Bayesian Phylogenetic Inference via Monte Carlo Methods

by

Liangliang Wang

M.Sc. Statistics, McGill University, 2007

M.Sc. Informatics and Signal Processing, Peking University, 2005

B.Sc. Computer Science, Zhengzhou University, 2002

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

in

THE FACULTY OF GRADUATE STUDIES

(Statistics)

The University Of British Columbia

(Vancouver)

August 2012

Abstract

A main task in evolutionary biology is phylogenetic tree reconstruction, which determines the ancestral relationships among different species based on observed molecular sequences, e.g. DNA data. When a stochastic model, typically continuous time Markov chain (CTMC), is used to describe the evolution, the phylogenetic inference depends on unknown evolutionary parameters (hyper-parameters) in the stochastic model. Bayesian inference provides a general framework for phylogenetic analysis, able to implement complex models of sequence evolution and to provide a coherent treatment of uncertainty for the groups on the tree. The conventional computational methods in Bayesian phylogenetics based on Markov chain Monte Carlo (MCMC) cannot efficiently explore the huge tree space, growing super exponentially with the number of molecular sequences, due to difficulties of proposing tree topologies. sequential Monte Carlo (SMC) is an alternative to approximate posterior distributions. However, it is non-trivial to directly apply SMC to phylogenetic posterior tree inference because of its combinatorial intricacies.

We propose the combinatorial sequential Monte Carlo (CSMC) method to generalize applications of SMC to non-clock tree inference based on the existence of a flexible partially ordered set (poset) structure, and we present it in a level of generality directly applicable to many other combinatorial spaces. We show that the proposed CSMC algorithm is consistent and fast in simulations. We also investigate two ways of combining SMC and MCMC to jointly estimate the phylogenetic trees and evolutionary parameters, particle Markov chain Monte Carlo (PMCMC) algorithms with CSMC at each iteration and an SMC sampler with MCMC moves. Further, we present a novel way to estimate the transition probabilities for a general CTMC, which can be used to solve the computing bottleneck in a general evolu-

tionary model, string-valued continuous time Markov Chain (SCTMC), that can incorporate a wide range of molecular mechanisms.

Table of Contents

List of Tables

List of Figures

List of Algorithms

Glossary

BDP birth-death process

CSMC combinatorial sequential Monte Carlo

CTMC continuous time Markov chain

CV coefficient of variation

ESS effective sample size

GS Gibbs sampler

GPUs graphics processing units

indels insertions or deletions

K2P Kimura two-parameter model

KF Kuhner Felsenstein

MB MrBayes

MCMC Markov chain Monte Carlo

MH Metropolis-Hastings

MJP Markov jump process

MSA multiple sequence alignment

NJ neighbor joining

NNI nearest neighbor interchange

PGS particle Gibbs sampler

PIMH particle independent Metropolis-Hastings

PM partition metric

PMCMC particle Markov chain Monte Carlo

PMMH particle marginal Metropolis-Hastings sampler

poset partially ordered set

RF Robinson-Foulds

SCTMC string-valued continuous time Markov Chain

SAWs self-avoiding walks

SMC sequential Monte Carlo

SSM slipped strand mispairing

Acknowledgments

I would like to take this opportunity to thank those who have influenced my life and helped me to complete my Ph.D. program and this thesis.

First and foremost, I would like to express my sincerest gratitude to my co-supervisors, Dr. Arnaud Doucet and Dr. Alexandre Bouchard-Côté. I am sincerely thankful to Arnaud for introducing me to the area of sequential Monte Carlo and believing in my academic potential. I would like to thank Alex for his exceptional generosity with his time, energy, support as well as resources. I will never forget the numerous inspiring discussions we had about phylogenetics. I would also like to acknowledge the financial support from Arnaud and Alex.

Many thanks go to the people who have been influential in my academic journey. I am sincerely thankful to Dr. Nandini Dendukuri and Dr. Alula Hadgu for collaborating with me in the area of Bayesian approaches for medical diagnostic tests and giving me enormous amount of support in my job applications. I am grateful to Dr. James O. Ramsay, for introducing me to the area of Functional Data Analysis (FDA) and inviting me to many holiday parties in his house. I would like to thank Dr. Jinwen Ma for teaching me about bioinformatics and neural networks. I am thankful to Mr. Haoran Qin, my high school biology teacher, for originally stimulating my interest in biology and mentoring me for the (Chinese) National Olympic Contest of Biology in 1997.

I would like to extend my thanks to the professors in UBC. They have given me excellent lectures and all kinds of help and support. Particularly, I would like to thank Dr. Paul Gustafson, Dr. Lang Wu, and Dr. Anne Condon for being in my examination committee and providing insightful comments and suggestions. I am very grateful to Dr. Jiahua Chen, Dr. Lang Wu, and Dr. Ruben H. Zamar for their

altruistic help with my job talk. I would like to thank Dr. Nancy Heckman and Dr. Matías Salibián-Barrera for advising me on robust FDA and supporting me in many aspects. I am thankful to Dr. John Petkau for helping me with the consulting projects and academic writing. I would like to thank Dr. William J. Welch and Dr. Harry Joe for making me feel comfortable in the department.

Furthermore I would like to thank my student colleagues and friends in UBC for creating a friendly environment for development and growth. Special thanks go to Yanling (Tara) Cai, Camila P. Estevam de Souza, Julia Viladomat, Mike Danilov, Xiaoli Yu, Li Xing, Xuekui Zhang, Yongliang (Vincent) Zhai, Ardavan Saeedi, Bonnie Kirkpatrick, Luke Bornn, Ehsan Karim, Chen Xu, Lei (Larry) Hua, Song Cai, and Yang (Seagle) Liu.

I deeply thank my parents and my sister who always stand by me and offer me endless love, care, and support. I would like to show my special appreciation to my husband, Dr. Jiguo Cao, for always being there for me and for being patient and supportive during this long journey.

For partial support of this work through graduate scholarships and research grants, thanks are due to the Natural Sciences and Engineering Research Council of Canada.

Chapter 1

Introduction

A main task in evolutionary biology is phylogenetic reconstruction, which reconstructing the ancestral relationships among different species and estimating evolutionary parameters as well. The available data are typically biological sequences of DNA, RNA, and protein. Bayesian phylogenetics aims to estimate the posterior distribution of phylogenetic tree and evolutionary parameters given the observed biological sequences, and then to make inferences about a function of the tree by computing its posterior expectation.

Calculating the posterior distribution of phylogenetic trees analytically would theoretically involve summation over all possible trees and for each tree integration over all possible combinations of branch lengths. There is no closed-form solution except for very small trees. Moreover, the total number of possible phylogenetic trees grows super exponentially with the number of biological sequences increases. This hard posterior computation is typically estimated using Markov chain Monte Carlo (MCMC) algorithms.

In phylogenetics, MCMC algorithms construct a Markov chain that has the desired posterior distribution of phylogenetic trees as its equilibrium distribution. At each MCMC iteration, a new tree is proposed by making small perturbations on the current tree and then we accept the proposed tree with the probability of the Metropolis-Hastings (MH) ratio or remain at the current tree with the remaining probability.

