time spent on each state. We denote the initial state distribution as $p_{\text{ini}}(\cdot)$. Given the rate matrix Q, the transition probability matrix is $P_Q(\Delta) = \exp(\Delta Q) = \sum_{j=0}^{\infty} (\Delta Q^j)/j!$, where $\Delta \geq 0$ corresponds to a time-step size. We use $\pi = (\pi_1, \pi_2, \ldots, \pi_{|\mathcal{X}|})$ to represent the stationary distribution of the CTMC.

We use $\boldsymbol{x}$ to represent N fully observed CTMC paths (example shown in Figure 1). We define $(\boldsymbol{n}, \boldsymbol{h}, \boldsymbol{c}) := (\boldsymbol{n}(\boldsymbol{x}), \boldsymbol{h}(\boldsymbol{x}), \boldsymbol{c}(\boldsymbol{x}))$ as the sufficient statistics of N fully observed CTMC paths. For all $x \in \mathcal{X}$, denote $n_x := n_x(\boldsymbol{x}) \in \{0, 1, 2, \ldots\}$ as the number of paths started at state x and $\boldsymbol{n}(\boldsymbol{x}) := (n_1(\boldsymbol{x}), n_2(\boldsymbol{x}), \ldots, n_{|\mathcal{X}|}(\boldsymbol{x}))$. Similarly, for all $x \in \mathcal{X}$, let $h_x \in [0, \infty)$ denote the total time spent in state x and $\boldsymbol{h}(\boldsymbol{x}) := (h_1(\boldsymbol{x}), h_2(\boldsymbol{x}), \ldots, h_{|\mathcal{X}|}(\boldsymbol{x}))$. For all distinct $(x, x') \in \mathcal{X}^{\text{distinct}} := \{(x, x') \in \mathcal{X}^2 : x \neq x'\}$, let $c_{x,x'}$ denote the number of jumps from state x to x' and let $\boldsymbol{c}(\boldsymbol{x})$ denote the vector of all transition counts organized according to some arbitrary fixed order of $\mathcal{X}^{\text{distinct}}$. For simplicity, we denote $\boldsymbol{z} := (\boldsymbol{n}, \boldsymbol{h}, \boldsymbol{c})$ and $\boldsymbol{z}(\boldsymbol{x}) := (\boldsymbol{n}(\boldsymbol{x}), \boldsymbol{h}(\boldsymbol{x}), \boldsymbol{c}(\boldsymbol{x}))$. The density of N fully observed paths $\boldsymbol{x}$ from a CTMC with rate matrix Q over time interval length Δ is given by

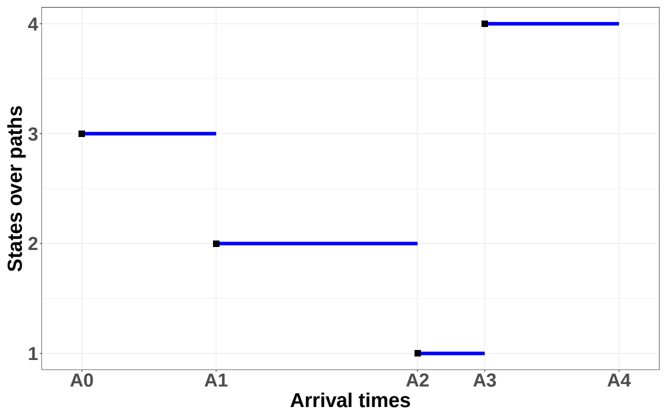

Figure 1: A fully observed realization of a single path of CTMCs at arrival time points A_1, A_2, A_3, A_4 over a state space $\{1, 2, 3, 4\}$.

$$
\begin{aligned}
f_{x|Q,\Delta}(\boldsymbol{x}|Q,\Delta) \quad := \quad & \left(\prod_{x\in\mathcal{X}} p_{\text{ini}}(x)^{n_x}\right)\left(\prod_{(x,x')\in\mathcal{X}^{\text{distinct}}} q_{x,x'}^{c_{x,x'}}\right) \\
& \times \left(\prod_{x\in\mathcal{X}} \exp\left(h_x q_{x,x}\right)\right) \mathbf{1}[\boldsymbol{x}\in S(\Delta)],
\end{aligned} \tag{1}
$$

where $S(\Delta)$ denotes the set of CTMC paths of total length Δ.

Usually, these paths are only partially observed at a finite number of points $\tau_0, \tau_1, \ldots, \tau_n$ with states $y_0, y_1, \ldots$ and y_n. We use $\mathcal{Y}$ to represent the partially observed CTMC path. Assuming there is no error in the observation of the states, the density over this single path is

$$
g_{\text{y}|\text{Q}}(\mathcal{Y}|Q) = p_{\text{ini}}(y_{\tau_0}) \prod_{k=1}^{K} \left(\exp(Q\Delta_k)\right)_{y_{\tau_{k-1}}, y_{\tau_k}}, \tag{2}
$$

where $\Delta_k = \tau_k - \tau_{k-1}$ is the length of the time interval between two consecutive observations.

The question of interest is often to estimate the rate matrix given partially observed paths. Pointwise evaluation of Equation 2 can be done in $\mathcal{O}(|\mathcal{X}|^3)$ using diagonalization method to evaluate the matrix exponential. We reduce the computational cost by introducing the substitution mapping auxiliary variables (instantiated via uniformization algorithms) as described in Zhao et al. (2016). With the augmented sufficient statistics, the density of the paths is given by Equation 1.

2.2 Bayesian GLM parameterization

We describe here a sparse rate matrix parameterization built in the framework of Zhao et al. (2016). This is useful since we propose Bayesian GLM chain General Time Reversible model (GTR) model in Section 3 on the basis of Bayesian GLM reversible rate matrix parameterization (Zhao et al., 2016). We first describe the GTR model and then review briefly the Bayesian GLM unnormalized reversible rate matrix parameterization. Finally, we review the Bayesian GLM rate matrix representation for a GTR model since the notation will be needed to introduce our new sparse model in Section 3.

We introduce the set of unordered distinct pairs of states as $\mathcal{X}^{\text{unordered,dist.}} := \{\{x, x'\} \in \mathcal{X}^2 : x \neq x'\}$. Recall that in a GTR model, Q is parameterized by the stationary distribution $\pi = (\pi_1, \pi_2, \ldots, \pi_{|\mathcal{X}|})$ and exchangeable parameters $\theta_{\{x,x'\}}$, where $q_{x,x'} = \theta_{\{x,x'\}}\pi_{x'}, x \neq x'$. In total, there are $p_2 = |\mathcal{X}|(|\mathcal{X}| - 1)/2$ exchangeable parameters $\theta_{x,x'}$ under reversibility. In the Bayesian GLM unnormalized reversible rate matrix parameterization, we have

$$\begin{aligned}
\theta_{\{x,x'\}}(\boldsymbol{w}^b) &:= \exp\left\{\langle \boldsymbol{w}^b, \boldsymbol{\phi}(\{x, x'\})\rangle\right\}, &(3)\\
\pi_x(\boldsymbol{w}^u) &:= \exp\left\{\langle \boldsymbol{w}^u, \boldsymbol{\psi}(x)\rangle - A(\boldsymbol{w}^u)\right\}, &(4)\\
A(\boldsymbol{w}^u) &:= \log \sum_{x\in\mathcal{X}} \exp\left\{\langle \boldsymbol{w}^u, \boldsymbol{\psi}(x)\rangle\right\}, &(5)\\
q^{(\text{rev})}_{x,x'}(\boldsymbol{w}) &:= \theta_{\{x,x'\}}\left(\boldsymbol{w}^b\right)\pi_{x'}\left(\boldsymbol{w}^u\right). &(6)
\end{aligned}$$

The parameters of interest are $\boldsymbol{w}$, where $\boldsymbol{w} = \begin{pmatrix}\boldsymbol{w}^u\\ \boldsymbol{w}^b\end{pmatrix}$. In Equation 3, we introduce bivariate feature functions $\boldsymbol{\phi} : \mathcal{X}^{\text{unordered,dist.}} \to \mathbb{R}^{p_2}$. In Equation 4, we introduce univariate feature functions $\boldsymbol{\psi} : \mathcal{X} \to \mathbb{R}^{p_1}$. We have $p_1 + p_2 = p$ and $\boldsymbol{w} \in \mathbb{R}^p$. The weights related to the *univariate features* $\boldsymbol{\psi}$ are denoted as univariate weights $\boldsymbol{w}^u$ and the weights related to the *bivariate features* $\boldsymbol{\phi}$ are denoted as bivariate weights $\boldsymbol{w}^b$.

The connection between Bayesian GLM rate matrix representation and a GTR model is discussed in details in Supplementary 8.1 and Supplementary 8.2 by Zhao et al. (2016). In Supplementary 8.1, Zhao et al. (2016) have proved that the Bayesian GLM can be used to represent any rate matrices under the GTR parameterization equivalently. Here, we review this construction since our new sparse model in Section 3 can be represented in a similar fashion. To represent the GTR model in a Bayesian GLM framework, if we have a GTR rate matrix Q with stationary distribution π_x for any $x \in \mathcal{X}$ and $\sum_{x\in\mathcal{X}} \pi_x = 1$, we can pick any reference state x^* such that

$$\pi_{x^*}(\boldsymbol{w}) = \frac{1}{1 + \sum_{x'\neq x^*:x'\in\mathcal{X}} \exp(\langle \boldsymbol{w}, \phi(x')\rangle)},$$

and for other state x',

$$\pi_{x'}(\boldsymbol{w}) = \pi_{x^*}(\boldsymbol{w}) \exp(\langle \boldsymbol{w}, \phi(x')\rangle).$$

Thus, for any stationary distribution π_x of a GTR rate matrix Q, there exists a unique set of weights $\boldsymbol{w}$ such that $\pi_x(\boldsymbol{w}) = \pi_x$.

Similarly, we define a map η: $\mathcal{X}^{\text{unordered,dist.}} \to \{1, 2, \ldots, |\mathcal{X}|(|\mathcal{X}| - 1)/2\}$ such that the ith element of *bivariate features* $\boldsymbol{\phi}^{\text{gtr}} : \mathcal{X}^{\text{unordered,dist.}} \to \mathbb{R}^{p_2}$ is defined as:

$$\boldsymbol{\phi}^{\text{gtr}}_i(\{x, x'\}) := \mathbf{1}(\eta(\{x, x'\}) = i), \tag{7}$$

where the ith feature $\boldsymbol{\phi}_i^{\text{gtr}}(\{x, x'\})$ is equal to one if and only if the pair of unordered states $\{x, x'\}$ is mapped to ith exchangeable parameter via $\eta(\{x, x'\})$. Under the definition of $\boldsymbol{\phi}^{\text{gtr}}$, for any $\theta_{x,x'}$ of a GTR rate matrix Q, there exists a unique set of weights such that $\theta_{x,x'} = \exp\left(\boldsymbol{w}_{\eta(\{x,x'\})}\right)$.

2.3 Bayesian inference

Under the Bayesian GLM rate matrix parameterization defined in Equation 3-6, the parameters of interest are $\boldsymbol{w}$, where $\boldsymbol{w} = \begin{pmatrix} \boldsymbol{w}^u \\ \boldsymbol{w}^b \end{pmatrix}$. We place a prior on $\boldsymbol{w}$ with density denoted by $g_{\text{w}}(\boldsymbol{w})$. Following Zhao et al. (2016), we use a Gaussian distribution with mean zero and precision κ for $g_{\text{w}}(\boldsymbol{w})$.

We assume the observations are partially observed states $y_1, y_2, \ldots, y_k$ at a finite number of time points $\tau_0, \tau_1, \ldots, \tau_k$ on each sample path. To simplify the notations, we assume that $\tau_0, \tau_1, \ldots, \tau_k$ are fixed and known. The density over one single path is given in Equation 2. Under the Bayesian GLM parameterization, we denote the density by $g_{\text{y}|\text{w}}(\mathcal{Y}|Q(\boldsymbol{w}))$. Following Equation 2, we obtain

$$g_{\text{y}|\text{w}}(\mathcal{Y}|Q(\boldsymbol{w})) = p_{\text{ini}}(y_{\tau_0}) \prod_{k=1}^{K} \left(\exp(Q(\boldsymbol{w})\Delta_k)\right)_{y_{\tau_{k-1}}, y_{\tau_k}}, \tag{8}$$

Thus, the target posterior density is given by

$$g_{\text{w}|\text{y}}(\boldsymbol{w}|\mathcal{Y}) \propto g_{\text{w}}(\boldsymbol{w}) g_{\text{y}|\text{w}}(\mathcal{Y}|Q(\boldsymbol{w})) = \exp(-\tilde{U}(\boldsymbol{w})), \tag{9}$$

where $\tilde{U}(\boldsymbol{w})$ represents the negative logarithm of the unnormalized posterior density function.

In Equation 9, the second term involves the computation of matrix exponentials. Zhao et al. (2016) have argued that using HMC directly on $\tilde{U}(\boldsymbol{w})$ leads to a high computational cost as a single gradient evaluation of Equation 8 has a running time of $\Theta(|\chi|^5)$ (Zhao et al., 2016). Applying other gradient-based MCMC methods such as BPS or LBPS would suffer from the same problem.

To circumvent this computational difficulty, we follow the auxiliary variable strategy of Zhao et al. (2016), where *substitution mapping* is used before each sampling step to augment the state space with (sufficient statistics of) full CTMC sample paths. More precisely, given a pair of consecutive observed states separated by a time interval Δ, we simulate a full path conditionally on the end-points and time interval Δ according to Equation 1 using the cached uniformization techniques described in Zhao et al. (2016). After doing this for each pair of consecutive observed states, we obtain a vector of sufficient statistics $\boldsymbol{z} := (\boldsymbol{n}(\boldsymbol{x}), \boldsymbol{h}(\boldsymbol{x}), \boldsymbol{c}(\boldsymbol{x}))$ as defined in Section 2.1.

Following Equation 1, the negative unnormalized posterior log-density with augmented sufficient statistics $\boldsymbol{z}$ is

$$\begin{aligned} U(\boldsymbol{w}) \coloneqq U_{\boldsymbol{z}}(\boldsymbol{w}) \quad &\coloneqq \quad \frac{1}{2}\kappa\|\boldsymbol{w}\|_2^2 - \sum_{x\in\mathcal{X}} h_x q_{x,x}(\boldsymbol{w}) \\ &\quad - \sum_{(x,x')\in\mathcal{X}^{\text{distinct}}} c_{x,x'} \log\left(q_{x,x'}(\boldsymbol{w})\right) - \sum_{x\in\mathcal{X}} n_x \log(\pi_x(\boldsymbol{w})), \\ &= \quad \frac{1}{2}\kappa\|\boldsymbol{w}\|_2^2 + \sum_{(x,x')\in\mathcal{X}^{\text{distinct}}} h_x q_{x,x'}(\boldsymbol{w}), \\ &\quad - \sum_{(x,x')\in\mathcal{X}^{\text{distinct}}} c_{x,x'} \log\left(q_{x,x'}(\boldsymbol{w})\right) - \sum_{x\in\mathcal{X}} n_x \log(\pi_x(\boldsymbol{w})).. \end{aligned} \tag{10}$$

In the remaining, we use the target density $\exp(-U_{\boldsymbol{z}}(\boldsymbol{w}))$ instead of $\exp(-\tilde{U}(\boldsymbol{w}))$. Appendix J shows that $\exp(-U_{\boldsymbol{z}}(\boldsymbol{w}))$ admits the distribution of interest $\exp(-\tilde{U}(\boldsymbol{w}))$ as a marginal, therefore there is no error involved in introducing the auxiliary variable $\boldsymbol{z}$. Again we introduce this auxiliary variable to make sampling more computationally tractable as shown in Section 5.

2.4 Background on BPS and LBPS

2.4.1 Basics of BPS

The BPS algorithm belongs to a type of emerging continuous-time non-reversible MCMC sampling algorithms constructed from PDMPs (Davis, 1984). It was first proposed by Peters et al. (2012), formalized, developed, and generalized by Bouchard-Côté et al. (2018); Vanetti et al. (2017).

PDMPs alternate between deterministic trajectory segment each of a length determined by the first arrival of an inhomogeneous Poisson process, interspersed with random "jumps." Three key components describe PDMPs:

The deterministic dynamics: a system of differential equations that characterize the process' deterministic behaviour between jumps.

The event rate: a non-negative function that determines the *intensity* of jumps. As the process follows its deterministic trajectory, an inhomogenous Poisson process is defined by composing the state with the intensity function. The first arrival in this Poisson process determines the length of each deterministic segment.

The transition distribution: given that a jump occurs, the transition kernel is used to sample the next state given the current one.

BPS is a special case of PDMPs with certain choices of the three aforementioned components. We denote the state of the BPS algorithm by $Y = (\boldsymbol{W}, \boldsymbol{V})$, which encodes the position and velocity of the particle. Let $\zeta(\boldsymbol{w}, \boldsymbol{v}) = \zeta(\boldsymbol{v})\zeta(\boldsymbol{w})$, where $\zeta(\boldsymbol{v})$ represents the standard multivariate Gaussian distribution and $\zeta(\boldsymbol{w})$ represents the target posterior distribution of interest. The BPS is $\zeta(\boldsymbol{w}, \boldsymbol{v})$-invariant.

Equipped with this notation, we can now define the BPS algorithm as a PDMP with the following three components:

The deterministic dynamics:

$$\frac{d\boldsymbol{w}}{dt} = \boldsymbol{v}, \frac{d\boldsymbol{v}}{dt} = 0.$$

The event rate: the intensity $\lambda(\boldsymbol{w}, \boldsymbol{v}) = \max\{0, \langle \nabla U(\boldsymbol{w}), \boldsymbol{v} \rangle\}$.

The transition distribution: a Dirac centered at $(\boldsymbol{w}, T_{\mathrm{w}}(\boldsymbol{v}))$, where

$$T_{\mathrm{w}}(\boldsymbol{v}) = \boldsymbol{v} - 2\frac{\langle \nabla U(\boldsymbol{w}), \boldsymbol{v}\rangle}{\|\nabla U(\boldsymbol{w})\|^2} \nabla U(\boldsymbol{w}).$$

To ensure the ergodicity of the Markov chain, "refreshment events" are also introduced. We use independent refreshment, where the velocity is sampled according to $\zeta(\boldsymbol{v})$ (see Vanetti et al. (2017) for alternatives). Refreshment events occur at times specified by independent and identically distributed exponential random variables with rate τ_{ref}.

We summarize the BPS sampler in Algorithm 1.

Algorithm 1 BPS algorithm

```
1: Initialization:
   Initialize the particle position and velocity $\left(\boldsymbol{w}^{(0)}, \boldsymbol{v}^{(0)}\right)$ arbitrarily on $\mathbb{R}^d \times \mathbb{R}^d$.
   Set $T$ to a certain fixed trajectory length.
2: for $i = 1, 2, \ldots,$ do
3:     Sample the first arrival times $\tau_{\mathrm{ref}}$ and $\tau_{\mathrm{bounce}}$ of a PP with intensity $\lambda^{\mathrm{ref}}$ and $\lambda^{\mathrm{bounce}}$ respectively, where $\lambda^{\mathrm{bounce}} = \max\left(\langle \boldsymbol{v}^{(i-1)}, \nabla U(\boldsymbol{w}^{(i-1)} + \boldsymbol{v}^{(i-1)} t)\rangle, 0\right)$ and the value of $\lambda^{\mathrm{ref}}$ is pre-fixed.
4:     Set $\tau_i \leftarrow \min(\tau_{\mathrm{bounce}}, \tau_{\mathrm{ref}})$.
5:     Update the position of the particle via $\boldsymbol{w}^{(i)} \leftarrow \boldsymbol{w}^{(i-1)} + \boldsymbol{v}^{(i-1)}\tau_i$.
6:     If $\tau_i = \tau_{\mathrm{ref}}$, sample the next velocity $\boldsymbol{v}^{(i)} \sim \mathcal{N}(0_d, I_d)$.
7:     If $\tau_i = \tau_{\mathrm{bounce}}$, obtain the next velocity $\boldsymbol{v}^{(i)}$ by applying the transition function $T_{\mathrm{w}^{(i)}}\left(\boldsymbol{v}^{(\boldsymbol{i-1})}\right)$.
8:     If $t_i = \sum_{j=1}^{i} \tau_j \geqslant T$, exit For Loop (line 2).
9: end for
```

2.4.2 Basics of LBPS

If the target posterior density $\zeta(\boldsymbol{w})$ can be represented as a product of positive factors in Equation 11,

$$\zeta(\boldsymbol{w}) \propto \prod_{f \in \mathcal{F}} \gamma_f(N_f), \tag{11}$$

where $\mathcal{F}$ is the set of index for all factors in the target density and N_f represents the subset of variables connected to factor f, then a "local" version of BPS referred to as LBPS can be computationally efficient by taking advantage of the structural properties of the target density, especially if the target density has a *strong sparsity* property described in definition 2. When LBPS is used, computationally cheaper refreshment scheme such as "local refreshment" (Bouchard-Côté et al., 2018) is often used by exploiting the structure of the factor graph. In the local refreshment scheme, one factor γ_f is chosen uniformly at random first and only the components of $\boldsymbol{v}$ for variables in N_f are resampled. The candidate collision time is recomputed only for the extended neighbour factors $\gamma_{f'}$ of γ_f, where $N_{f'} \cap N_f \neq \emptyset$.

Now we present a brief description of LBPS algorithm and we will see how sparsity plays an important role in improving the computational efficiency. Detailed description about an efficient implementation of LBPS via the priority queue can be found in Bouchard-Côté et al. (2018). If the target unnormalized posterior density can be factorized according to Equation 11, its associated energy function is

$$U(\boldsymbol{w}) = \sum_{f \in \mathcal{F}} U_f(N_f), \tag{12}$$

where $U_f(N_f) = -\log(\gamma_f(N_f))$. We define the local intensity $\lambda_f(\boldsymbol{w}, \boldsymbol{v})$ and local transition function $T^f_{\mathrm{w}}(\boldsymbol{v})$ for factor γ_f as:

$$\lambda_f(\boldsymbol{w}, \boldsymbol{v}) = \max\{0, \langle \nabla U_f(\boldsymbol{w}), \boldsymbol{v} \rangle\}, \tag{13}$$

$$T^f_{\mathrm{w}}(\boldsymbol{v}) = \boldsymbol{v} - 2\frac{\langle \nabla U_f(\boldsymbol{w}), \boldsymbol{v} \rangle}{\|\nabla U_f(\boldsymbol{w})\|^2} \nabla U_f(\boldsymbol{w}). \tag{14}$$

Note that for variables that are not the neighbour variables for factor γ_f, $T^f_{\mathrm{w}}(\boldsymbol{v})_k = \boldsymbol{v}_k$. In LBPS, the next collision time is the first arrival time of a PP with intensity $\lambda(\boldsymbol{w}, \boldsymbol{v}) = \sum_{f \in \mathcal{F}} \lambda_f(\boldsymbol{w}, \boldsymbol{v})$. We can sample the first arrival time using the methods described in Bouchard-Côté et al. (2018) (see Section 5.2 for specific examples for the models explored in this paper). We sample τ_f for each factor γ_f from a PP with intensity $\lambda_f(\boldsymbol{w}, \boldsymbol{v}) = \max\{0, \langle \nabla U_f(\boldsymbol{w}), \boldsymbol{v} \rangle\}$. The first arrival time for PP with intensity $\lambda(\boldsymbol{w}, \boldsymbol{v})$ is $\tau = \min_{f \in \mathcal{F}} \tau_f$. Once a bounce event takes place, LBPS only needs to manipulate a subset of the variables and factors. To be specific, if we denote f^* as the factor index such that $\tau_{f^*} = \min_{f \in \mathcal{F}} \tau_f$, we use γ_{f^*} to denote the collision factor. Following the priority queue implementation described in Bouchard-Côté et al. (2018), LBPS will update the position of neighbour variables for the collision factor γ_{f^*} and then apply the corresponding local transition function $T^{f^*}_{\mathrm{w}}(\boldsymbol{v})$ to update the velocity. Next, the algorithm will sample the candidate bounce time of the next event for all the extended neighbour factors $\gamma_{f'}$ for factor γ_{f^*} such that $N_{f'} \cap N_{f^*} \neq \emptyset$. If strong sparsity (defined in Section 4.1) is satisfied for a family of factor graphs, then the number of neighbour variables for the collision factor grows much slower than the dimension of the parameter space and is negligible compared to the dimension of the parameter space. The number of operations needed to update the position of the neighbour variables is negligible. When computing the candidate collision time for the extended neighbour factors, the number of factors involved is also negligible compared to the total number of factors in the factor graph, which is desirable in LBPS. We need to point out that although LBPS manipulates a subset of variables and factors, each local bounce will lead to changes in all the variables instead of just the neighbour variables connected with the current collision factor.

3. Proposed Bayesian GLM chain GTR model

In Section 2.2, we introduced a function $\eta(\{x, x'\})$ which embeds state transitions into integers. We will call this η an "ordering function". Here we use such ordering function to model the exchangeable rates between each pair of states through the current pair and its nearest neighbour pair. We would like to find an order of each pair of distinctive states that is biologically meaningful so that parameters that are neighbours in this ordering can share statistical strength. Take protein evolution as an example. Most frequent amino acid exchanges take place between residues with similar physicochemical properties. Under the Bayesian GLM chain GTR model we proposed below, we assume that mutations between two pairs of amino acids sharing one common state are expected to have similar exchangeable rates if the unshared distinctive states between the two pairs have similar physicochemical properties. We would like to find an ordering $\eta(\{x, x'\})$ such that amino acid pairs that are nearest neighbours share the most similar physicochemical properties given one common state. The Nearest Neighbour Pairwise Amino Acid Ordering (NNPAAO) Algorithm (described in Appendix F) proposed in Section 7 provides an ordering of amino acid pairs that satisfies such property. We denote it as $\eta_{\text{dist}}(\{x, x'\})$ since the order is determined according to the Euclidean distance between amino acids defined in Equation 18. Given this ordering, neighbour pairs share more biological similarities than non-neighbour pairs.

In this paper, based on the intuition of $\eta_{\text{dist}}(\{x, x'\})$, we define a novel Bayesian GLM **chain** GTR model with feature function

$$\boldsymbol{\phi}_i^{\text{chain}}(\{x, x'\}) := \mathbf{1}(\eta(\{x, x'\}) \in \{i, i+1\}), \tag{15}$$

where $i = 1, 2, \ldots, |\mathcal{X}|(|\mathcal{X}| - 1)/2$. If we set $\eta((x, x')) = \eta_{\text{dist}}(\{x, x'\})$, the intuition of the Bayesian GLM chain GTR model is that the value of each exchangeable parameter $\theta_{\{x,x'\}}$ depend on $\{x, x'\}$ and its neighbour pair denoted as $\{x'', x'''\}$. Its neighbour pair satisfies that $|\eta(\{x'', x'''\}) - \eta(\{x, x'\})| = 1$. Equivalently, we can define the Bayesian chain GTR model through the exchangeable parameters $\theta_{\{x,x'\}}(\boldsymbol{w}^b)$ by plugging in the definition of $\boldsymbol{\phi}_i^{\text{chain}}(\{x, x'\})$ in Equation 15:

$$\begin{aligned}\theta_{\{x,x'\}}(\boldsymbol{w}^b) &= \exp\left\{\langle \boldsymbol{w}^b, \boldsymbol{\phi}^{\text{chain}}(\{x, x'\})\rangle\right\} \\ &= \begin{cases}\exp\left(\boldsymbol{w}^b_{\eta(\{x,x'\})} + \boldsymbol{w}^b_{\eta(\{x,x'\})-1}\right), & \text{if } \eta(\{x, x'\}) = 2, 3, \ldots, (|\mathcal{X}|(|\mathcal{X}| - 1)/2), \\ \exp\left(\boldsymbol{w}^b_{\eta(\{x,x'\})}\right), & \text{if } \eta(\{x, x'\}) = 1.\end{cases}\end{aligned}$$

For simplicity, we focus on the simple example of chains, but this model could be generalized to other sparse graphs. We provide a general characterization of the running time analysis for both arbitrary factor graphs and factor graphs with sparse structure in Section 4.2. The chain GTR model is general in the sense that it is able to represent any rate matrices. We prove this in Appendix A.

4. Characterization of factor graphs where LBPS is efficient

4.1 Strong sparsity of factor graphs

A factor graph is a bipartite graph used to represent the factorization of a function, often a density function (as in Bouchard-Côté et al. (2018)), or a conditional density function (as in Section 5.1). We will use factor graphs to define a non-standard notion of sparsity, *strong sparsity*, that implies computational efficiency guarantees for the LBPS algorithm.

We start by formalizing the notion of factor graph and some associated definitions:

Definition 1 *Given a factorization of a function*

$$f(\boldsymbol{w}) = \prod_{f=1}^{m} \gamma_f(N_f),$$

*where $N_f \subset \{w_1, w_2, \dots, w_p\}$ denotes **neighbour variables** for factor γ_f, the **factor graph** $G = (\boldsymbol{w}, \Gamma, E)$ consists of a set of variables $\boldsymbol{w} = \{w_1, w_2, \dots, w_p\}$, a set of factors $\Gamma = \{\gamma_1, \gamma_2, \dots, \gamma_m\}$ and a set of edges E defined as follows. We place an undirected edge between factor γ_f and variable w_k if and only if $w_k \in N_f$. The **neighbour factors** for variable w_k is $S_k \coloneqq \{\gamma_f : w_k \in N_f\}$. The **extended neighbour variables** for factor γ_f is $\overline{N}_f \coloneqq \{w_k : w_k \in N_{f'}$ such that $N_{f'} \cap N_f \neq \emptyset\}$. The **extended neighbour factors** for factor γ_f is $\overline{S}_f \coloneqq \{\gamma_{f'} : N_{f'} \cap N_f \neq \emptyset\}$.*

In Figure 2, we use the following factor graph to illustrate definition 1:

$$f(\boldsymbol{w}) = \gamma_1(w_1)\gamma_2(w_1, w_2)\gamma_3(w_2)\gamma_4(w_2, w_3)\gamma_5(w_3, w_4).$$

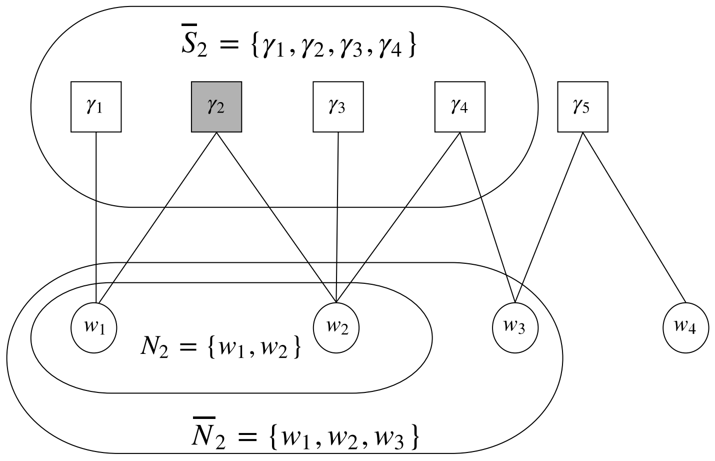
$\overline{S}_2 = \{\gamma_1, \gamma_2, \gamma_3, \gamma_4\}$
γ_1
γ_2
γ_3
γ_4
γ_5
w_1
w_2
w_3
w_4
$N_2 = \{w_1, w_2\}$
$\overline{N}_2 = \{w_1, w_2, w_3\}$

Figure 2: For factor γ_2, its neighbour variables $N_2 = \{w_1, w_2\}$. Its extended neighbour variables $\overline{N}_2 = \{w_1, w_2, w_3\}$. Its extended neighbour factors $\overline{S}_2 = \{\gamma_1, \gamma_2, \gamma_3, \gamma_4\}$.