Various MCMC methods for phylogenetics differ from one another because of

different evolutionary models, proposal distributions, and prior distributions. The development of modern Bayesian phylogenetics originated in the earlier work of [52, 67, 70, 77, 82, 113]. [82] and [113] showed how posterior probabilities of trees could be calculated under a linear birth-death prior for a small number of species. [77] proposed the GLOBAL algorithm in which all branch lengths are changed with every cycle. In these publications, a common assumption is that the phylogenetic tree is a molecular clock. [67] discussed two types of MCMC algorithms, the GLOBAL algorithm and LOCAL algorithm, programmed in their software package BAMBE. [67] also extended these algorithms from clock trees to nonclock trees. [70] restricted their model to molecular clock trees. Their MCMC algorithm uses auxiliary variables of the sequences of internal nodes to improve computational efficiency per cycle by avoiding integrating over these sequences. [52] provided an overview of the traditional and Bayesian approaches for phylogeny estimation.

Bayesian phylogenetics is becoming more and more popular for several reasons. Many studies have focused on comparing nonparametric bootstrap and posterior probabilities in terms of estimating the tree uncertainty [1, 20, 29, 96, 102]. These studies show that bootstrap tends to be too conservative. Moreover, Bayesian methods might be a faster way to assess support for trees than maximum likelihood bootstrapping [25]. In addition, the Bayesian framework can incorporate complex evolutionary models, and there are many user-friendly software packages available for Bayesian phylogenetics using MCMC; e.g. BAMBE [97], MrBayes [55, 90, 91], and BEAST [28].

The main difficulty in Bayesian phylogenetic inference with MCMC lies in the efficiency with which topology proposals sample tree space. Since the target posterior distribution of phylogenetic trees is high dimensional and multimodal, it can be difficult to design appropriate proposal distributions needed to achieve efficient MCMC algorithm. [66] evaluated two classes of proposal distributions that change topology and branch lengths simultaneously for phylogenetic inference for unrooted trees. [51] studied different proposal distributions for constrained rooted clock trees. The results in [66] and [51] show that the design of the currently used transition kernels leads to a low acceptance rate of transitions unless a small neighborhood is used. As a result, MCMC algorithms need a long computing time

to give reliable estimates for the parameters under study. [50] explored new MCMC transition kernels for sampling the posterior distribution in tree space. However, due to the huge size of the tree space, it remains challenging to design proposal distributions that make Markov chains mix well and move quickly among all the trees with high probability mass. Another main disadvantage of using MCMC is that it can be difficult to determine whether the MCMC approximation has achieved convergence.

Moreover, the MCMC algorithms are often computationally expensive because each iteration requires recalculating the likelihood of the full tree even when only a small perturbation is made on the current tree. This computation bottleneck has prevented the application of Bayesian methods to large phylogenetic studies, and increase in the CPU speed is unlikely to address this issue because increasingly cheap sequencing technologies are creating massive molecular data at a faster rate. Such data represent an opportunity for estimating more accurate models of evolutionary histories underlying species. At the same time, the massive amount of molecular data makes a computational challenge for Bayesian phylogenetics, where running MCMC chains for several months is not uncommon.

sequential Monte Carlo (SMC) is an increasingly popular alternative to MCMC, because of its speed and scalability. [42] contributed a pioneering paper of SMC focusing on tracking applications. Since then, various SMC algorithms have been proposed to obtain full posterior distributions for problems in nonlinear dynamic systems in science and engineering. To date, these algorithms for nonlinear and non-Gaussian state-space models have been successfully applied in various fields including computer vision, signal processing, tracking, control, econometrics, finance, robotics, and statistics [15, 26, 27, 71]. However, it is still an open question how to use SMC to break the computing bottleneck imposed by many combinatorial problems in computational biology, such as phylogenetic inference, inference over alignments, and gene networks.

[105] and [43] have applied SMC to a restricted class of phylogenetic models, coalescent trees. However, since ultrametric trees, such as the coalescents, put unrealistic constraints on phylogenetic inference, which prevents this method from fitting the real data. Their approach cannot be applied to standard phylogenetic models, based on non-clock trees. There has also been a large body of work apply-

ing sequential methods that are closely related to SMC, e.g., importance sampling and approximate Bayesian computation, to problems in population genetics, but these methods also focus on coalescent models ([45]; [7]; [76]; [57]; [81]).

[9] extended the application of classical SMC to phylogenetic data analysis based on partially ordered sets (posets). However, this work requires an artificial restriction on proposals, which is difficult to be satisfied for the case of non-clock trees. [109] applied sequential sampling and reweighting methods to phylogenetics, but in the context of pooling results from stratified analyses rather than constructing a single joint tree posterior.

[22, 23] proposed the SMC sampler to consider a sequence of distributions that are defined on a common continuous space. Their proposed SMC sampler uses MCMC kernels to propose particles, and they introduced artificial backward (in time) Markov kernels in the importance weight update. However, the SMC sampler in [22, 23] is focused on state-space models and it cannot be readily applied to phylogenetics.

We propose a new algorithm, the combinatorial sequential Monte Carlo (CSMC) algorithm, to approximate posteriors of phylogenetic trees given fixed evolutionary parameters. SMC is proposed to improve the efficiency of the standard MCMC methods. The technique we use to generalize SMC to non-clock trees depends on the existence of a flexible poset structure, and we present it in a level of generality directly applicable to many other combinatorial spaces. We show that the proposed CSMC algorithm is consistent and fast in simulations.

We investigate two frameworks of combining SMC and MCMC, the two main Monte Carlo tools to sample from complex and high-dimensional probability distributions, to jointly estimate phylogenetic trees and evolutionary parameters (hyperparameters). The first framework is particle Markov chain Monte Carlo (PMCMC) [3, 53], in which each MCMC iteration uses our proposed CSMC to approximate the posterior distribution of the phylogenetic trees and the marginal likelihoods of data, given parameters in the evolution model. The second algorithm is an SMC sampler, in which each SMC iteration uses an MCMC kernel to propose artificially intermediate states, which are full trees with tempering parameters.

Ideally, the evolutionary model in Bayesian phylogenetics should be flexible enough to incorporate a wide range of evolutionary phenomena, e.g. slipped strand

mispairing (SSM), a well known explanation for the evolution of repeated sequence. We propose more realistic evolutionary models by incorporating a wide range of evolutionary phenomena. Specifically, the model is a string-valued continuous time Markov Chain (SCTMC) where the exponential rates are allowed to depend on an unbounded context. We propose a novel SMC algorithm to sample the posterior distribution of the phylogenetic tree and hidden molecular sequences given the unaligned sequence data.

The rest of this dissertation is organized as follows. In Chapter 2 we provide a brief review of Monte Carlo methods, including MCMC, SMC, and PMCMC. Chapter 3 introduces backgrounds on phylogenetics, starting with the phylogenetic trees, followed by a review of methods of phylogeny reconstruction. Especially, Bayesian phylogenetics (Section 3.4.3) is necessary to understand the rest of this dissertation. Chapter 4 presents a novel SMC algorithm, CSMC, for Bayesian phylogenetic inference for non-clock trees. Note that the proposed CSMC is general enough to be applied to many other applications. In Chapter 5 we presents different ways of combining MCMC and SMC, focusing on PMCMC and an SMC sampler with MCMC kernels. In Chapter 6, we propose a general evolutionary model based on SCTMC, and we implement Bayesian phylogenetic inference using a novel sequential Monte Carlo (SMC) algorithm. Finally, Chapter 7 concludes the contributions of this dissertation and outlines some future work.

Chapter 2

Monte Carlo Methods

Monte Carlo Methods are a class of computational algorithms that make use of random samples to compute expectations. Monte Carlo Methods are commonly used to approximate intractable integrals of high dimensions. From Monte Carlo's view, any probability distribution π, regardless of its complexity, can always be represented by a discrete (Monte Carlo) sample from it. An important issue is to find a convenient and efficient discrete representation. [88] provided an excellent review on Monte Carlo statistical Methods. In this chapter, we will briefly review the major Monte Carlo Methods that will be used in the following chapters.

Throughout this chapter, π denotes the target distribution of some random variable defined on a measurable space $(E, \mathcal{E})$. Assume the density function is denoted by $\pi(x)$, $x \in E$. We are interested in computing the integral of a measurable function $\phi : E \to \mathbb{R}$ with respect to π

$$\mathbb{E}_\pi(\phi) = \int_E \phi(x)\pi(dx).$$

To simplify notation, for the most part, especially algorithms, of this dissertation we do not distinguish random variables from their realization.

2.1 Markov Chain Monte Carlo (MCMC)

Since it is typically difficult to obtain a large number of independent and identically distributed samples from a distribution of interest, π, MCMC methods [88] are

widely used to draw a dependent sample from π by constructing an ergodic Markov chain which has the desired target distribution π as its equilibrium distribution. A *Markov chain*, $\{X_n\}$, can be thought of as a sequence of random variables evolving over time that has the following property: given the present state, future transitions of the chain are independent of the past history. More precisely, for every initial distribution μ,

$$P(X_{n+1} \in A | x_0, x_1, \cdots, x_n) = P(X_{n+1} \in A | x_n) = \int_A M(x_n, dx),$$

where M is a *transition kernel* defined in a measurable space $(E, \mathcal{E})$ such that:

1. $\forall x \in E,\ M(x, \cdot)$ is a probability measure;
2. $\forall A \in \mathcal{E},\ M(\cdot, A)$ is measurable.

Providing a discrete Markov chain, $\{X_n\}$, is irreducible and aperiodic, and π is the equilibrium distribution, we have

$$P(X_n = y | X_0 = x) \to \pi(y) \text{ as } n \to \infty \text{ for all } x, y,$$

i.e. the equilibrium distribution of this Markov chain is our target distribution. Further, given the chain is irreducible, for any function $\phi(\cdot)$ such that $\mathbb{E}_\pi |\phi(X)| < \infty$,

$$\frac{1}{K} \sum_{k=1}^{K} \phi(X_k) \to \mathbb{E}(\phi(X)) \text{ as } K \to \infty,$$

with almost sure convergence in probability. In other words, this Markov chain converges to the required target distribution. Please refer to [88] for more details.

In order to construct an appropriate Markov chain, we require the transition kernel to satisfy the *detailed balance condition* defined as follows

Definition 1. *A Markov chain with transition kernel M satisfies the detailed balance condition if there exists a function f satisfying*

$$M(y, x) f(y) = M(x, y) f(x) \tag{2.1}$$

for every (x, y).

According to [88], we have the following result:

Theorem 2. *Suppose that a Markov chain with transition function M satisfies the detailed balance condition with π a probability density function. Then:*

1. *The density π is the invariant density of the chain.*
2. *The chain is π-reversible.*

2.1.1 Metropolis-Hastings (MH)

The most popular algorithm to construct appropriate Markov chains is the Metropolis-Hastings (MH) algorithm, which was first proposed by [78] and later adapted by [48]. To construct a Markov chain $\{X_n\}_{n=1,2,\cdots}$, proposal distributions $K(x,\cdot)$ are chosen to propose a candidate value for the state, y, based on the current value x with the density $K(x,y)$. The following MH acceptance probability $\alpha(x,y)$ is calculated

$$\alpha(x,y) = \min\left(1, \frac{\pi(y)K(y,x)}{\pi(x)K(x,y)}\right).$$

With probability $\alpha(x,y)$, the new state of the Markov chain is updated to y, otherwise it remains x. Algorithm 2.1 summarizes the MH algorithm. The initial value of the chain can be selected arbitrarily. The choice of $K(x,\cdot)$ is also arbitrary as long as it satisfies the requirement that the constructed Markov chain is irreducible.

Algorithm 2.1 Metropolis-Hastings Algorithm

1: Choose a starting value x_0.
2: **for** $k > 1$ **do**
3: Sample $x^* \sim K(x_{k-1},\cdot)$.
4: Compute an MH acceptance ratio $\alpha(x_{k-1}, x^*)$.
5: Accept x^* with the above probability. If x^* is not accepted, then $x_k = x_{k-1}$.
6: **end for**

We can show that the MH algorithm produces a Markov kernel which has the desired invariant distribution by showing that this Markov kernel satisfies the detailed balance condition for π. Using Algorithm 2.1, the Markov transition kernel,

$M : E \to \mathcal{E}$, is

$$\begin{aligned} M(x, dy) &= \alpha(x, y)K(x, dy) + (1 - \alpha(x, y))\delta_x(dy) \\ &= \left[1 \wedge \frac{\pi(y)K(y, x)}{\pi(x)K(x, y)}\right] K(x, dy) + \left[1 - \left(1 \wedge \frac{\pi(y)K(y, x)}{\pi(x)K(x, y)}\right)\right] \delta_x(dy), \end{aligned}$$

where δ_x is the Dirac delta function for $x \in E$. Using the MH acceptance probability $\alpha(x, y) \leq 1$, it is easy to show that

$$\begin{aligned} \pi(x)M(x, y) &= \pi(x)\left[\frac{\pi(y)K(y, x)}{\pi(x)K(x, y)}K(x, y) + \left(1 - \frac{\pi(y)K(y, x)}{\pi(x)K(x, y)}\right)\delta_x(y)\right] \\ &= \pi(y)\left[K(y, x) + \left(\frac{\pi(x)}{\pi(y)} - \frac{K(y, x)}{K(x, y)}\right)\delta_x(y)\right] \\ &= \pi(y)K(y, x) \qquad (\alpha(x, y) \leq 1 \Rightarrow \alpha(y, x) = 1) \\ &= \pi(y)\left[\alpha(y, x)K(y, x) + (1 - \alpha(y, x))\delta_y(x)\right] \\ &= \pi(y)M(y, x). \end{aligned}$$

2.1.2 Gibbs Sampler (GS)

Gibbs sampler (GS) is applicable to problems of multivariate state. The value of each element is sampled from the full conditional distribution of the element given the most recent values of all the other variables.