Next, we define the notion of **strong sparsity**, which applies to factor graph families in which the number of factors m and variables p goes to infinity.

Definition 2 *A family of factor graphs is* ***strongly sparse*** *if* $\max_f |\overline{S}_f| = o(m)$, *for* $1 \leqslant k \leqslant p$, *and* $\max_f |\overline{N}_f| = o(p)$, *for* $1 \leqslant f \leqslant m$.

Strong sparsity ensures that for any factor in the factor graph, the cardinality of its extended neighbour factors and extended neighbour variables grows asymptotically slower than the dimension of the parameter space and is ultimately negligible compared to the dimension of the parameter space and the total number of factors in the factor graph. We will show later that the factor graph under our proposed CTMC GLM model sampling scheme satisfies the strong sparsity property.

4.2 Running time analysis of LBPS for factor graphs

Recent work (Deligiannidis et al., 2020; Andrieu et al., 2018) has analyzed the scaling limits of samplers constructed from PDMPs including BPS and Zig-zag processes. However, we are not aware of existing analyses of the running time of the LBPS algorithm. Here we only partially fill this gap. Specifically, we provide an analysis of the running time for a fixed trajectory length. This leaves open the scaling property of the Effective Sample Size (ESS) for LBPS ran for a fixed trajectory length. Previous work has shown that the ESS of the BPS algorithm is proportional to the trajectory length (Deligiannidis et al., 2020), but this result is for the global BPS algorithm.

Despite this limitation, our per-iteration cost analysis already reveals an interesting running time gap between LBPS executed on strongly sparse graphs versus general factor graphs. We stress that the emphasis here is not to build a comprehensive scaling limit for the purpose of comparing LBPS to other MCMC algorithms, but rather to guide the choice of factorization to use when constructing an LBPS algorithm. Specifically, we show in Section 5.1 that by using a blocking strategy, we can increase the sparsity of LBPS's factor graph, by going from the factorization of the joint to the factorization of a conditional.

We first introduce some notations needed in the analysis. Denote $\left|\overline{N}_*\right| = \max_f \left|\overline{N}_f\right|$, $|N_*| = \max_f |N_f|$ and $\left|\overline{S}_*\right| = \max_f \left|\overline{S}_f\right|$. Since $N_f \subset \overline{N}_f$, we have $|N_f| \leqslant \left|\overline{N}_f\right|$ and $|N_*| \leqslant \left|\overline{N}_*\right|$. Denote the cost for computing the collision time for a factor γ_f as c_f and $c_* = \max_f c_f$. Let c_{U_f} denote the running time for computing $\nabla U_f(\boldsymbol{w})$ and $c_{U_*} = \max_f c_{U_f}$. Now, we analyze the running time of LBPS via a priority queue implementation (Bouchard-Côté et al., 2018) for factor graph $G = (\boldsymbol{w}, \Gamma, E)$.

1. Compute the collision time for all factors and build a priority queue: $\mathcal{O}(mc_* + m \log m)$.
2. (a) If a collision takes place,
    - i. Update the position of extended neighbour variables: $\mathcal{O}(\left|\overline{N}_*\right|)$.
    - ii. Update the velocity of neighbour variables according to Equation 14: $\mathcal{O}(|\overline{N}_*| + c_{U_*})$.

iii. Add new samples to the trajectory list L_k: $\mathcal{O}(|N_*|)$. We use L_k to denote a list of triplets $\left(w_k^{(i)}, v_k^{(i)}, t_k^{(i)}\right)$, with $w_k^{(0)}$ and $v_k^{(0)}$ representing the initial position, and velocity and $t_k^{(0)} = 0$. For $i > 0$, $v_k^{(i)}$ represents the velocity after the ith collision or refreshment event and $t_k^{(i)}$ represents the time for the ith event.

iv. Compute the collision time for extended neighbour factors: $\mathcal{O}\left(\left|\overline{S}_*\right| c_*\right)$.

v. Insert the extended neighbour factors and its collision time into the priority queue: $\mathcal{O}(\log m)$. The elements of the priority queue are stored according to an increasing order of the collision time.

(b) If a refreshment event takes place and a local refreshment scheme (described in Section 2.4.2) is used:

i. Pick a factor uniformly at random and refresh the velocity: $\mathcal{O}(|N_*|)$.

ii. Update the position of the neighbour variables of the selected factor, compute the collision time for the extended neighbour factors and update the collision time in the priority queue: $\mathcal{O}(|\overline{N}_*| + \left|\overline{S}_*\right| c_* + \left|\overline{S}_*\right| \log m)$.

iii. Sample the next refreshment time $\mathcal{O}(1)$.

Thus, assuming there are J_1 collision events and J_2 refreshment events when the particle travels along a fixed trajectory length, the total running time for LBPS is:

$$\mathcal{O}\Bigg(mc_* + J_1\left(\left|\overline{N}_*\right| + c_{U_*} + \left|\overline{S}_*\right| c_* + \log m\right) + J_2(\left|\overline{N}_*\right| + \left|\overline{S}_*\right| c_* + \left|\overline{S}_*\right| \log m)\Bigg).$$

For a factor graph with no sparsity, the total running time for LBPS is

$$\mathcal{O}\Bigg(J_1\left(p + c_{U_*} + mc_* + \log m\right) + J_2(p + mc_* + m\log m)\Bigg).$$

For a factor graph with strong sparsity, the total running time for LBPS is

$$\mathcal{O}\Bigg(mc_* + J_1\left(p^{\alpha_1} + c_{U_*} + m^{\alpha_2}c_* + \log m\right) + J_2(p^{\alpha_1} + m^{\alpha_2}c_* + m^{\alpha_2}\log m)\Bigg), \tag{16}$$

where $0 \leqslant \alpha_1 < 1, 0 \leqslant \alpha_2 < 1$.

If m and J_1 are the dominant terms, as typical in applications, this yields a reduction in per-iteration cost from $\mathcal{O}(J_1 m)$ down to $\mathcal{O}(J_1 m^{\alpha_2})$ when moving from a general factor graph to a strongly sparse one.

5. Methodology

5.1 Achieving strong sparsity through LBPS-HMC alternation

In this section, we first show that applying LBPS naively to the CTMC GLM models introduced in Section 2.2 would lead to a factor graph which does not satisfy strong sparsity, and hence to an inefficient LBPS algorithm. To address this issue, we will introduce a sampling strategy that alternates between two moves: one which samples a subset of variables based on LBPS, while conditioning on the rest, followed by one move which samples a small set of more connected variables based on HMC while conditioning on the rest.

To motivate the LBPS-HMC alternation algorithm, let us return to the problem of posterior inference of the parameters of CTMC GLM models. In any such models, we can decompose the augmented negative log density $U(\boldsymbol{w})$ (Equation 10) into a sum of factors. We first introduce notations to group these factors into the following categories:

Normal factor: $\frac{1}{2}\kappa\|\boldsymbol{w}\|_2^2$,

Sojourn time factor: $H_{x,x'} := h_x q_{x,x'}(\boldsymbol{w})$, for all $(x,x') \in \mathcal{X}^{\text{distinct}}$,

Transition count factor: $C_{x,x'} := -c_{x,x'} \log\left(q_{x,x'}(\boldsymbol{w})\right)$, for all $(x,x') \in \mathcal{X}^{\text{distinct}}$,

Initial count factor: $\boldsymbol{\pi}_x := -n_x \log(\pi_x(\boldsymbol{w}))$, for all $x \in \mathcal{X}$, with $\boldsymbol{\pi}_x$ representing the initial count factor but $\pi_x(\boldsymbol{w})$ denotes the stationary distribution for state x given $\boldsymbol{w}$.

Consider first the GTR version of this model. The factor graph of the full posterior distribution is shown in Figure 3. It is the sparsest possible CTMC GLM model, in the sense that there is no parameter sharing among the exchangeable nor stationary parameters, yet, it can be readily seen that this graph is not strongly sparse. Indeed, this graph has the property that for any factor γ, the extended neighbourhood of γ includes all other factors since they all share common neighbour variables $\boldsymbol{w}^u$. Hence the graph is not strongly sparse and LBPS does not enjoy the computational savings described in Section 4.2.

To address this issue, instead of using LBPS on the full posterior distribution, we only apply LBPS to a subset of the parameters while conditioning on the others. Within one LBPS round, only a subset of the variables are updated while keeping the other ones fixed. This strategy can be described as an "LBPS-within-Gibbs" method, in parallel to the terminology "Metropolis-within-Gibbs" (see e.g. Diggle et al. (1998)). We still maintain invariance with respect to the correct distribution since we alternate between LBPS and an HMC move sampling the variables not updated by LBPS. As we will show shortly, in the context of CTMC GLM models, it is advantageous to fix the parameters $\boldsymbol{w}^u$ during the LBPS phase of the alternation. The proposed sampling scheme is summarized in Algorithm 2, and its invariance is established formally in Appendix J.

When performing LBPS on a subset of the latent variables, the variables that are temporarily fixed can be removed from the factor graph, as their values do not change in this segment of LBPS trajectory. The power of our LBPS-HMC alternation method comes from the fact that in certain situations we only need to fix a small number of variables in order to gain strong sparsity.

Let us illustrate this on the chain GTR model introduced in Section 3. Specifically, we will next present a concrete example where the LBPS-HMC alternation strategy achieves strong sparsity. We show in Figure 4 the factor graph corresponding to the distribution of the parameters $\boldsymbol{w}^b$ given all the other variables.

Regardless of the size of the state space $\mathcal{X}$, we next argue that for any $\{x,x'\} \in \mathcal{X}^{\text{unordered,dist.}}$, any sojourn time factor $H_{x,x'}$ or transition count $C_{x,x'}$ has at most twelve extended neighbour factors and three extended neighbour variables under Bayesian GLM chain GTR when conditioning on $\boldsymbol{w}^u$. For the pairs of states $\{x,x'\}$, when $\eta(\{x,x'\}) \neq$

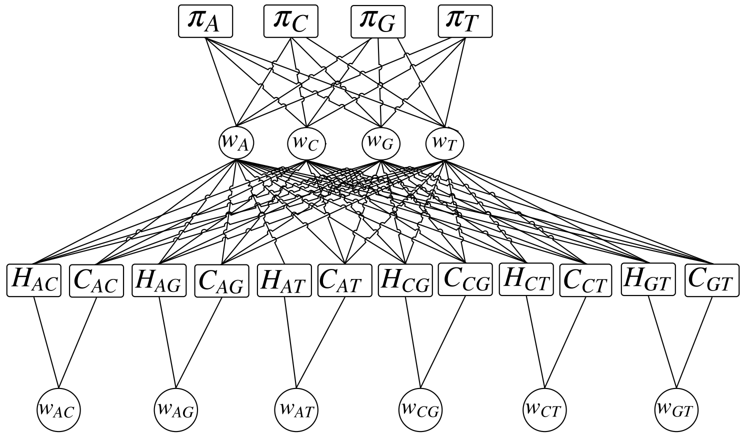

Figure 3: Factor graph corresponding to the full posterior of the parameters of the GLM representation of GTR model (shown here for DNA data). Top row shows the initial count factors, second row, stationary distribution parameters, third row, transition and sojourn time factors (C and H respectively), last row, exchangeable parameters. The graph is not strongly sparse. To make the figure simpler, we have omitted redundant factors which do not affect the strong sparsity argument (those corresponding to the normal prior, the other half of the transition count factors and sojourn time factors for the symmetric pair of states (x', x) such as $H_{CA}, C_{CA}, H_{GA}, C_{GA}, \ldots$.

$|\mathcal{X}|(|\mathcal{X}|-1)/2$, the twelve extended neighbour factors of factor $H_{x,x'}$ or $C_{x,x'}$ include the factors only connected with $\boldsymbol{w}_i^b$ and $\boldsymbol{w}_{i+1}^b$, where $i = \eta(\{x, x'\})-2, \eta(\{x, x'\})-1$, and $\eta(\{x, x'\})$. Similarly, for the pair of states $\{x, x'\}$ such that $\eta(\{x, x'\}) = |\mathcal{X}|(|\mathcal{X}|-1)/2$, $H_{x,x'}$ or $C_{x,x'}$ has eight extended neighbour factors, where each of the eight factors is only connected with $\boldsymbol{w}_i^b$ and $\boldsymbol{w}_{i+1}^b$, where $i = \eta(\{x, x'\}) - 2$ and $\eta(\{x, x'\}) - 1$. All factors have three extended neighbour variables $\boldsymbol{w}^b_{\eta(\{x,x'\})-1}, \boldsymbol{w}^b_{\eta(\{x,x'\})}$ and $\boldsymbol{w}^b_{\eta(\{x,x'\})+1}$, except that factor $H_{x'',x'''}$ or $C_{x'',x'''}$ has two extended neighbour variables $\boldsymbol{w}_1^b$ and $\boldsymbol{w}_2^b$ when $\eta(\{x'', x'''\}) = 1$.

Thus, when updating the position of extended neighbour variables or computing the candidate bounce time for extended neighbour factors of the collision factor, there is only a small constant number of extended neighbour variables and extended neighbour factors involved. We can conclude that the factor graph in Figure 4 satisfies the desirable strong sparsity in definition 2.

5.2 Simulation of bounce inter-arrival times for GLM GTR models

In this section, we show how to simulate the candidate times τ_f for the next bounce for each factor f. The simulation algorithm for a Gaussian distribution can be found in Bouchard-Côté et al. (2018), so we focus on the other factors.