Let $\boldsymbol{x} = (x_1, \cdots, x_D)$ denote the vector of a multivariate random variable, and let $\boldsymbol{x}_k = (x_{k,1}, \cdots, x_{k,D})$ denote the k-th sample of $\boldsymbol{x}$. Algorithm 2.2 describes a deterministic scan GS in which the order to elements to update is fixed. In this algorithm, $\boldsymbol{x}_{k,-d} = (x_{k,1}, \cdots, x_{k,d-1}, x_{k-1,d+1}, \cdots, x_{k-1,D})$. We can also use a random scan GS in which we randomly choose the element to update.

Algorithm 2.2 Gibbs Sampling

1: Choose a starting value $\boldsymbol{x}_0$.
2: **for** $k > 1$ **do**
3: **for** $d = 1$ to D **do**
4: Sample $x_{k,d} \sim \pi(\cdot|\boldsymbol{x}_{k,-d})$
5: **end for**
6: **end for**

GS can be regarded as one special case of the MH sampler [39], in which the MH acceptance probability is always 1, shown as follows:

$$\begin{aligned}\alpha(\boldsymbol{x},(\boldsymbol{x}_{-d},y_d)) &= \frac{\pi((\boldsymbol{x}_{-d},y_d))\pi(\boldsymbol{x}_d|\boldsymbol{x}_{-d})}{\pi(\boldsymbol{x})\pi(y_d|\boldsymbol{x}_{-d})} \\ &= \frac{\pi(\boldsymbol{x}_{-d})\pi(y_d|\boldsymbol{x}_{-d})\pi(\boldsymbol{x}_d|\boldsymbol{x}_{-d})}{\pi(\boldsymbol{x}_{-d})\pi(\boldsymbol{x}_d|\boldsymbol{x}_{-d})\pi(y_d|\boldsymbol{x}_{-d})} = 1,\end{aligned}$$

where $(\boldsymbol{x}_{-d},y_d) = (x_1,\cdots,x_{d-1},y_d,x_{d+1},\cdots,x_D)$.

Since the GS requires sampling from the full conditional distributions, it might be impossible to sample from some of the full conditional distributions for many complex models. In such cases, an MH step is used to sample from the full conditional distribution. In addition, the GS can be very inefficient when the variables are closely correlated or sampling from the full conditionals is extremely expensive or inefficient.

2.2 Sequential Monte Carlo (SMC)

The sequential Monte Carlo (SMC) method is an increasingly popular alternative to MCMC. In this section, we first review the generic SMC algorithm and use the application to state space models as a specific example. Then, we summarize some resampling methods used in SMC. Finally, we shall introduce an SMC sampler that is relevant to our work.

2.2.1 Generic SMC

In this section, we will briefly review the generic SMC algorithms to sample from a given target distribution π defined on a measurable space $(E,\mathcal{E})$; the density function is denoted $\pi(\boldsymbol{x})$, $\boldsymbol{x}\in E$. This method requires us to introduce a sequence of intermediate probability distributions

$$\{\pi_r(\boldsymbol{x}_r), r = 1,\cdots,R\} \tag{2.2}$$

defined on a sequence of measurable spaces $\{(E_r,\mathcal{E}_r), r = 1,\cdots,R\}$ such that $\pi_R(\boldsymbol{x}_R) = \pi(\boldsymbol{x})$. Here r is a general index for the intermediate distributions which is not necessary to be an index for time as in state space models. Typically, $\boldsymbol{x}_r = x_{1:r} =$

$(x_1, \cdots, x_r)$, where $x_r \in \mathcal{X}$ implying that $E_r = \mathcal{X}^r$.

Each density of the intermediate distributions is assumed known up to a normalizing constant; i.e. for $r = 1, \cdots, R$,

$$\pi_r(\boldsymbol{x}_r) = Z_r^{-1} \gamma_r(\boldsymbol{x}_r),$$

where $\gamma_r : \mathcal{E}_r \to \mathbb{R}^+$ can be evaluated pointwise, but the normalizing constant Z_r is unknown. We will denote Z_R by Z.

A concrete example of the general applications of SMC is state space models [6, 10, 10, 11, 30, 31, 41, 41, 42], in which r is an index for time; the intermediate distributions are given by state space models themselves, i.e.

$$\pi_r(\boldsymbol{x}_r) = p(x_{1:r}|y_{1:r})$$

defined on the product space $E_r = \mathcal{X}^r$, where $x_{1:r}$ are from a hidden $\mathcal{X}$-valued discrete-time Markov process, and $y_{1:r}$ are the observations. The unnormalized density can be evaluated pointwise as follows

$$\gamma_r(\boldsymbol{x}_r) = p(x_{1:r}, y_{1:r}) = \mu(x_1) \prod_{n=2}^{r} f(x_n|x_{n-1}) \prod_{n=1}^{r} g(y_n|x_n),$$

where $f(x_n|x_{n-1})$ are the transition probability densities, and $g(y_n|x_n)$ are the marginal densities of the observations.

We wish to estimate expectations of test functions $\phi_r : E_r \to \mathbb{R}$

$$\mathbb{E}_{\pi_r}(\pi_r) = \int \phi_r(\boldsymbol{x}_r) \pi_r(d\boldsymbol{x}_r),$$

and the normalizing constants Z_r. In SMC algorithms, we do this sequentially; i.e. we first estimate π_1 and Z_1 at iteration 1, then π_2 and Z_2 at iteration 2 and so on.

A generic SMC algorithm is described in Algorithm 2.3. At the r-th iteration of the algorithm, the proposal distributions q_r are used to propose the particles, $\{\boldsymbol{x}_{r,k}\}_{k=1,\cdots,K}$, of the current iteration based on the particles, $\{\tilde{\boldsymbol{x}}_{r-1,k}\}$, from the previous iteration. The intermediate distribution π_r is approximated by the set of K weighted samples (also called particles), $\{\boldsymbol{x}_{r,k}, W_{r,k}\}_{k=1,\cdots,K}$, where $\boldsymbol{x}_{r,k}$ denotes the

Algorithm 2.3 Generic Sequential Monte Carlo

At iteration $r = 1$,

sample $x_{1,k} \sim q_1(\cdot)$

set its unnormalized weight $w_{1,k} = \gamma_1(x_{1,k})/q_1(x_{1,k})$.

normalize weights $W_{1,k} = w_{1,k}/\sum_{k=1}^{K} w_{1,k}$

resample $\{x_{1,k}, W_{1,k}\}$ to obtain new particles denoted $\{\tilde{x}_{1,k}\}$

At iteration $r \geq 2$,

sample $x_{r,k} \sim q_r(\tilde{\boldsymbol{x}}_{r-1,k} \rightarrow \cdot)$, and $\boldsymbol{x}_{r,k} = (\tilde{\boldsymbol{x}}_{r-1,k}, x_{r,k})$

compute

$$w_{r,k} = w(\tilde{\boldsymbol{x}}_{r-1,k}, \boldsymbol{x}_{r,k}) = \frac{\gamma_r(\boldsymbol{x}_{r,k})}{\gamma_{r-1}(\tilde{\boldsymbol{x}}_{r-1,k}) q_r(\tilde{\boldsymbol{x}}_{r-1,k} \rightarrow x_{r,k})}$$

normalize weights $W_{r,k} = w_{r,k}/\sum_{k=1}^{K} w_{r,k}$

resample $\{\boldsymbol{x}_{r,k}, W_{r,k}\}$ to obtain new particles denoted $\{\tilde{\boldsymbol{x}}_{r,k}\}$

k-th particle and $W_{r,k}$ represents its normalized weight correspondingly, i.e.