Algorithm 2 Proposed sampling scheme LBPS-HMC for CTMCs

```
1: Initialization:
   Initialize weights $\boldsymbol{w}^0 = (\boldsymbol{w}^u, \boldsymbol{w}^b)$ from $N(0,1)$, where $\boldsymbol{w}^u$ represents the weights corresponding
   to the univariate features to compute the stationary distribution and $\boldsymbol{w}^b$ represents the weights
   associated with bivariate features to compute the exchangeable parameters.
2: for $i = 1, 2, \ldots, N$, do
3:     Compute the rate matrix $Q$ given $\boldsymbol{w}^{(i-1)}$ under Bayesian GLM rate matrices.
4:     Use an end-point sampler as in Zhao et al. (2016) to simulate a path given time interval $\Delta_e$
       with two consecutive observations observed at time points $e = (t, t+1)$ of the time series
       according to Equation 1 using the cached uniformization technique.
5:     Compute the aggregate sum of the sufficient statistics of all time series obtained in Step 4.
6:     Update univariate weights $\boldsymbol{w}^u$ for the stationary distribution via HMC:

       $$(\boldsymbol{w}^u)^{(i+1)} \mid \left(\boldsymbol{w}^u, \boldsymbol{w}^b\right)^{(i)}, Z^{(i)} \sim \text{HMC}(\cdot \mid \left(\boldsymbol{w}^u, \boldsymbol{w}^b\right)^{(i)}, Z^{(i)}, L, \epsilon),$$

       where $L$ and $\epsilon$ are tuning parameters representing the number of leapfrog jumps and step size
       in HMC. Recall that $Z := (N, H, C)$ (defined in Section 2.1) denotes the sufficient statistics
       of the augmented CTMC paths. Function evaluation and gradient calculation required by
       HMC are described in Equation 20 and Equation 21 in Appendix B.
7:     Update bivariate weights $\boldsymbol{w}^b$ used to compute the exchangeable parameters:

       $$\left(\boldsymbol{w}^b\right)^{(i+1)} \mid (\boldsymbol{w}^u)^{(i+1)}, \left(\boldsymbol{w}^b\right)^{(i)}, Z^{(i)} \sim \text{LBPS}(\cdot \mid \left((\boldsymbol{w}^u)^{(i+1)}, \left(\boldsymbol{w}^b\right)^{(i)}\right), Z^{(i)}, T),$$

       where $T$ is the tuning parameter in LBPS representing the fixed length of the trajectory.
8: end for
```

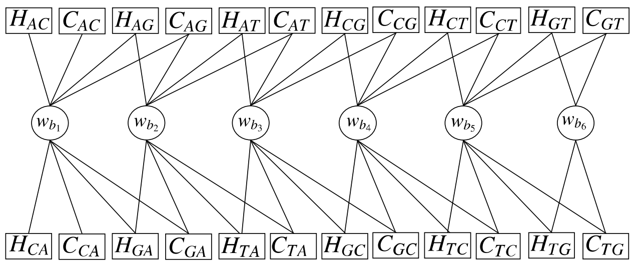

Figure 4: Factor graphs involved in the LBPS phase of our LBPS-HMC alternation algorithm applied to the chain GTR model. This is the factor graph corresponding to the distribution of the parameters $\boldsymbol{w}^b$ given all the other variables.

5.2.1 Sojourn time factors

We first sample the energy gap $-\log(E) > 0$ with $E \sim \mathcal{U}(0,1)$. This gap represents the difference between the energy denoted as E_0 at the current position of a particle and a higher energy $E_0 - \log(E)$. The goal is to determine a time interval δ such that at this time point the particle has an energy of $E_0 - \log(E)$.

For sojourn factors, the potential energy of the particle is:

$$\begin{aligned} U(\boldsymbol{w}_0) &= h_x^{(i)} q_{x,x'}^{(\text{rev})}(\boldsymbol{w}_0) \\ &= h_x^{(i)} \pi_{x'} \exp(\langle \boldsymbol{w}_0, \boldsymbol{\phi}(\{x,x'\})\rangle). \end{aligned}$$

The potential energy after a time interval δ:

$$\begin{aligned} U(\boldsymbol{w}_0 + \boldsymbol{v}\delta) &= h_x^{(i)} q_{x,x'}^{(\text{rev})}(\boldsymbol{w}_0 + \boldsymbol{v}\delta) \\ &= h_x^{(i)} \pi_{x'} \exp(\langle \boldsymbol{w}_0 + \boldsymbol{v}\delta, \boldsymbol{\phi}(\{x,x'\})\rangle). \end{aligned}$$

We observe that the particle is travelling to a higher energy area if and only if $\langle \boldsymbol{v}, \boldsymbol{\phi}(\{x,x'\})\rangle > 0$. Therefore, we set

$$\begin{aligned} -\log(E) &= U(\boldsymbol{w}_0 + \boldsymbol{v}\delta) - U(\boldsymbol{w}_0) \\ &= h_x^{(i)} q_{x,x'}^{(\text{rev})}(\boldsymbol{w}_0)\Big(\exp(\langle \boldsymbol{v}\Delta, \boldsymbol{\phi}(\{x,x'\})\rangle) - 1\Big). \end{aligned}$$

We denote $c = -\log(E) > 0$ and obtain:

$$\delta = \begin{cases} \frac{1}{\langle \boldsymbol{v}, \boldsymbol{\phi}(\{x,x'\})\rangle} \log\left(\frac{c}{h_x^{(i)} q_{x,x'}^{(\text{rev})}(\boldsymbol{w}_0)} + 1\right) & \text{if} \quad \langle \boldsymbol{v}, \boldsymbol{\phi}(\{x,x'\})\rangle > 0, \\ \infty, & \text{otherwise.} \end{cases}$$

5.2.2 Transition count factors

Similarly, the transition count factor for pairs of states (x, x') is $-c_{x,x'}^{(i)} \log\left(q_{x,x'}^{(\text{rev})}(\boldsymbol{w}_0)\right)$ in the ith iteration, and the gradient is $-c_{x,x'}^{(i)} \boldsymbol{\phi}(\{x,x'\})$. Thus, the potential energy of the particle is:

$$\begin{aligned} U(\boldsymbol{w}_0) &= -c_{x,x'}^{(i)} \log\left(q_{x,x'}^{(\text{rev})}(\boldsymbol{w}_0)\right) \\ &= -c_{x,x'}^{(i)} \Big(\log \pi_{x'} + \langle \boldsymbol{w}_0, \boldsymbol{\phi}(\{x,x'\})\rangle \Big). \end{aligned}$$

The potential energy after a time interval Δ:

$$U(\boldsymbol{w}_0 + \boldsymbol{v}\delta) = -c_{x,x'}^{(i)} \Big(\log(\pi_{x'}) + \langle \boldsymbol{w}_0 + \boldsymbol{v}\delta, \boldsymbol{\phi}(\{x,x'\})\rangle \Big).$$

We sample $E \sim \mathcal{U}(0,1)$, set $c = -\log(E)$, $U(\boldsymbol{w}_0 + \boldsymbol{v}\delta) - U(\boldsymbol{w}_0) = c$, and obtain:

$$\delta = \begin{cases} -\frac{c}{c_{x,x'}^{(i)} \langle \boldsymbol{v}, \boldsymbol{\phi}(\{x,x'\})\rangle} & \text{if} \quad \langle \boldsymbol{v}, \boldsymbol{\phi}(\{x,x'\})\rangle \leqslant 0, \\ \infty, & \text{otherwise.} \end{cases}$$

5.3 Running time analysis of LBPS-HMC

In Section 4.2, we have provided the running time analysis of LBPS under a general factor graph G. In this section, we analyze the running time of LBPS-HMC under the Bayesian GLM chain GTR model for CTMCs by simplifying the computational cost of a factor graph G with sparse property under the condition that $c_* = \mathcal{O}(1)$ and $c_{U_*} = \mathcal{O}(1)$, which is satisfied in the Bayesian GLM chain GTR model. The condition that $c_* = \mathcal{O}(1)$ and $c_{U_*} = \mathcal{O}(1)$ indicates that the computational cost of computing the collision time and the gradient in terms of $\boldsymbol{w}$ for any factor in the factor graph is of constant time regardless of the dimension of the state space $\mathcal{X}$ in the CTMC. Recall that the computational cost for a strongly sparse factor graph G is given in Equation 16.

The computational cost we derive not only holds for Bayesian GLM chain GTR model but also holds for any Bayesian GLM rate matrix model that satisfies $c_* = \mathcal{O}(1), c_{U_*} = \mathcal{O}(1), \left|\overline{N}_*\right| = \mathcal{O}(1)$ and $\left|\overline{S}_*\right| = \mathcal{O}(1)$. Under the Bayesian GLM chain GTR rate matrix parameterization, we have $m = \mathcal{O}\left(|\mathcal{X}|^2\right)$ and $p = \mathcal{O}\left(|\mathcal{X}|^2\right)$.

The running time of the key steps in Algorithm 2 is as follows:

1. Sample the auxiliary sufficient statistics using the end-point sampler: $\mathcal{O}\left(|\mathcal{X}|^3\right)$.

2. Update the univariate weights $\boldsymbol{w}^u$ using HMC: $\mathcal{O}(J_0|\mathcal{X}|)$, where J_0 represents the number of leapfrog steps in one HMC iteration. When updating $\boldsymbol{w}^u$, the number of non-zero entries for all possible features is $|\mathcal{X}|$.

3. Update the bivariate weights $\boldsymbol{w}^b$ using LBPS: $\mathcal{O}(J_1 \log|\mathcal{X}| + J_2 \log|\mathcal{X}|)$. This is obtained by plugging $\alpha_1 = \alpha_2 = 0$ into Equation 16, where $\alpha_1 = \alpha_2 = 0$ since we have $\left|\overline{N}_*\right| = \mathcal{O}(1)$ and $\left|\overline{S}_*\right| = \mathcal{O}(1)$. A local refreshment scheme is adopted.

Thus, the total cost of one iteration of our LBPS-HMC alternation algorithm is

$$\mathcal{O}(J_0|\mathcal{X}| + J_1 \log|\mathcal{X}| + J_2 \log|\mathcal{X}| + |\mathcal{X}|^3). \tag{17}$$

In comparison, one HMC iteration for Bayesian GLM chain GTR takes $\mathcal{O}\left(J_0|\mathcal{X}|^2 + |\mathcal{X}|^3\right)$ since the total number of non-zero entries for all possible features of Bayesian GLM chain GTR is $|\mathcal{X}|(|\mathcal{X}|+1)/2$ using only HMC, where J_0 is the number of leapfrog steps in one HMC iteration. In our experiments, we find that $|\mathcal{X}|$, $\log|\mathcal{X}|$ are both lower order terms compared with $|\mathcal{X}|^2$ using HMC. This potentially explains why our LBPS-HMC is more computationally efficient than HMC. The running time analysis discussed here is not a whole story since we do not provide information on how J_1 and J_2 should be scaled in order to obtain a constant number of effective samples. To provide the readers an idea of how J_0 in HMC scales, consider the case of normally distributed random vectors of dimension p with an identity covariance matrix. To reach a nearly independent point, the number of leapfrog updates under the independent and identically distributed case grows as $\mathcal{O}\left(p^{\frac{1}{4}}\right)$ (Neal et al., 2011), where p is the dimension of the parameters. For LBPS, under the independent and identically distributed normal distribution with dimension p or weakly dependent case, we expect J_1 to grow as $O(p)$.

6. Experiments

All numerical experiments are run on Compute Canada resources "cedar" with Intel "Broadwell" CPUs at 2.1Ghz and an E5-2683 v4 model. A detailed description of "cedar" can be found at `https://docs.computecanada.ca/wiki/Cedar`. The algorithm is implemented in Python 3.6.4 and available at `https://github.com/zhaottcrystal/rejfreePy_main`. Across all experiments, the ESS is evaluated via the R package *mcmcse* (Flegal et al., 2012). We set the first 30% of the posterior samples as the burnin period to be discarded.

All the synthetic datasets are simulated under a Bayesian GLM chain GTR rate matrix parameterization. We assign a standard Gaussian distribution as the prior over both the univariate and bivariate weights, which are both generated from $\mathcal{U}(0, 1)$ across all simulation studies. The refreshment rate of LBPS is set to one and the number of leapfrog jumps and the stepsize of each jump are set as $L = 40$ and $\epsilon = 0.001$ in HMC. The combination of the two tuning parameters are simply chosen among the ones with the best computational efficiency given multiple combination of the parameter values.

Across all synthetic experiments, we generate 500 sequences given a total length of the observation time interval as three and the states of all sequences are observed every 0.5 unit time under a Bayesian GLM chain GTR with different dimensions of rate matrices. Later in this paper, we use "LBPS" for short to represent the combined sampling scheme LBPS-HMC, which uses HMC to sample the univariate weights and LBPS to sample the bivariate weights. When HMC sampling scheme is mentioned later, it refers to the sampling scheme using HMC to sample all the weight parameters $\boldsymbol{w} = \left(\boldsymbol{w}^u, \boldsymbol{w}^b\right)$.

In both synthetic data experiments in Section 6 and real data analysis in Section 7, we conduct computational efficiency comparison via ESS per second and we also check the correctness of the two sampler implementation. For the synthetic data experiments, we perform Exact Invariance Test (EIT) (Bouchard-Côté et al., 2019) in the spirit of Geweke (2004). We compare the density of the posterior samples between LBPS and HMC, and evaluate their difference using Absolute Relative Difference (ARD) (defined in Appendix I). Since both algorithms share the same limiting distribution, as we increase the number of iterations of the Markov chains, after an initial burnin period, the distributions of the posterior samples from the two algorithms are expected to be closer to the target posterior distribution. For higher dimensional rate matrices, since the total number of parameters is $\mathcal{O}(|\mathcal{X}|^2)$, it is hard to display the density plots for all parameters. We use ARD as the metric to describe the similarities between posterior samples from the two algorithms. Smaller values of ARD indicate more similarities. In Section 7 with the real dataset, we also compute the ARD and report the results in Appendix I.

6.1 Correctness check of LBPS and HMC using EIT

To check the correctness of our software implementation, we perform out testing procedure EIT, a simple extension of Geweke (2004), with details provided in Appendix D. We performed tests on both LBPS and HMC kernels on various dimensions of the rate matrices (see Appendix D).

To further validate the correctness of LBPS and HMC, we compare the distribution of the posterior samples collected from LBPS and HMC respectively. The shared synthetic dataset is generated under an 8-by-8 Bayesian GLM chain GTR. The prior distribution for

$\boldsymbol{w}$ is standard Gaussian distribution. We obtain 40,000 and 10,000 posterior samples from LBPS and HMC separately with first 30% of the samples discarded. With a long chain from both algorithms, we expect their density plots to be close and we demonstrate this in Figure 5 using the density plot of all exchangeable parameters. The boxplot is provided in Figure 9 in Appendix E. We also provide summary statistics of the posterior samples among different parameters.