$$W_{r,k} = \frac{w_{r,k}}{\sum_{k=1}^{K} w_{r,k}}.$$

We have the following approximation of $\pi_r(\boldsymbol{x}_r)$,

$$\pi_{r,K}(d\boldsymbol{x}_r) = \sum_{k=1}^{K} W_{r,k} \delta_{\boldsymbol{x}_{r,k}}(d\boldsymbol{x}_r).$$

A byproduct of the SMC algorithm is an estimate of the normalizing constant Z. We can rewrite the first constant normalizing constant as

$$Z_1 = \int \frac{\gamma_1(x_1)}{q_1(x_1)} q_1(x_1) dx_1 = \int w_1(x_1) q_1(x_1) dx_1.$$

Correspondingly, an estimate of Z_1 is

$$Z_{1,K} = \frac{1}{K} \sum_{k=1}^{K} w_{1,k}.$$

Similarly, we can rewrite the ratio of the normalizing constants as

$$\begin{aligned}
\frac{Z_r}{Z_{r-1}} &= \frac{\int \gamma_r(\boldsymbol{x}_r) d\boldsymbol{x}_r}{Z_{r-1}} = \frac{\int \gamma_r(\boldsymbol{x}_r) d\boldsymbol{x}_r}{\gamma_{r-1}(\boldsymbol{x}_{r-1})/\pi_{r-1}(\boldsymbol{x}_{r-1})} \\
&= \int \frac{\gamma_r(\boldsymbol{x}_r)}{\gamma_{r-1}(\boldsymbol{x}_{r-1})} \pi_{r-1}(\boldsymbol{x}_{r-1}) d\boldsymbol{x}_r \\
&= \int \frac{\gamma_r(\boldsymbol{x}_r)}{\gamma_{r-1}(\boldsymbol{x}_{r-1}) q_r(\boldsymbol{x}_{r-1} \to x_r)} \pi_{r-1}(\boldsymbol{x}_{r-1}) q_r(\boldsymbol{x}_{r-1} \to x_r) d\boldsymbol{x}_r \\
&= \int w_r(\boldsymbol{x}_r) \pi_{r-1}(\boldsymbol{x}_{r-1}) q_r(\boldsymbol{x}_{r-1} \to x_r) d\boldsymbol{x}_r.
\end{aligned}$$

Straightforwardly, an estimate of Z_r/Z_{r-1} is provided by

$$\widehat{\frac{Z_r}{Z_{r-1}}} = \frac{1}{K} \sum_{k=1}^{K} w_{r,k}.$$

Since the estimate of the normalizing constant can be rewritten as

$$Z \equiv Z_R = Z_1 \prod_{r=2}^{R} \frac{Z_r}{Z_{r-1}},$$

an estimate of the normalizing constant Z is

$$Z_{R,K} = \prod_{r=1}^{R} \left(\frac{1}{K} \sum_{k=1}^{K} w_{r,k} \right), \tag{2.3}$$

which can be obtained from an SMC algorithm readily. Moreover, Equation (2.3) is a consistent estimate of Z [21, 26].

The performance of SMC algorithms can be improved by using sophisticated resampling methods, strategically designing the sequence of target distributions, and carefully choosing the proposal distributions. In the next section, we will only briefly review some resampling methods.

2.2.2 Resampling Methods

As the iteration index increases, the variance of the unnormalized weights $\{w_{r,k}\}$ tend to increase and all the mass is concentrated on a few number of particles. A

resampling step is used to reset the approximation by pruning the particles with low weights and multiply the particles with high weights. The rationale is that if a particle at iteration r has a low weight then often it will still have a low weight at iteration $r+1$. We want to focus our computational efforts on the promising particles.

Resampling at every iteration is neither necessary nor efficient. We need a resampling schedule to determine when we implement the resampling step. A resampling schedule can be either deterministic or dynamic. In a deterministic schedule one conducts resampling at iteration $r_0, 2r_0, \cdots$, where r_0 depends on a particular problem. In a dynamic schedule, we resample only when the variation of the weights is higher than some prefixed threshold.

We can use two criteria, the effective sample size (ESS) or the coefficient of variation (CV), to measure the variation of the weights. ESS can be computed using

$$ESS = \left(\sum_{k=1}^{K} W_{r,k}^2 \right)^{-1}.$$

If $W_{r,k} = 1/K$ for any k, we have $ESS = K$; if $W_{r,i} = 1$ and $W_{r,j} = 0$ for $j \neq i$, we have $ESS = 0$. CV can be computed using

$$CV = \left(\frac{1}{K} \sum_{k=1}^{K} (KW_{r,k} - 1)^2 \right)^{1/2}$$

If $W_{r,k} = 1/K$ for any k, we have $CV = 0$; if $W_{r,i} = 1$ and $W_{r,j} = 0$ for $j \neq i$, we have $CV = \sqrt{K-1}$.

In the following, we will summarize two commonly-used resampling methods, multinomial resampling and stratified resampling.

Multinomial Resampling

The simplest resampling scheme is multinomial resampling which samples K times $\tilde{\boldsymbol{x}}_{r,k} \sim \pi_{r,K}(d\boldsymbol{x}_r)$ to build the new approximation

$$\tilde{\pi}_{r,K}(d\boldsymbol{x}_r) = \frac{1}{K}\sum_{k=1}^{K} \delta_{\tilde{\boldsymbol{x}}_{r,k}}(d\boldsymbol{x}_r).$$

We can rewrite the above by

$$\tilde{\pi}_{r,K}(d\boldsymbol{x}_r) = \sum_{k=1}^{K} \frac{N_{r,k}}{K} \delta_{\tilde{\boldsymbol{x}}_{r,k}}(d\boldsymbol{x}_r),$$

where $(N_{r,1}, \cdots, N_{r,K}) \sim \text{Multinomial}(K; W_{r,1}, \cdots, W_{r,K})$ thus

$$\mathbb{E}[N_{r,k}] = KW_{r,k}, \mathbf{V}ar[N_{r,k}] = KW_{r,k}(1 - W_{r,k}).$$

Stratified Resampling

A better resampling method can be designed such that

$$\mathbb{E}[N_{r,k}] = KW_{r,k}, \mathbf{V}ar[N_{r,k}] < KW_{r,k}(1 - W_{r,k}).$$

Stratified resampling, described in Algorithm 2.4, is a popular alternative to multinomial resampling.

Algorithm 2.4 Stratified Resampling

Normalise the weights $\{w_{r,k}\}$ of the K particles and label them according to the order of the corresponding $\boldsymbol{x}_{r,k}$ to obtain $W_{r,(k)}, k = 1, \cdots, K$.
Construct the corresponding cumulative distribution function: for $k = 1, \cdots, K$,

$$Q_r(0) \equiv 0, \quad Q_r(k) \equiv \sum_{j \le k} W_{r,(j)},$$

Sample U_1 uniformly on $[0, 1/K]$ and set $U_j = U_1 + (j-1)/K$ for $j = 2, \cdots, K$.
For $k = 1, \cdots, K$, if there exists $j \in \{1, \cdots, K\}$ such that $Q_r(k-1) \le U_j < Q_r(k)$, then $\boldsymbol{x}_{r,k}$ survives. Update $N_{r,k} = \#\{U_j : Q_r(k-1) \le U_j < Q_r(k)\}$.