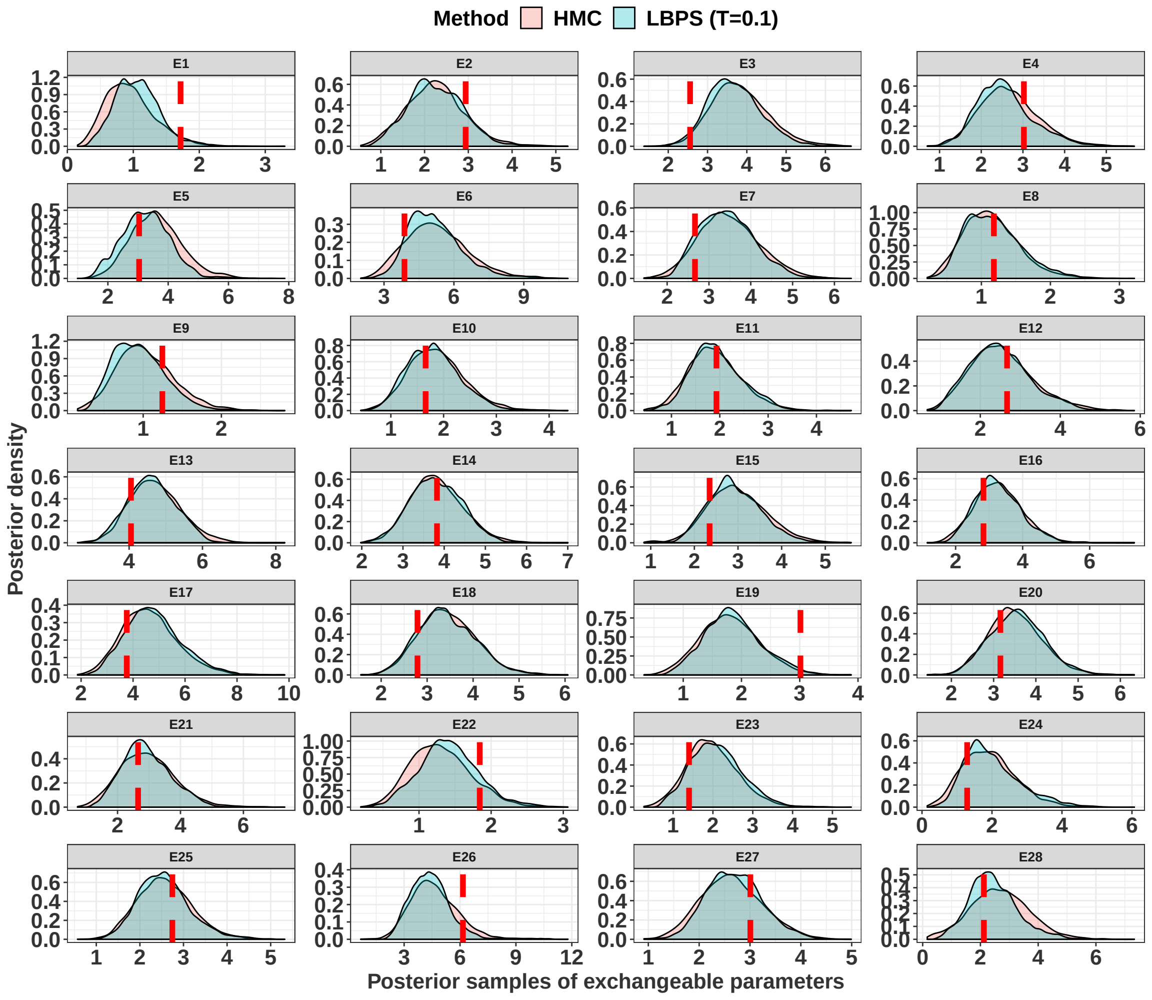

Figure 5: Density plot of posterior density comparison across exchangeable parameters of an 8-by-8 rate matrix between LBPS (with trajectory length 0.1) and HMC.

6.2 Computational efficiency comparison between LBPS and HMC

In order to explore the scalability of our algorithm as the dimension of the parameters increases, we compared the ESS per second in the $\log_{10}$ scale among all the parameters for different sizes of the rate matrices. Larger ESS per second indicates better computational

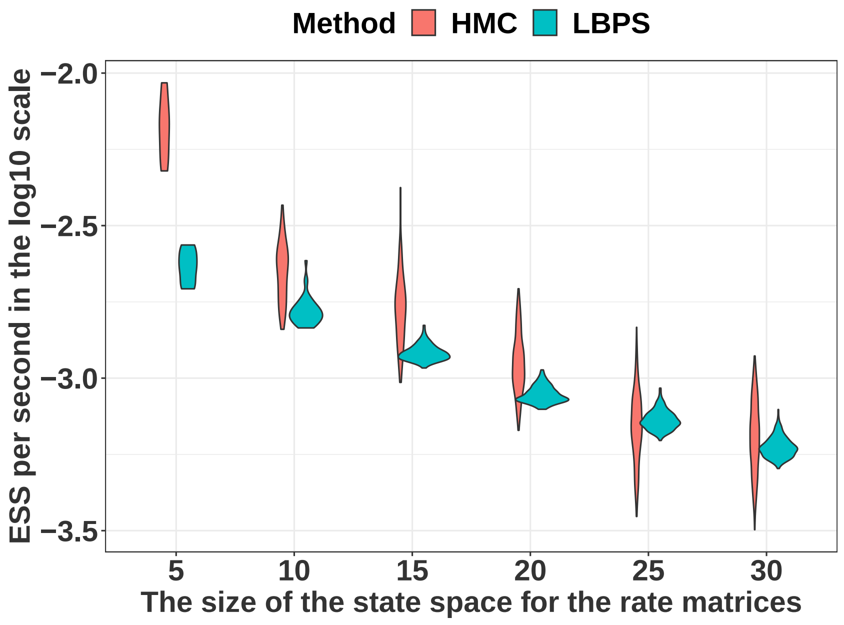

Figure 6: ESS per second in the $\log_{10}$ with the dimension of the rate matrices ranging from 5-by-5, 10-by-10, ... to 30-by-30 with a step size of 5.

efficiency of the algorithm. One obvious choice to compare the efficiency of MCMC algorithms is the average MCMC efficiency over all parameters, which is represented by the mean ESS per second. However, the algorithm efficiency is better represented by the minimum ESS per second since the entire posterior samples are valid unless all parameters have mixed adequately. Thus, we focus on the minimum ESS per second over all parameters.

The dimension of the rate matrices ranges from 5-by-5, 10-by-10, ... to 30-by-30. We obtain a total number of 60,000 posterior samples from LBPS and 10,000 samples from HMC. To speed up the actual running time of the experiments, for rate matrices of dimension lower than 15-by-15, the trajectory length is fixed as 0.1. For rate matrices with higher dimensions, the trajectory length is shortened to 0.05. Under the Bayesian GLM chain GTR model parameterization, the number of exchangeable parameters is $|\mathcal{X}|(|\mathcal{X}|-1)/2$. The result is displayed in Figure 6.

From Figure 6, we can see that when the dimension of the weight parameters is low, for a 5-by-5 rate matrix, HMC has better performance than LBPS. As the dimension increases, it shows that the minimum of the ESS per second for LBPS outperforms HMC. The larger variation in ESS per second of HMC in the violin plot in Figure 6 indicates that HMC is not efficient in exploring certain directions of the parameter space. We find that the minimum ESS per second and its lower quantiles for LBPS are better than HMC as the dimension increases. This result suggests that LBPS has better computational efficiency than HMC for the class of high-dimensional CTMCs considered in this section.

6.3 ESS per second using different trajectory lengths of LBPS comparisons

The computation efficiency of HMC has been found to depend highly on the choices of the tuning parameters, which are the number of leapfrog jumps L and the size of each jump ϵ. Various strategies (Wang et al., 2013; Hoffman and Gelman, 2014; Zhao et al., 2016) have been developed to effectively tune the parameters. LBPS also involves a tuning parameter, which is the fixed trajectory length. Thus, it is of interest to examine whether the computational efficiency of LBPS is sensitive to different choices of the trajectory length.

We simulate a synthetic dataset from a 20-by-20 rate matrix with 190 exchangeable parameters. A 20-by-20 rate matrix is chosen since it has the same dimension as protein evolution in the real data analysis. We use ESS per second as the metric to evaluate the computation efficiency of LBPS. Our summary statistics include the minimum, first quantile, mean, median, third quantile and maximum of the ESS per second across 190 exchangeable parameters. We obtain 40,000 posterior samples of the parameters of interest. We show the actual walltime (in seconds) of our algorithm with fixed trajectory lengths at 0.025, 0.1, 0.15, 0.2 and 0.25 in Table 1. Since the first 30% of the samples are discarded, the actual running time used to compute the ESS per second are scaled by 70% of the total walltime in Table 1.

Trajectory Lengths	0.025	0.10	0.15	0.20	0.25
Walltime (in seconds)	240677	282233	338242	381565	403932

Table 1: Actual walltime of LBPS for 40,000 iterations with trajectory lengths at 0.025, 0.1, 0.15, 0.2, 0.25.

The results are shown in Figure 7. We have found that except the maximum of the ESS per second, the computational efficiency of LBPS is similar across different trajectory lengths and especially the minimum of ESS per second is very robust. The longer the trajectory lengths, the better performance of the maximum of the ESS per second. Similar conclusions are achieved in our real data analysis. The minimum of the ESS per second is more important than the maximum since the performance of a sampler depends on the direction that is the hardest to sample.

7. Real Data Analysis

7.1 Background

We use the real dataset from Zhao et al. (2016) with 641 amino acid sequences and each sequence has 415 sites from the protein kinase domain family. It is available at `https://github.com/zhaottcrystal/rejfreePy_main/tree/master/Dataset`. In phylogenetics, the evolutionary process is often inferred using multiple homologous biological sequences under a evolutionary tree with the same rate matrix across the tree. For simplicity, we estimate the rate matrix from a pair of sequences. We pick randomly a pair of amino acid sequences to study the rate matrix from its posterior distribution.

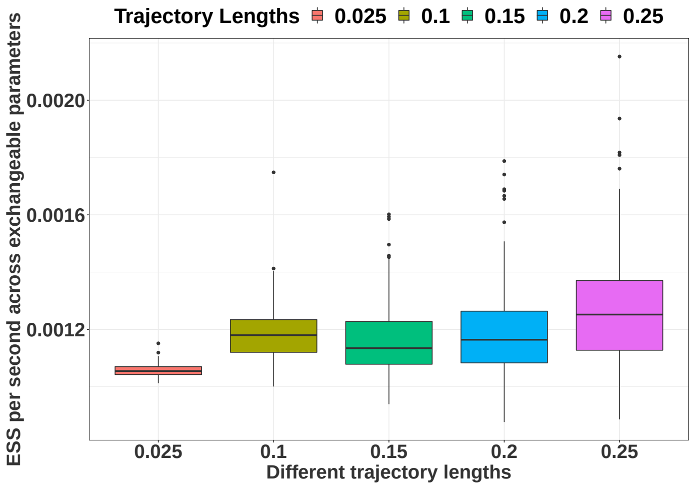

Figure 7: ESS per second of the summary statistics across exchangeable parameters with respect to different trajectory lengths of LBPS.

In Section 2.2, we have provided intuition behind the chain GTR model since we assume that the exchangeable rates between a pair of states is affected by its neighbour pairs with similar biological properties. Then another question arises as how to pick a reasonable ordering of pairs of amino acids to allow neighbour pairs share similar biological properties. According to He et al. (2011), we determine the ordering according to the closeness among amino acids based on their pairwise Euclidean distance defined by their physiochemical properties. The distance satisfies nonnegative, reflective, symmetric, and triangle properties and is given by Grantham (1974) as :

$$D_{ij} = \Big(\alpha(c_i - c_j)^2 + \beta(p_i - p_j)^2 + \gamma(v_i - v_j)^2\Big)^{\frac{1}{2}}, \tag{18}$$

where c = composition, p = polarity, v = molecular volume, and α, β, γ are defined in Grantham (1974). Larger pairwise distance indicates less similarity between pairs of amino acids, where mutation between them may be deteriorative. On the contrary, smaller distance indicates more similarities and easier mutations. The pairwise Euclidian distance between amino acids is given in Table 3 by He et al. (2011). We provide the table in Appendix F.

Given the distance in Appendix F, the ordering of pairs of amino acids regarding their closeness is determined according to our proposed NNPAAO Algorithm 3 in Appendix G. Some notations are introduced to help understand the algorithm. Denote AminoAcidDist as the pairwise Euclidean distance between amino acids shown in Appendix F, $\mathcal{X}$ as the state space of 20 amino acids and $\mathcal{X}^{\text{unordered,dist.}}$ as the set of unordered pairs of distinct amino acids, where $|\chi| = 20$ and $\left|\mathcal{X}^{\text{unordered,dist.}}\right| = 190$. The algorithm outputs a dictionary AminoAcidsPairRank with keys from $\mathcal{X}^{\text{unordered,dist.}}$ and the value associated with each key represents the rank of this amino acid pair. Neighbour pairs indicate closeness for similarities.

The algorithm first picks the amino acid pair with the smallest positive Euclidean distance, which is pair "IL", where D(I, L) = 5. The nearest neighbour to "IL" is chosen among two candidate pairs with the smallest nonzero distance to "I" or "L", which are "IM" with D(I, M)=10 or "LM" with D(L, M)=15. Among them, we pick the pair with a smaller distance and "IM" is defined as the nearest neighbour to "IL". Next our current pair is set as "IM" and similarly, we find the nearest neighbour to "IM" and we keep searching the nearest neighbour for the current pair until we have iterated over all pairs in $\mathcal{X}^{\text{unordered,dist.}}$.

7.2 Numerical results: computational efficiency comparison

We provide the correctness check via ARD between the two samplers in Appendix I. We compare the computational efficiency of LBPS and HMC via comparing the summary statistics of the ESS per second across 190 exchangeable parameters of a 20-by-20 reversible rate matrix using the protein kinase domain family dataset in Zhao et al. (2016). Figure 8 demonstrates the computational advantages of using LBPS over HMC. The trajectory lengths of LBPS is chosen as 0.1, 0.15 and 0.2. We find that when the trajectory length is set at 0.2, LBPS outperforms HMC across all summary statistics of ESS per second. The minimum of ESS per second of HMC is markedly worse than LBPS, this implies that the inefficiency of HMC to explore certain direction of the parameter space. Under a fixed trajectory length of LBPS, there is no big difference between the first quantile and third quantile of the ESS per second across all parameters. It indicates that LBPS has similar computational efficiency across different directions of the parameter space. In Table 2, we provide the ESS runing HMC for 811380 seconds (9.4 days) compared to LBPS running only 81.25% of the walltime of HMC, which further demonstrates the superior performance of LBPS compared to HMC. A traceplot scaled by running time of a selected exchangeable parameter showing better mixing using LBPS compared to HMC is provided in Figure 10 in Appendix H.

	Summary Statistics					
Method	Min.	1st Quantile	Median	Mean	3rd Quantile	Max.
LBPS (0.2)	773	1656	1864	1845	2108	2532
HMC	95	626	740	736	840	1233

Table 2: Summary of ESS across exchangeable parameters between HMC and LBPS with trajectory length 0.2.

8. Discussion

In this paper, we have developed a computationally efficient sampling scheme combining LBPS and HMC to explore high dimensional CTMCs under a novel Bayesian GLM chain GTR rate matrix. In this model, we assume that the mutation rates of the amino acid evolutionary processes depend on its neighbour pairs with similar physiochemical properties.

We also provide a framework for assessing the running time of LBPS algorithm, based on a notion of *strong sparsity*. In terms of empirical performance, we found that in the

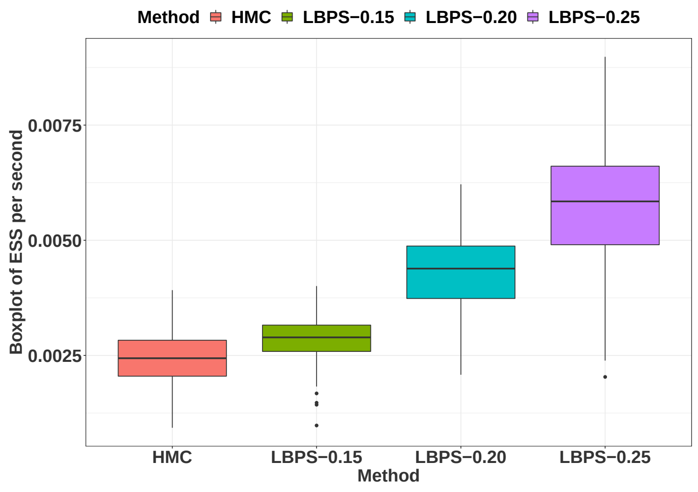

Figure 8: Summary statistics of ESS per second of across exchangeable parameters with different trajectory lengths of LBPS at 0.1, 0.15 and 0.2, compared to HMC.

problems considered, HMC alone leads to inefficient mixing of certain directions of the parameter space. In the contrary, the computational performance of our method is more uniform across different directions of the parameter space. In the real application, our method outperforms stand-alone HMC in terms of ESS per second for all chosen summary statistics.

Moreover, we provide the first proof-of-concept real data application applying LBPS to high dimensional CTMCs, showing the great potential of this computational method in this domain. This shows promises for evolutionary processes with a large state space, such as codon evolution (Anisimova and Kosiol, 2008). Our sampling algorithm also has the potential to be applied to co-evolution of groups of interacting amino acids residues which also involves large rate matrices.

For phylogenetic applications, one of the limitations of our algorithm in its current form is that it assumes an unnormalized reversible rate matrices since we were only able to obtain analytical solution to the collision time under this situation. It is customary in phylogenetics to use normalized rate matrix, where the normalization ensures that the expected number of mutations is one given one unit time. To resolve this issue, one can normalize the branch lengths instead of the rate matrix to recover identifiability. A related issue is that changing the prior on the weights requires that the algorithms used to simulate the collision times need be adapted. Fortunately several methods can be used to approach these collision time simulation algorithms, see e.g. Bouchard-Côté et al. (2018), Section 2.3.

As another direction for future work, it would be interesting to combine our method with Bayesian variable selection techniques (Ishwaran and Rao, 2005) and a binary version of BPS (Pakman, 2017) to select subset of features that have a large effect on the underlying CTMCs.

Acknowledgments

We would like to thank the action editor, Prof. Zhihua Zhang, and the anonymous reviewers for their helpful comments and suggestions that help improve the paper. We would also like to thank Prof. Sarah P. Otto, Prof. Paul Gustafson and Prof. Cécile Ané for their insightful comments and suggestions. This research was supported by a Discovery Grant from the Natural Science and Engineering Research Council of Canada, and a Canadian Statistical Science Institute Collaborative Research Team Grant.

Appendix A. Connection between Bayesian GLM GTR model and Bayesian GLM chain GTR model

We have defined $\boldsymbol{\phi}^{\text{gtr}}$ in Section 2.2 such that any GTR rate matrices can be represented via the Bayesian GLM rate matrix parameterization. Our objective is to prove that there exists $\boldsymbol{w}_*$ such that for any rate matrices under the Bayesian GLM parameterization, for $\forall \boldsymbol{w}, q^{(\text{rev})}_{x,x'}(\boldsymbol{w})$ can be represented under the Bayesian GLM chain GTR parameterization.

Zhao et al. (2016) have shown that for any rate matrix Q, we can find weights $\boldsymbol{w}$ such that $Q_{(x,x')} = q_{x,x'}(\boldsymbol{w}) = \exp\left\{\langle \boldsymbol{w}^b, \boldsymbol{\phi}^{\text{gtr}}(\{x,x'\})\rangle\right\}\pi_{x'}(\boldsymbol{w}^u)$ under the Bayesian GLM GTR model. In the Bayesian GLM chain GTR model, we show that there exists $\boldsymbol{w}_*$ such that for $\forall \boldsymbol{w} = \begin{pmatrix}\boldsymbol{w}^u \\ \boldsymbol{w}^b\end{pmatrix}$,

$$\begin{aligned}
q^{\text{gtr}}_{x,x'}(\boldsymbol{w}) &= \exp\left\{\langle \boldsymbol{w}^b, \boldsymbol{\phi}^{\text{gtr}}(\{x,x'\})\rangle\right\}\pi_{x'}(\boldsymbol{w}^u) \\
&= q^{\text{chain}}_{x,x'}(\boldsymbol{w}_*) \\
&= \exp\left\{\langle \boldsymbol{w}^b_*, \boldsymbol{\phi}^{\text{chain}}(\{x,x'\})\rangle\right\}\pi_{x'}(\boldsymbol{w}^u_*)
\end{aligned}$$

Thus, we can set $\boldsymbol{w}^u_* = \alpha \boldsymbol{w}^u$ for any $\alpha \in \mathbb{R}$ so that $\pi_{x'}(\boldsymbol{w}^u) = \pi_{x'}(\boldsymbol{w}^u_*)$, for $\forall x' \in \mathcal{X}$. By plugging the definition of $\boldsymbol{\phi}^{\text{gtr}}$ and $\boldsymbol{\phi}^{\text{chain}}$ in Equation 7 and Equation 15 respectively, we require that:

$$\begin{cases}
\left(\boldsymbol{w}^b_*\right)_{\eta(\{x,x'\})} + \left(\boldsymbol{w}^b_*\right)_{\eta(\{x,x'\})-1} = \left(\boldsymbol{w}^b\right)_{\eta(\{x,x'\})}, & \text{for } \eta(\{x,x'\}) = 2,3,\ldots,\mathcal{X}|(|\mathcal{X}|-1)/2), \\
\left(\boldsymbol{w}^b_*\right)_1 = \left(\boldsymbol{w}^b\right)_1, & \text{for } \eta(\{x,x'\}) = 1.
\end{cases}$$

Set $p = \mathcal{X}|(|\mathcal{X}|-1)/2)$, it is equivalent to solve

$$\begin{aligned}
\left(\boldsymbol{w}^b_*\right)_1 &= \left(\boldsymbol{w}^b\right)_1 \\
\left(\boldsymbol{w}^b_*\right)_1 + \left(\boldsymbol{w}^b_*\right)_2 &= \left(\boldsymbol{w}^b\right)_2 \\
\left(\boldsymbol{w}^b_*\right)_2 + \left(\boldsymbol{w}^b_*\right)_3 &= \left(\boldsymbol{w}^b\right)_3 \\
&\vdots \\
\left(\boldsymbol{w}^b_*\right)_{p-1} + \left(\boldsymbol{w}^b_*\right)_p &= \left(\boldsymbol{w}^b\right)_p.
\end{aligned}$$

We obtain the solution $\boldsymbol{w}^b_* = \boldsymbol{B}\boldsymbol{w}^b$, where

$$\boldsymbol{B}_{ij} = \begin{cases}
(-1)^{i+j-2}, & \text{if } j \leqslant i \text{ and } i,j = 1,2,\ldots,(|\mathcal{X}|(|\mathcal{X}|-1)/2), \\
0, & \text{otherwise.}
\end{cases}$$

Thus, the solution exists, where $\boldsymbol{w}_* = \begin{pmatrix}\boldsymbol{w}^u_* \\ \boldsymbol{w}^b_*\end{pmatrix} = \begin{pmatrix}\alpha \boldsymbol{w}^u \\ \boldsymbol{B}\boldsymbol{w}^b\end{pmatrix}.$

Appendix B. Gradient Computation for Univariate Weights in the HMC step under combined sampling scheme

In our proposed sampling scheme LBPS-HMC described in Algorithm 2, we use HMC to update only the univariate weights $\boldsymbol{w}^u$ while fixing the bivariate weights $\boldsymbol{w}^b$ and the auxiliary variable $\boldsymbol{z}$ and LBPS to update the bivariate weights $\boldsymbol{w}^b$ while fixing $\boldsymbol{w}^u$ and $\boldsymbol{z}$. We provide the gradient information for $\boldsymbol{w}^u$ required by HMC of our combined sampling scheme under a reversible, unnormalized rate matrix.

We review that the augmented joint density given the sufficient statistics $\boldsymbol{z}(\boldsymbol{y}) := (\boldsymbol{n}(\boldsymbol{y}), \boldsymbol{h}(\boldsymbol{y}), \boldsymbol{c}(\boldsymbol{y}))$ for a sample CTMC path $\boldsymbol{y}$ under a reversible, unnormalized rate matrix is:

$$\begin{aligned}\log f_{\mathrm{w}|\mathrm{z},\mathrm{y}}(\boldsymbol{w}|\boldsymbol{z},\boldsymbol{y}) &:= -\frac{1}{2}\kappa\|\boldsymbol{w}\|_2^2 - \sum_{x\in\mathcal{X}} h_x \sum_{x'\in\mathcal{X}:x\neq x'} q_{x,x'}(\boldsymbol{w}), \\ &\quad + \sum_{x\in\mathcal{X}}\sum_{x'\in\mathcal{X}:x\neq x'} c_{x,x'} \log\left(q_{x,x'}(\boldsymbol{w})\right) + \sum_{x\in\mathcal{X}} n_x \log(\pi_x(\boldsymbol{w})). \quad (19)\end{aligned}$$

In the ith iteration of LBPS-HMC, while using HMC kernel, we only update $\boldsymbol{w}^u$ while conditioning on the rest. Thus, the exchangeable parameters are fixed at the values obtained in the $(i-1)$th iteration denoted as $\theta_{\{x,x'\}}\left(\left(\boldsymbol{w}^b\right)^{i-1}\right)$. To simplify the notation, we denote $\theta_{\{x,x'\}}\left(\left(\boldsymbol{w}^b\right)^{i-1}\right)$ as $\theta_{\{x,x'\}}$. We can rewrite Equation 19 as:

$$\begin{aligned}\log f_{\mathrm{w^u}|\mathrm{w^b},\mathrm{z},\mathrm{y}}(\boldsymbol{w}^u|\boldsymbol{w}^b,\boldsymbol{z},\boldsymbol{y}) &= -\frac{1}{2}\kappa\|\boldsymbol{w}^u\|_2^2 - \sum_{x\in\mathcal{X}} h_x \sum_{x'\in\mathcal{X}:x\neq x'} \pi_{x'}(\boldsymbol{w}^u)\theta_{\{x,x'\}} \quad (20) \\ &\quad + \sum_{x\in\mathcal{X}}\sum_{x'\in\mathcal{X}:x\neq x'} c_{x,x'}\log\left(\pi_{x'}(\boldsymbol{w}^u)\theta_{\{x,x'\}}\right) + \sum_{x\in\mathcal{X}} n_x \log(\pi_x(\boldsymbol{w}^u))\end{aligned}$$

Thus, the gradient of the augmented joint density with respect to $\boldsymbol{w}^u$ is:

$$\begin{aligned}\nabla \log f_{\mathrm{w^u}|\mathrm{w^b},\mathrm{z},\mathrm{y}}(\boldsymbol{w}^u|\boldsymbol{w}^b,\boldsymbol{z},\boldsymbol{y}) &= -\kappa\boldsymbol{w}^u - \sum_{x\in\mathcal{X}}\sum_{x'\in\mathcal{X}:x\neq x'} h_x q_{x,x'}(\boldsymbol{w}^u)\left(\boldsymbol{\psi}(x') - \sum_{x\in\mathcal{X}}\pi_x(\boldsymbol{w}^u)\boldsymbol{\psi}(x)\right) \\ &\quad + \sum_{x\in\mathcal{X}}\sum_{x'\in\mathcal{X}:x\neq x'} c_{x,x'}\left(\boldsymbol{\psi}(x') - \sum_{x\in\mathcal{X}}\pi_x(\boldsymbol{w}^u)\boldsymbol{\psi}(x)\right) \\ &\quad + \sum_{x\in\mathcal{X}} n_x\left(\boldsymbol{\psi}(x) - \sum_{x\in\mathcal{X}}\pi_x(\boldsymbol{w}^u)\boldsymbol{\psi}(x)\right), \quad (21)\end{aligned}$$

where:

$$\nabla A(\boldsymbol{w}) = \sum_{x\in\mathcal{X}} \boldsymbol{\psi}(x)\pi_x(\boldsymbol{w}).$$

Appendix C. Gradient Computation for Univariate and Bivariate Weights using only HMC kernel as benchmark sampling algorithm

To investigate the computational efficiency of LBPS-HMC, we choose state-of-the-art sampling algorithm HMC to sample $\boldsymbol{w} = \left(\boldsymbol{w}^u, \boldsymbol{w}^b\right)$.

Since HMC requires the gradient of $\boldsymbol{w}$, we derive it in Equation 19:

$$
\begin{aligned}
\nabla \log f_{\mathrm{w}|\mathrm{z},\mathrm{y}}(\boldsymbol{w}|\boldsymbol{z},\boldsymbol{y}) \quad = \quad & -\kappa \boldsymbol{w} - \sum_{x\in\mathcal{X}} \sum_{x'\in\mathcal{X}:x\neq x'} h_x q_{x,x'}\left(\boldsymbol{w}\right) \left(\boldsymbol{\psi}(x') - \sum_{x\in\mathcal{X}} \pi_x(\boldsymbol{w})\boldsymbol{\psi}(x)\right) \\
& - \sum_{x\in\mathcal{X}} \sum_{x'\in\mathcal{X}:x\neq x'} h_x q_{x,x'}\left(\boldsymbol{w}\right) \boldsymbol{\phi}(\{x,x'\}) \\
& + \sum_{x\in\mathcal{X}} \sum_{x'\in\mathcal{X}:x\neq x'} c_{x,x'} \left(\boldsymbol{\psi}(x') - \sum_{x\in\mathcal{X}} \pi_x(\boldsymbol{w})\boldsymbol{\psi}(x)\right) \\
& + \sum_{x\in\mathcal{X}} \sum_{x'\in\mathcal{X}:x\neq x'} c_{x,x'} \boldsymbol{\phi}(\{x,x'\}) + \sum_{x\in\mathcal{X}} n_x \left(\boldsymbol{\psi}(x) - \sum_{x\in\mathcal{X}} \pi_x(\boldsymbol{w})\boldsymbol{\psi}(x)\right).
\end{aligned}
$$

Appendix D. Software implementation correctness checks

We outline in this section a simple extension of Geweke (2004), which allows us to test the correctness of our software implementation. See Geweke (2004) for more background. We call the extension outlined in this section "Exact Invariance Test" (EIT), which is described in more details in the website `https://www.stat.ubc.ca/~bouchard/blang/Testing_Blang_models.html`.

As in Geweke (2004), EIT relies on comparing two sets of simulators, both targeting the joint distribution over the unobserved and observed variables. These methods exploit the fact that exact samples from this joint distribution can be obtained in two ways, described below. In what follows, we denote the prior by g_{w}, while $W'|W \sim K(\cdot|W)$ denotes the MCMC kernel. In our case, it is a Markov kernel based on HMC and LBPS. We also define a test function t. In our context, an example of $t(\cdot)$ is the function used to compute the exchangeable parameters given W. In general, t could take as input both the unobserved and observed variables, but for simplicity it only depends on the unobserved one W here.

The first simulator consists in the following steps:

- For $m \in \{1, 2, \dots, M_1\}$,
  - Sample $W_m \sim g_{\text{w}}$.
  - Optional step: generate a dataset using the likelihood conditionally on W_m (would only be needed if t depends on both the synthetic data and W).
  - $F_m = t(W_m)$.

The second simulator consists in the following steps:

- For $m \in \{1, 2, \dots, M_2\}$:
  - Sample $W_{1,m} \sim g_{\text{w}}$.
  - Generate a dataset using the likelihood conditionally on $W_{1,m}$.
  - For $j \in \{2, 3, \dots, J\}$: $W_{j,m}|W_{j-1,m} \sim K(\cdot|W_{j-1,m})$.
  - $H_m = t(W_{J,m})$.