2.2.3 Sequential Monte Carlo Samplers

Recent work on SMC has focused on the problem of sampling from arbitrary sequences of distributions. The SMC sampler framework proposed by [22, 23] is a very general method for obtaining a set of samples from a sequence of distributions which can exist on the same or different spaces. This is a generalization of the standard SMC method [27] in which the target distribution exists on a space of strictly increasing dimension.

[22, 23] mainly addressed the case of the sequence of target distributions $\{\pi_r\}$ that are defined on a common continuous space $\mathcal{X}$, e.g. π_r is the posterior distribution of a parameter given the data collected until time r, i.e. $\pi_r(x) = p(x|y_{1:r})$. This SMC sampler can be obtained by defining a sequence of distributions that admit the distribution of interest, $\pi_r(x_r)$, as the recent iteration marginal

$$\tilde{\pi}_r(\boldsymbol{x}_r) = \pi_r(x_r) \prod_{j=1}^{r-1} L_j(x_{j+1}, x_j),$$

where $L_j(x_{j+1}, x_j)$ is the artificial backward Markov kernels from iteration $j+1$ to j. Then we apply the standard SMC on this sequence of distributions. Sample at iteration r,

$$x_{r,k} \sim K_r(x_{r-1,k}, \cdot),$$

where K_r is a Markov kernel defined on $E_{r-1} \times \mathcal{E}_r$. The resulting sampler has a weight update

$$W_{r,k} \propto \frac{\pi_r(x_{r,k}) L_{r-1}(x_{r,k}, x_{r-1,k})}{\pi_{r-1}(x_{r,k}) K_r(x_{r-1,k}, x_{r,k})},$$

which is different from the one in a standard SMC.

Algorithm 2.5 summarizes the SMC sampler. A common approach in SMC samplers is to choose $K_r(x_{r-1}, x_r)$ to be π_r-invariant, typically MCMC kernels. A convenient backward Markov kernel that allows an easy evaluation of the importance weight is

$$L_{r-1}(x_r, x_{r-1}) = \frac{\pi_r(x_{r-1}) K_r(x_{r-1}, x_r)}{\pi_r(x_r)}.$$

Algorithm 2.5 An SMC Sampler

sample $x_{1,k} \sim q_1(\cdot)$
set its unnormalized weight $w_{1,k} = \gamma_1(x_{1,k})/q_1(x_{1,k})$.
normalize weights $W_{1,k} = w_{1,k}/\sum_{k=1}^{K} w_{1,k}$
resample $\{x_{1,k}, W_{1,k}\}$ to obtain new particles denoted $\{\tilde{x}_{1,k}\}$
for $r \in 2, \ldots, R$ **do**
  sample $x_{r,k} \sim K_r(\tilde{x}_{r-1,k}, \cdot)$
  compute

$$w_{r,k} = w(\tilde{x}_{r-1,k}, x_{r,k}) = \frac{\gamma_n(x_{r,k})}{\gamma_{r-1}(\tilde{x}_{r-1,k})} \cdot \frac{L_{r-1}(x_{r,k}, \tilde{x}_{r-1,k})}{K_r(\tilde{x}_{r-1,k}, x_{r,k})}$$

  normalize weights $W_{r,k} = w_{r,k}/\sum_{k=1}^{K} w_{r,k}$
  resample $\{x_{r,k}, W_{r,k}\}$ to obtain new particles denoted $\{\tilde{x}_{r,k}\}$
end for

With this backward kernel, the incremental importance weight becomes

$$\begin{aligned} w_r &= w(x_{r-1}, x_r) = \frac{\gamma_r(x_r)}{\gamma_{r-1}(x_{r-1})} \cdot \frac{L_{r-1}(x_r, x_{r-1})}{K_r(x_{r-1}, x_r)} \\ &= \frac{\gamma_r(x_r)}{\gamma_{r-1}(x_{r-1})} \cdot \frac{\pi_r(x_{r-1}) K_r(x_{r-1}, x_r)}{\pi_r(x_r)} \cdot \frac{1}{K_r(x_{r-1}, x_r)} \\ &= \frac{Z_r \pi_r(x_{r-1})}{\gamma_{r-1}(x_{r-1})} = \frac{\gamma_r(x_{r-1})}{\gamma_{r-1}(x_{r-1})}. \end{aligned}$$

2.3 Particle MCMC

[3] proposed a whole class of efficient and flexible MCMC algorithms, named particle Markov chain Monte Carlo methods (PMCMC), to approximate a target distribution, π. Typically, the target distribution is a posterior distribution; i.e. $\pi(\boldsymbol{x}) \equiv p(\boldsymbol{x}|\mathcal{Y})$. The PMCMC algorithms combine respective strengths of MCMC and SMC algorithms by using SMC methods to build efficient high dimensional proposal distributions at each of the MCMC steps. Particle MCMC can make bold moves in exploring the parameter space. The PMCMC algorithms include the particle independent Metropolis-Hastings (PIMH), particle marginal Metropolis-Hastings sampler (PMMH), and particle Gibbs sampler (PGS).

2.3.1 Particle Independent Metropolis-Hastings (PIMH)

A standard independent Metropolis-Hastings (IMH) sampler targeting $\pi(\boldsymbol{x}) \equiv p(\boldsymbol{x}|\mathcal{Y})$ uses a proposal distribution $q(\cdot)$ to propose candidates $\boldsymbol{x}^*$ independent of the current state $\boldsymbol{x}$, and accepts $\boldsymbol{x}^*$ with probability

$$\min\left\{1, \frac{\pi(\boldsymbol{x}^*)}{\pi(\boldsymbol{x})} \cdot \frac{q(\boldsymbol{x})}{q(\boldsymbol{x}^*)}\right\}.$$

The particle independent Metropolis-Hastings (PIMH) algorithm uses SMC approximations of $\pi(\boldsymbol{x})$ as a proposal [3], i.e. using $q(\boldsymbol{x}) = \pi_{R,K}(\boldsymbol{x})$. After we obtain a set of weighted samples $\{(W_{R,k}, \boldsymbol{x}_{R,k})\}_{k=1,\cdots,K}$ using SMC, we draw the proposed sample using a resampling method (see Section 2.2.2) from

$$\pi_{R,K}(d\boldsymbol{x}_r) = \sum_{k=1}^{K} W_{r,k}\delta_{\boldsymbol{x}_{r,k}}(d\boldsymbol{x}_r).$$

The resulting PIMH sampler has a simple form as described in Algorithm 2.6.

Generally, PIMH and SMC perform similarly with respect to estimate accuracy given the same running time. Actually, for many cases, it is preferable to use SMC [3, 53]. However, the memory limit of our computers might prevent us from running a large number of particles which is required for a complicated model, and consequently the estimate may not be accurate. In contrast, the benefit of using PIMH is its nature of iteration, which allows us to run it for any specified amount of time. We could run PIMH until a sufficient number of particles have been generated.