For any $J \geqslant 1$, $a \in \{1, 2, \dots, M_1\}$, $b \in \{1, 2, \dots, M_2\}$, the random variable F_a (generated from the first simulator) is equal in distribution to the random variable H_b (generated by the second simulator) if and only if the kernel K is f invariant. This follows directly from the definition of global balance of Markov chains. This exact equality of distributions can then be used as the basis of a frequentist point-null hypothesis test.

EIT Results

Table 3 and Table 4 show that our proposed LBPS-HMC has passed the EIT since all pvalues are bigger than a threshold of $0.05/n$, where n represents the number of parameters in the test to take multiple comparisons into account. We show the test results of using HMC alone in Table 5 and Table 6.

	Parameter Index				
KS test	1	2	3	4	5
Pvalue	0.640	0.200	0.329	0.114	0.167
Test statistics	0.060	0.087	0.077	0.097	0.090

Table 3: EIT results for the weights of the stationary distribution using LBPS-HMC.

	Parameter Index									
KS test	1	2	3	4	5	6	7	8	9	10
Pvalue	0.200	0.504	0.441	0.061	0.571	0.838	0.382	0.709	0.238	0.382
Test statistics	0.087	0.067	0.070	0.107	0.063	0.050	0.073	0.057	0.083	0.073

Table 4: EIT results for exchangeable parameters using LBPS-HMC.

	Parameter Index				
KS test	1	2	3	4	5
Pvalue	0.967	0.200	0.640	0.838	0.640
Test statistics	0.040	0.087	0.060	0.050	0.060

Table 5: EIT results for the weights of the stationary distribution using HMC alone.

	Parameter Index									
KS test	1	2	3	4	5	6	7	8	9	10
Pvalue	0.139	0.076	0.504	0.281	0.838	0.640	0.936	0.504	0.139	0.238
Test statistics	0.093	0.103	0.067	0.080	0.050	0.060	0.043	0.067	0.093	0.083

Table 6: EIT results for exchangeable parameters using HMC alone.

Appendix E. Posterior sample comparison between LBPS-HMC and HMC

In this section, we provide the summary statistics including the minimum, first quantile, median, mean, third quantile and maximum of the ARD defined in Equation 22, across all exchangeable parameters from one single run of LBPS-HMC and HMC in Table 7 under the 8-by-8 Bayesian GLM chain GTR model described in Section 3. We also provide the boxplots of posterior samples collected from LBPS-HMC and HMC under the same set-up in Figure 9.

Min.	1st Quantile	Median	Mean	3rd Quantile	Max.
1.4%	1.16%	2.70%	3.05%	3.63%	12.06%

Table 7: Summary of ARD across exchangeable parameters between HMC and LBPS.

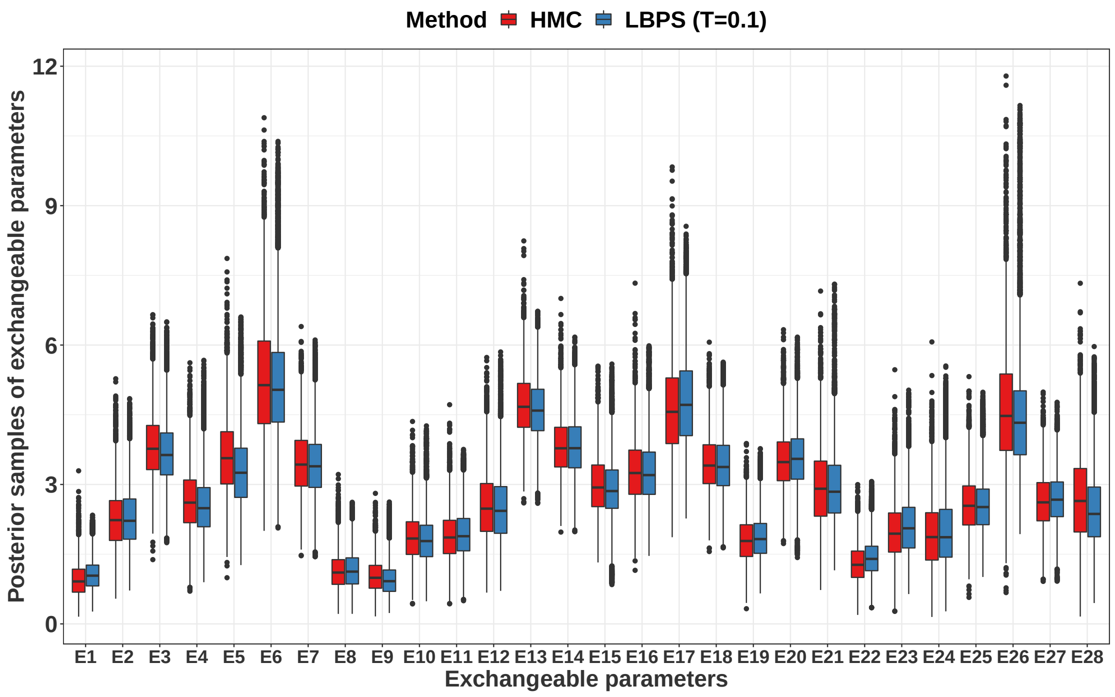

Figure 9: Boxplot of posterior samples comparison for exchangeable parameters of an 8-by-8 rate matrix between LBPS-HMC and HMC.

Appendix F. Pairwise Euclidean Distance of Amino Acids

	Y	H	Q	R	T	N	K	D	E	G	F	L	A	S	P	I	M	V	C	W
Y	0	83	99	77	92	143	85	160	122	147	22	36	112	144	110	33	36	55	194	37
H	83	0	24	29	47	68	32	81	40	98	100	99	86	89	77	94	87	84	174	115
Q	99	24	0	43	42	46	53	61	29	87	116	113	91	68	76	109	101	96	154	130
R	77	29	43	0	71	86	26	96	54	125	97	102	112	110	103	97	91	96	180	102
T	92	47	42	71	0	65	78	85	65	59	103	92	58	58	38	89	81	69	149	128
N	143	68	46	86	65	0	94	23	42	80	158	153	111	46	91	149	142	133	139	174
K	85	32	53	26	78	94	0	101	56	127	102	107	106	121	103	102	95	97	202	110
D	160	81	61	96	85	23	101	0	45	94	177	172	126	65	108	168	160	152	154	181
E	122	40	29	54	65	42	56	45	0	98	140	138	107	80	93	134	126	121	170	152
G	147	98	87	125	59	80	127	94	98	0	153	138	60	56	42	135	127	109	159	184
F	22	100	116	97	103	158	102	177	140	153	0	22	113	155	114	21	28	50	205	40
L	36	99	113	102	92	153	107	172	138	138	22	0	96	145	98	5	15	32	198	61
A	112	86	91	112	58	111	106	126	107	60	113	96	0	99	27	94	84	64	195	148
S	144	89	68	110	58	46	121	65	80	56	155	145	99	0	74	142	135	124	112	177
P	110	77	76	103	38	91	103	108	93	42	114	98	27	74	0	95	87	68	169	147
I	33	94	109	97	89	149	102	168	134	135	21	5	94	142	95	0	10	29	198	61
M	36	87	101	91	81	142	95	160	126	127	28	15	84	135	87	10	0	21	196	67
V	55	84	96	96	69	133	97	152	121	109	50	32	64	124	68	29	21	0	192	88
C	194	174	154	180	149	139	202	154	170	159	205	198	195	112	169	198	196	192	0	215
W	37	115	130	102	128	174	110	181	152	184	40	61	148	177	147	61	67	88	215	0

Appendix G. Nearest Neighbour Pariwise Amino Acid Ordering

Algorithm 3 Nearest neighbour pairwise amino acid ordering

```
 1: Initialization:
    $\mathcal{X}^{\text{dynamic, distinct}} = \mathcal{X}^{\text{distinct}}$, counter=0.
    AminoAcidPairRank=DynamicSupportForAminoAcid =dict(), which is an empty dictionary.
    Set $\alpha$= $\beta$=$i_0$=$i_1$=NULL.
 2: for $i \in \mathcal{X}$ do
 3:     DynamicSupportForAminoAcid[$i$] = $\mathcal{X} \setminus \{i\}$.
 4: end for
 5: while $\mathcal{X}^{\text{dynamic, distinct}}$ is not empty do
 6:     if $i_0$ is not NULL and $i_1$ is not NULL then
 7:         Find support of $i_0$: $\alpha$=DynamicSupportForAminoAcid[$i_0$].
 8:         Find support of $i_1$: $\beta$=DynamicSupportForAminoAcid[$i_1$].
 9:     end if
10:     if neither of $\alpha$ or $\beta$ is empty then
11:         rowDist = AminoAcidDist[$i_0$, $\alpha$]
12:         colDist = AminoAcidDist[$\beta$, $i_1$]
13:         rowMinInd = $\alpha$[argmin(rowDist)]
14:         colMinInd = $\beta$[argmin(colDist)]
15:         if AminoAcidDist[$i_0$, rowMinInd] $\leqslant$ AminoAcidDist[colMinInd, $i_1$] then
16:             $i_1$ = rowMinInd
17:         else if AminoAcidDist[$i_0$, rowMinInd] > AminoAcidDist[colMinInd, $i_1$] then
18:             $i_0$ = colMinInd
19:         end if
20:     else if $\alpha$ is NULL and $\beta$ is not NULL then
21:         colDist=AminoAcidDist[$\beta$, $i_1$]
22:         colMinInd=$\beta$[argmin(colDist)]
23:         $i_0$= colMinInd
24:     else if $\alpha$ is not NULL and $\beta$ is NULL then
25:         rowDist = AminoAcidDist[$i_0$, $\alpha$]
26:         rowMinInd=$\alpha$[argmin(rowDist)]
27:         $i_1$ = rowMinInd
28:     else
29:         Find the amino acid pair SmallestPair $\in \mathcal{X}^{\text{dynamic,distinct}}$ with nonzero minimum in AminoAcidDist.
30:         $i_0$= index of the first amino acid in SmallestPair in $\mathcal{X}$.
31:         $i_1$ = index of the second amino acid in SmallestPair in $\mathcal{X}$.
32:     end if
33:     AminoAcidPairRank[$i_0$, $i_1$]=AminoAcidPairRank[$i_1$, $i_0$] =counter, counter ++.
34:     AminoAcidDist[$i_0$, $i_1$]=AminoAcidDist[$i_1$, $i_0$]=$\infty$.
35:     if $i_1 \in$ DynamicSupportForAminoAcid[$i_0$] then
36:         Remove $i_1$ from DynamicSupportForAminoAcid[$i_0$]
37:     end if
38:     if $i_0 \in$ DynamicSupportForAminoAcid[$i_1$] then
39:         Remove $i_0$ from DynamicSupportForAminoAcid[$i_1$]
40:     end if
41:     $s_0 = \mathcal{X}_{i_0} + \mathcal{X}_{i_1}$ (obtain amino acid pair $s_0$), $s_1 = \mathcal{X}_{i_1} + \mathcal{X}_{i_0}$ (obtain amino acid pair $s_1$).
42:     if $s_0 \in \mathcal{X}^{\text{dynamic, ordered,distinct}}$ then
43:         Remove $s_0$ from $\mathcal{X}^{\text{dynamic, distinct}}$.
44:     end if
45:     if $s_1 \in \mathcal{X}^{\text{dynamic, distinct}}$ then
46:         Remove $s_1$ from $\mathcal{X}^{\text{dynamic, distinct}}$
47:     end if
48: end while
49: Return AminoAcidPairRank.
```

Appendix H. Traceplot comparison for real data analysis

We provide the traceplot on a randomly selected exchangeable parameter using LBPS-HMC compared with HMC. There is a small difference between posterior means with ARD 0.43%.

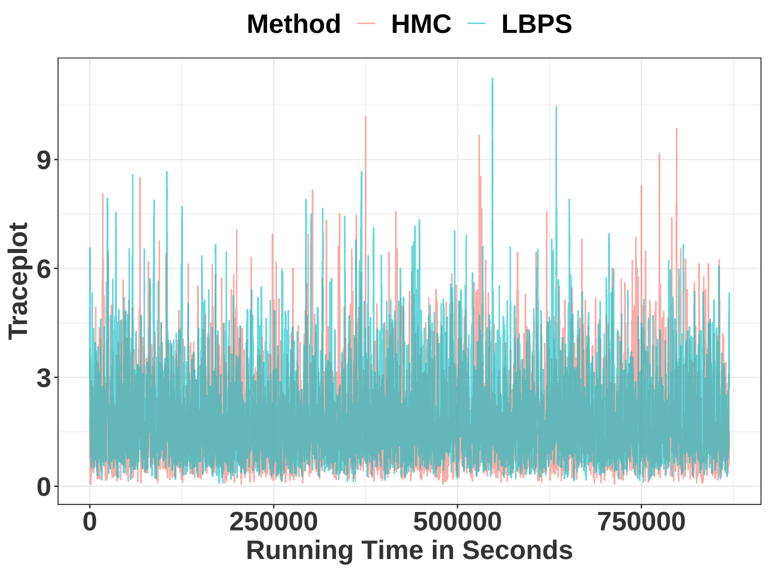

Figure 10: Traceplot for the 145th exchangeable parameter using LBPS-HMC compared to HMC with ARD 0.43% in terms of the posterior mean.

Appendix I. Correctness check via ARD between two samplers

With the same prior distribution and likelihood for the parameters (weights) of interest, LBPS-HMC and HMC share the same posterior distribution. We evaluate the discrepancy among the posterior samples from the two algorithms using ARD:

$$ARD(x, y) \quad = \quad \frac{|x - y|}{\max(x, y)}, \text{ for } \forall x, y > 0. \tag{22}$$

Figure 12 and Table 8 demonstrate the ARD across all exchangeable parameters between HMC and LBPS-HMC with trajectory length 0.2.

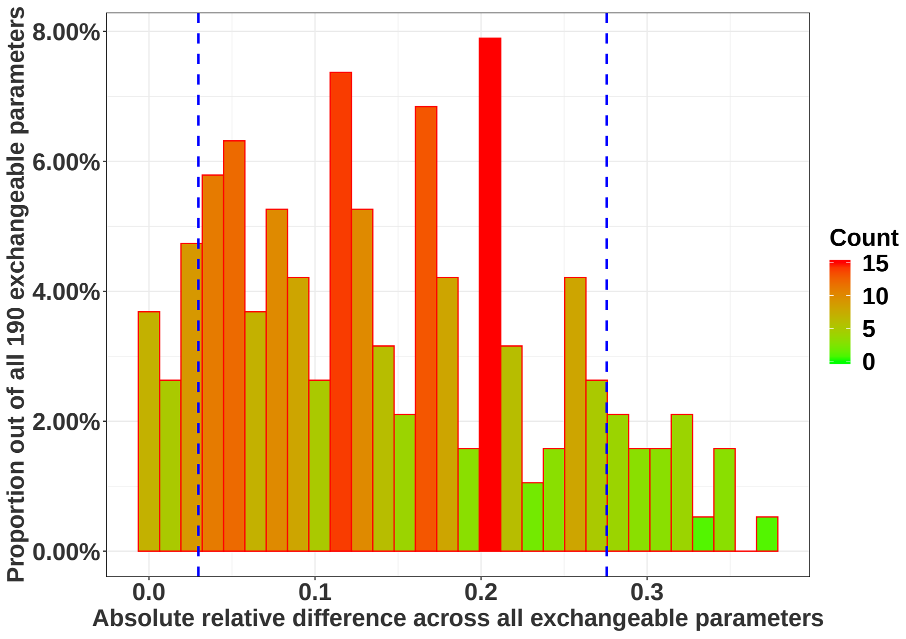

Figure 11: ARD across exchangeable parameters between LBPS-HMC compared to HMC with blue dotted lines representing the 10% and 90% quantile of ARD.