2.3.2 Particle Marginal Metropolis-Hastings Sampler (PMMH)

We are interested in sampling from a joint distribution of static parameters $\theta \in \Theta$ and a high dimensional variable of interest $\boldsymbol{x} \in E$

$$\pi(\theta, \boldsymbol{x}) = \frac{\gamma(\theta, \boldsymbol{x})}{Z},$$

Algorithm 2.6 The Particle Independent Metropolis-Hastings Algorithm

1. Initialization, $i = 0$,
   (a) run an SMC algorithm targeting $\pi(\boldsymbol{x})$,
   (b) sample $\boldsymbol{x}(0) \sim \pi_{R,K}(\cdot)$ and compute $Z_{R,K}(0)$
2. For iteration $i \geq 1$,
   (a) run an SMC algorithm targeting $\pi(\boldsymbol{x})$,
   (b) sample $\boldsymbol{x}^* \sim \pi_{R,K}(\cdot)$ and calculate $Z^*_{R,K}$ using (2.3),
   (c) with probability

$$\min\left\{1, \frac{Z^*_{R,K}}{Z_{R,K}(i-1)}\right\},$$

   set $\boldsymbol{x}(i) = \boldsymbol{x}^*$ and $Z_{R,K}(i) = Z^*_{R,K}$; otherwise set $\boldsymbol{x}(i) = \boldsymbol{x}(i-1)$ and $Z_{R,K}(i) = Z_{R,K}(i-1)$.

where $\gamma : \Theta \times E \to \mathbb{R}^+$ is known pointwise. As an example, there is often some static parameter $\theta \in \Theta$ in the state space models mentioned in Section 2.2.1. More precisely, the unnormalized target distribution is

$$\gamma(\theta, \boldsymbol{x}) = p(\theta, x_{1:R}, y_{1:R}) = \mu_\theta(x_1) \prod_{n=2}^{R} f_\theta(x_n|x_{n-1}) \prod_{n=1}^{R} g_\theta(y_n|x_n).$$

Consider an MH algorithm of target density $\pi(\theta, \boldsymbol{x})$ and proposal density

$$q((\theta, \boldsymbol{x}) \to (\theta^*, \boldsymbol{x}^*)) = q(\theta \to \theta^*)\pi(\boldsymbol{x}^*|\theta^*),$$

Then the resulting MH acceptance ratio is given by

$$\begin{aligned} & \min\left(1, \frac{\pi(\theta^*, \boldsymbol{x}^*)}{\pi(\theta, \boldsymbol{x})} \frac{q((\theta^*, \boldsymbol{x}^*) \to (\theta, \boldsymbol{x}))}{q((\theta, \boldsymbol{x}) \to (\theta^*, \boldsymbol{x}^*))}\right) \\ = & \min\left(1, \frac{\pi(\theta^*, \boldsymbol{x}^*)}{\pi(\theta, \boldsymbol{x})} \frac{q(\theta^* \to \theta)\pi(\boldsymbol{x}|\theta)}{q(\theta \to \theta^*)\pi(\boldsymbol{x}^*|\theta^*)}\right) \\ = & \min\left(1, \frac{\pi(\theta^*)}{\pi(\theta)} \frac{q(\theta^* \to \theta)}{q(\theta \to \theta^*)}\right) \end{aligned}$$

where $\pi(\theta)$ is the marginal distribution.

To build a proposal approximating $\pi(\boldsymbol{x}^*|\theta^*)$, we use an SMC algorithm using a sequence of distributions π_r. PMMH utilizes the fact that the normalizing constant estimate obtained at iteration R of SMC is an estimate of the unnormalized $\pi(\theta)$, denoted $\gamma(\theta)$. Algorithm 2.7 shows the PMCMC algorithm for sampling from $\pi(\theta, \boldsymbol{x})$.

Algorithm 2.7 The Particle Marginal Metropolis-Hastings Sampler

Step 1: initialization, $i = 0$,
set $\theta(0)$ arbitrarily and
run an SMC algorithm targeting $\pi(\boldsymbol{x}|\theta(0))$,
sample $\boldsymbol{x}(0) \sim \pi_{R,K}(\cdot|\theta(0))$ and let $\gamma_{R,K}(\theta(0)) = Z_{R,K}$

Step 2: for iteration $i \geq 1$,
sample $\theta^* \sim q(\theta(i-1) \to \cdot)$,
run an SMC algorithm targeting $\pi(\boldsymbol{x}|\theta^*)$,
sample $\boldsymbol{x}^* \sim \pi_{R,K}(\cdot|\theta^*)$ and $\gamma_{R,K}(\theta^*) = Z_{R,K}$
with probability

$$\min\left(1, \frac{\gamma_{R,K}(\theta^*)}{\gamma_{R,K}(\theta(i-1))} \frac{q\{\theta^* \to \theta(i-1)\}}{q\{\theta(i-1) \to \theta^*\}}\right) \tag{2.4}$$

set $\theta(i) = \theta^*$, $\boldsymbol{x}(i) = \boldsymbol{x}^*$, and $\gamma_{R,K}(\theta(i)) = \gamma_{R,K}(\theta^*)$; otherwise set $\theta(i) = \theta(i-1)$, $\boldsymbol{x}(i) = \boldsymbol{x}(i-1)$, and $\gamma_{R,K}(\theta(i)) = \gamma_{R,K}(\theta(i-1))$.

2.3.3 Particle Gibbs Sampler (PGS)

A standard Gibbs sampler for sampling from $\pi(\theta, \boldsymbol{x})$ iteratively samples from the full conditionals $\pi(\theta|\boldsymbol{x})$ and $\pi(\boldsymbol{x}|\theta)$. Sampling from $\pi(\theta|\boldsymbol{x})$ is relatively easy. In

many cases it is possible to sample exactly from $\pi(\theta|\boldsymbol{x})$. Otherwise an MH step of invariant density $\pi(\theta|\boldsymbol{x})$ can be used. In contrast, since $\boldsymbol{x}$ of interest is typically high dimensional, e.g. a large phylogenetic tree, sampling from $\pi(\boldsymbol{x}|\theta)$ might be very challenging.

The particle Gibbs sampler (PGS) approximates the standard Gibbs sampler using a special type of PMCMC update called the *conditional SMC* update to sample from $\pi(\boldsymbol{x}|\theta)$. The conditional SMC, described in Algorithm 2.8, is similar to a standard SMC algorithm but it ensures that a prespecified particle and its ancestral lineage survive and samples the remaining $K-1$ particles as usual. Suppose the j-th particle is the "frozen" one and its ancestral lineage is denoted $(A_1^j, A_2^j, \cdots, A_R^j)$, where A_r^j represents the index of the "parent" at iteration $r-1$ of particle $\boldsymbol{x}_{r,j}$ for $r = 2, \cdots, R$. The PGS is described in Algorithm 2.9.