Min.	1st Quantile	Median	Mean	3rd Quantile	Max.
0.3%	7.5%	14.8%	15.1%	22.0%	44.4%

Table 8: Summary of ARD across exchangeable parameters.

Since the maximum value of ARD is 44.4%, we further check whether it will decrease as expected if we run the chain longer. The discrepancy between the posterior samples from the two algorithms decreases if we have a longer chain. We validate this by providing a violin plot below comparing the ARD from the current run denoted as "shorter"with a longer run denoted as "longer". HMC had a walltime of 12 days and LBPS had a walltime of 10 days. Later, we ran each method for two more days. The maximum of ARD dropped from 44% to 37% .

We provide the quantile of ESS in Table 9 and the actual ESS in Table 10 of the exchangeable parameters that have ARD bigger than its 95% quantile between LBPS-HMC and HMC. We have found that a relatively bigger discrepancy between the posterior means occurs when either both methods have low ESS or when LBPS-HMC has high ESS (such as 97.9% quantile and 94.2% quantile across all exchangeable parameters) but HMC does not obtain an equivalently high ESS for the same parameter.

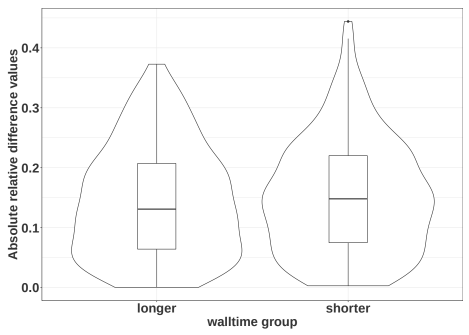

Figure 12: Violin plot of ARD across exchangeable parameters between a "shorter" and "longer" run.

LBPS-HMC	2.63%	2.105%	8.42%	5.79%	77.4%	94.2%	75.8%	97.9 %	0.526%	3.16%
HMC	16.84%	0.526%	15.79%	2.63%	53.7%	84.2%	73.7%	98.4%	1.053%	5.26%

Table 9: Quantiles of ESS of exchangeable parameters with ARD bigger than its 95% quantile.

LBPS	1071	1051	1248	1147	2145	2330	2115	2451	773	1118
HMC	544	152	604	488	927	1032	1090	591	190	718

Table 10: The ESS of exchangeable parameters with ARD bigger than its 95% quantile.

Appendix J. Correctness of the sampling scheme LBPS-HMC

In this section, we establish the correctness of our sampling scheme LBPS-HMC described in Algorithm 2, i.e. that it is invariant with respect to the distribution with density $g_{\mathrm{w|y}}(\boldsymbol{w}|\mathcal{Y})$ in Equation 9. The sufficient statistics Z defined in Section 2 are used as "transient auxiliary variable," i.e. quantities used to define a transition kernel but discarded after each transition is performed, a common technique in the MCMC literature, e.g. Zhao et al. (2016). We define $\boldsymbol{x}$ as the augmented sample path and a mapping $\boldsymbol{d}(\cdot)$ that takes $\boldsymbol{x}$ as input and outputs a vector containing the state of the CTMC at each observation time, $\boldsymbol{y}$ as the states of the partially observed paths, and g_{emi} as an emission probability or density function.

The joint distribution after auxiliary variable augmentation is

$$g_{\mathrm{w,x,y,z}}(\boldsymbol{w},\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) := g_{\mathrm{w}}(\boldsymbol{w}) g_{\mathrm{x|Q}}(\boldsymbol{x} \mid Q(\boldsymbol{w})) g_{\mathrm{y|d}}(\boldsymbol{y} \mid \boldsymbol{d}(\boldsymbol{x})) \delta_{\boldsymbol{z}(\boldsymbol{x})}(\boldsymbol{z}) \tag{23}$$

$$g_{\mathrm{y|d}}(\boldsymbol{y} \mid \boldsymbol{d}) := \prod_{\boldsymbol{v}\in \mathrm{V}} g_{\mathrm{emi}}\left(y_v \mid d_v\right). \tag{24}$$

Since $g_{\mathrm{x|Q}}$ and $\delta_{\boldsymbol{z}(\boldsymbol{x})}$ integrate to one when marginalizing $\boldsymbol{x}$ and $\boldsymbol{z}$, we have that the augmented target admits the posterior distribution of interest ($g_{\mathrm{w|y}}$, Equation 9) as a marginal.

In the following, we define a Markov chain $W^{(1)}, W^{(2)}, \ldots,$ on the state space $\mathbb{R}^P$. Our sampling scheme is a Markov transition kernel T which is itself constructed using an alternation of several kernels T_1, T_2, T_3. We show that each step in this sequence keeps $g_{\mathrm{w|y}}(\boldsymbol{w}|\mathcal{Y})$ invariant. It follows from standard results in MCMC theory that the alternation of these three kernels is also invariant with respect to $g_{\mathrm{w|y}}$ (Tierney, 1994). A first kernel T_1 is used to instantiate the auxiliary variable Z. Then, a second kernel T_2 is used to perform changes in W conditionally on Z, and finally, a last kernel T_3 projects the augmented space back to the target space $\mathbb{R}^P$. The broad lines of the arguments follow Appendix A in Zhao et al. (2016), with the key difference being that a combination of LBPS and HMC is used to design the kernel T_2 which updates W given Z. We focus on the points in which the argument differs compared to Zhao et al. (2016).

Our sequence of transition kernels can be denoted as: $\boldsymbol{w} \xmapsto{T_1} (\boldsymbol{w},\boldsymbol{z}) \xmapsto{T_2} (\boldsymbol{w}',\boldsymbol{z}) \xmapsto{T_3} \boldsymbol{w}'$. Given the observations Y and current weights W, the transition kernel T_1 performs the auxiliary variable Z sampling with density $T_1(\boldsymbol{w}',\boldsymbol{z}' \mid \boldsymbol{w}) := \delta_{\boldsymbol{w}}(\boldsymbol{w}') g_{\mathrm{z|w,y}}(\boldsymbol{z} \mid \boldsymbol{w},\boldsymbol{y})$. Details of this sampling step are outlined in Section 4.3 and Supplement 3 of Zhao et al. (2016).

The kernel T_2 performs several Metropolis-within-Gibbs steps on $\boldsymbol{w}$ while keeping the auxiliary variable $\boldsymbol{z}$ fixed. In Zhao et al. (2016), AHMC is used to update all elements of $\boldsymbol{w}$. In this paper, for reasons described in Section 5.1, we partition $\boldsymbol{w}$ into two blocks: a first block corresponding to the univariate weights $\boldsymbol{w}^u$ and a second block, to the bivariate weights $\boldsymbol{w}^b$ (see Algorithm 2). We use HMC to update $\boldsymbol{w}^u$ while fixing $\boldsymbol{w}^b$ and $\boldsymbol{z}$ and LBPS to update $\boldsymbol{w}^b$ while fixing $\boldsymbol{w}^u$ and $\boldsymbol{z}$. This alternation can be denoted as $(\boldsymbol{w},\boldsymbol{z}) = ((\boldsymbol{w}^u,\boldsymbol{w}^b),\boldsymbol{z}) \overset{\text{HMC}}{\longmapsto} ((\boldsymbol{w}'^u,\boldsymbol{w}^b),\boldsymbol{z}) \overset{\text{LBPS}}{\longmapsto} ((\boldsymbol{w}'^u,\boldsymbol{w}'^b),\boldsymbol{z}) = (\boldsymbol{w}',\boldsymbol{z})$. Again, as long as both the HMC and LBPS components are invariant with respect to an arbitrary conditional distribution of the augmented target $g_{\mathrm{w,y,z}}$, their alternation is also guaranteed to preserve invariance. For HMC, see for example Neal (2010), for LBPS invariance was shown for an arbitrary distribution in Bouchard-Côté et al. (2018), Appendix F, so in particular it holds for a conditional distribution.

References

Christophe Andrieu, Alain Durmus, Nikolas Nüsken, and Julien Roussel. Hypercoercivity of piecewise deterministic Markov Process Monte Carlo. *arXiv preprint arXiv:1808.08592*, 2018.

Maria Anisimova and Carolin Kosiol. Investigating protein-coding sequence evolution with probabilistic codon substitution models. *Molecular biology and evolution*, 26(2):255–271, 2008.

Joris Bierkens. Non-reversible metropolis-hastings. *Statistics and Computing*, 26(6):1213–1228, 2016.

Joris Bierkens, Paul Fearnhead, and Gareth Roberts. The zig-zag process and super-efficient sampling for Bayesian analysis of big data. *The Annals of Statistics*, 47(3):1288–1320, 2019.

Alexandre Bouchard-Côté, Sebastian J Vollmer, and Arnaud Doucet. The bouncy particle sampler: A nonreversible rejection-free Markov chain Monte Carlo method. *Journal of the American Statistical Association*, 113(522):855–867, 2018.

Alexandre Bouchard-Côté, Kevin Chern, Davor Cubranic, Sahand Hosseini, Justin Hume, Matteo Lepur, Zihui Ouyang, and Giorgio Sgarbi. Blang: Bayesian declarative modelling of arbitrary data structures. *arXiv preprint arXiv:1912.10396*, 2019.

Hsiu-Hsi Chen, Stephen W Duffy, and Laszlo Tabar. A Markov chain method to estimate the tumour progression rate from preclinical to clinical phase, sensitivity and positive predictive value for mammography in breast cancer screening. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 45(3):307–317, 1996.

Ting-Li Chen and Chii-Ruey Hwang. Accelerating reversible Markov chains. *Statistics & Probability Letters*, 83(9):1956–1962, 2013.

Christelle Combescure, Pascal Chanez, Philippe Saint-Pierre, Jean Pierre Daures, Herve Proudhon, and Philippe Godard. Assessment of variations in control of asthma over time. *European Respiratory Journal*, 22(2):298–304, 2003.

Mark HA Davis. Piecewise-deterministic Markov processes: A general class of non-diffusion stochastic models. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 353–388, 1984.

George Deligiannidis, Daniel Paulin, Alexandre Bouchard-Côté, and Arnaud Doucet. Randomized Hamiltonian Monte Carlo as scaling limit of the bouncy particle sampler and dimension-free convergence rates. *Annals of Applied Probability*, 2020.

P. J. Diggle, J. A. Tawn, and R. A. Moyeed. Model-Based Geostatistics. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 47(3):299–350, 1998.

Paul Fearnhead, Joris Bierkens, Murray Pollock, and Gareth O Roberts. Piecewise deterministic Markov processes for continuous-time Monte Carlo. *Statistical Science*, 33(3): 386–412, 2018.

James M Flegal, John Hughes, Dootika Vats, and Ning Dai. mcmcse: Monte Carlo standard errors for MCMC. *Riverside, CA and Minneapolis, MN. R package version*, pages 1–0, 2012.

John Geweke. Getting it right: Joint distribution tests of posterior simulators. *Journal of the American Statistical Association*, 99(467):799–804, 2004.

Rao Grantham. Amino acid difference formula to help explain protein evolution. *Science*, 185(4154):862–864, 1974.

Peter Guttorp and Vladimir N Minin. *Stochastic modeling of scientific data.* CRC Press, 2018.

Masami Hasegawa, Hirohisa Kishino, and Taka-aki Yano. Dating of the human-ape splitting by a molecular clock of mitochondrial DNA. *Journal of Molecular Eevolution*, 22:160–174, 1985.

Matthew X He, Miguel A Jiménez-Montaño, and Paolo E Ricci. Amino acids, Euclidean distance and symmetric matrix. In *Proceedings of the International Conference on Bioinformatics & Computational Biology (BIOCOMP)*, page 1. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2011.

Matthew D Hoffman and Andrew Gelman. The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(1): 1593–1623, 2014.

Hemant Ishwaran and J Sunil Rao. Spike and slab variable selection: frequentist and bayesian strategies. *The Annals of Statistics*, 33(2):730–773, 2005.

Christopher H Jackson, Linda D Sharples, Simon G Thompson, Stephen W Duffy, and Elisabeth Couto. Multistate Markov models for disease progression with classification error. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 52(2):193–209, 2003.

Thomas H. Jukes and Charles R. Cantor. *Evolution of Protein Molecules.* New York: Academic Press., 1969.

JD Kalbfleisch and Jerald F Lawless. The analysis of panel data under a Markov assumption. *Journal of the American Statistical Association*, 80(392):863–871, 1985.

Richard Kay. Proportional hazard regression models and the analysis of censored survival data. *Applied Statistics*, 26(3):227–237, 1977.

Motoo Kimura. A simple method for estimating evolutionary rate of base substitutions through comparative studies of nucleotide sequences. *Journal of Molecular Evolution*, 16: 111–120, 1980.

Yu-Ying Liu, Shuang Li, Fuxin Li, Le Song, and James M Rehg. Efficient learning of continuous-time hidden Markov models for disease progression. In *Advances in neural information processing systems*, pages 3600–3608, 2015.

Cleve Moler and Charles Van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM review*, 45(1):3–49, 2003.

Radford M Neal. Improving asymptotic variance of MCMC estimators: Non-reversible chains are better. *arXiv preprint math/0407281*, 2004.

Radford M Neal. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 54:113–162, 2010.

Radford M Neal et al. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2(11):2, 2011.

James R Norris. *Markov chains*. Number 2. Cambridge university press, 1998.

Ari Pakman. Binary bouncy particle sampler. *arXiv preprint arXiv:1711.00922*, 2017.

Ari Pakman, Dar Gilboa, David Carlson, and Liam Paninski. Stochastic bouncy particle sampler. In *International Conference on Machine Learning*, pages 2741–2750, 2017.

Elias AJF Peters et al. Rejection-free Monte Carlo sampling for general potentials. *Physical Review E*, 85(2):026703, 2012.

Ardavan Saeedi and Alexandre Bouchard-Côté. Priors over recurrent continuous time processes. In *Advances in Neural Information Processing Systems*, pages 2052–2060, 2011.

Yi Sun, Jürgen Schmidhuber, and Faustino J Gomez. Improving the asymptotic performance of Markov chain Monte Carlo by inserting vortices. In *Advances in Neural Information Processing Systems*, pages 2235–2243, 2010.

Luke Tierney. Markov Chains for Exploring Posterior Distributions. *Annals of Statistics*, 22(4):1701–1728, 1994.

Paul Vanetti, Alexandre Bouchard-Côté, George Deligiannidis, and Arnaud Doucet. Piecewise deterministic Markov Chain Monte Carlo. *arXiv preprint arXiv:1707.05296*, 2017.

Ziyu Wang, Shakir Mohamed, and Nando Freitas. Adaptive Hamiltonian and Riemann manifold Monte Carlo. In *International Conference on Machine Learning*, pages 1462–1470, 2013.

G George Yin and Qing Zhang. *Continuous-time Markov chains and applications: A two-time-scale approach*, volume 37. Springer Science & Business Media, 2012.

Tingting Zhao, Ziyu Wang, Alexander Cumberworth, Joerg Gsponer, Nando de Freitas, Alexandre Bouchard-Côté, et al. Bayesian analysis of continuous time markov chains with application to phylogenetic modelling. *Bayesian Analysis*, 11(4):1203–1237, 2016.