Algorithm 2.8 Conditional SMC

At iteration 1

For $k \neq A_1^j$, sample $x_{1,k} \sim q_1(\cdot)$, and set its unnormalized weight

$$w_{1,k} = \frac{\gamma_\theta(x_{1,k})}{q_1(x_{1,k})},$$

Resample $K-1$ times from $\{x_{1,k}, w_{1,k}\}$ to obtain $\{\tilde{x}_{1,k} : k \neq A_1^j\}$ and set $\tilde{x}_{1,A_1^j} = x_{1,A_1^j}$.

At iteration $r \geq 2$,

For $k \neq A_r^j$,

Sample $x_{r,k} \sim q_r(\tilde{\boldsymbol{x}}_{r-1,k} \to \cdot)$, set $\boldsymbol{x}_{r,k} = (\tilde{\boldsymbol{x}}_{r-1,k}, x_{r,k})$,

Compute

$$w_{r,k} = \frac{\gamma_\theta(\boldsymbol{x}_{r,k})}{\gamma_\theta(\tilde{\boldsymbol{x}}_{r-1,k}) q_r(\tilde{\boldsymbol{x}}_{r-1,k} \to x_{r,k})},$$

Resample $\{\boldsymbol{x}_{r,k}, w_{r,k}\}$ to obtain new particles denoted $\{\tilde{\boldsymbol{x}}_{r,k} : k \neq A_r^j\}$ and set $\tilde{\boldsymbol{x}}_{r,A_r^j} = \boldsymbol{x}_{r,A_r^j}$.

Algorithm 2.9 The Particle Gibbs Sampler

Initialization: $i = 0$
  Sample $\theta(0)$ arbitrarily
  Run an SMC algorithm targeting $\pi(\boldsymbol{x}|\theta(0))$
  Sample $\boldsymbol{x}(0) \sim \pi_{R,K}(\cdot|\theta(0))$ and record its ancestral lineage.
for $i \geq 1$ **do**
  Sample $\theta(i) \sim \pi(\cdot|\boldsymbol{x}(i-1))$
  Run a conditional SMC algorithm targeting $\pi(\boldsymbol{x}|\theta(i-1))$ conditional on $\boldsymbol{x}(i-1)$ and its ancestral lineage.
  Sample $\boldsymbol{x}(i) \sim \pi_{R,K}(\cdot|\theta(i-1))$.
end for

Chapter 3

Background on Phylogenetics

3.1 Phylogenetic Tree

Let X be a set of observed taxa, typically modern species, each of which is represented by a biological sequence, a string of symbols or characters from a finite alphabet Σ. For example, the alphabet for DNA sequences is the set of four nucleotides $\{A, C, G, T\}$. A phylogenetic X-tree represents the relationship among taxa via a *tree topology*, which is represented by a connected acyclic graph, (V, E), where V is the set of vertices for the observed and unobserved taxa and E is the set of edges between taxa. The vertices in X are called *external nodes* or *leaves*, which are nodes with only one neighbour. Note that in a tree any two vertices are connected by exactly one simple path. In other words, a tree is a connected graph without cycles. Further, a *forest* is a disjoint union of trees.

In this dissertation, we only consider *binary phylogenetic X-trees*, which is also the main type of trees studied in the phylogenetics literature. Each vertex of a binary phylogenetic tree has a maximum of three neighbors. A binary phylogenetic tree can be rooted or unrooted. Figure 3.1 shows an example of a rooted tree and an unrooted tree of 5 taxa. A *rooted phylogenetic tree* is a directed tree with a unique node corresponding to the most recent common ancestor of all the leaf taxa of the tree. In a rooted binary tree, vertices with one, two, and three neighbors correspond to leaves, root, and *internal nodes*, respectively. In contrast, an *unrooted phylogenetic tree* represents the relatedness of leaf taxa without making assumptions about

their common ancestor. In an unrooted binary tree, each vertex has either one or three neighbors.

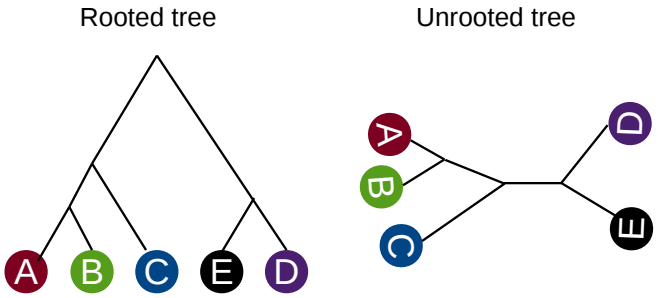

Figure 3.1: Examples of a rooted tree and an unrooted tree.

A rooted phylogenetic X-tree describes the evolution of a set of species in the tree from a common ancestor at the root. We regard the edges as being directed away from the root. *Branch lengths* are positive real numbers associated with each edge, specifying the amount of evolution between nodes.

We consider the *ultrametric* (or *clock*) trees where the leaf nodes are all equally distant from the root. If a tree is a molecular clock, its edge lengths can be specified corresponding to distances among the taxa. Since a binary tree with n leaves has $n-1$ internal nodes including the root, the branch lengths of a clock tree are determined by the $n-1$ distances among the taxa. Typically, the branch lengths of a clock tree are reparameterized as distances between heights from the leaves, shown in Figure 3.2.

Although a clock tree is a convenient representation of the evolutionary relationship among taxa, it is too restricted and unrealistic because it implies a constant evolutionary rate. In contrast, a *non-clock tree* refers to a more general tree without these constraints. A non-clock tree is typically represented by an unrooted tree. However, most evolutionary models are reversible models in which the likelihood for an unrooted tree is equal to the likelihood of a rooted tree by rooting at an arbi-

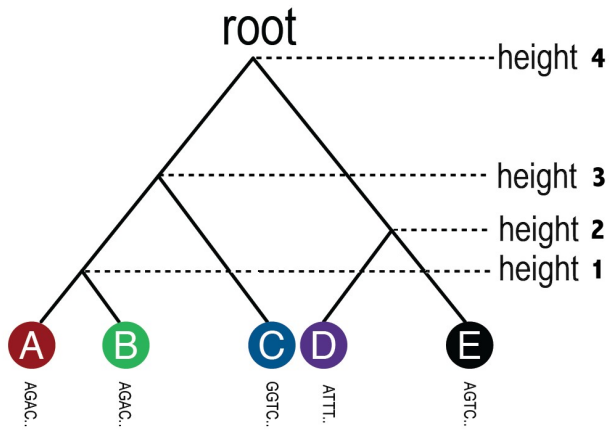

Figure 3.2: Heights of subtrees on a clock tree. Height 4 is the height of this clock tree.

trary point in the tree. For a non-clock tree, the branch lengths are determined by $2(n-1)$ values.

3.2 Data for Phylogenetics

3.2.1 Data Types

Although all kinds of characteristics of species can be used for phylogenetic reconstruction, modern technologies have allowed us to study the evolutionary relationship among species at a molecular level. *Deoxyribonucleic acid* (DNA), *Ribonucleic acid* (RNA), and proteins are the three major macromolecules that are essential for all known forms of life. Therefore, the commonly-used types of data for phylogeny analysis include DNA, RNA, and amino acid data. In this dissertation, we reconstruct the phylogenetic relationship among the taxa using the DNA sequences or RNA sequences to illustrate the performance of our methods. But the methodologies can be easily adapted to any types of biological sequence data.

Usually DNA is a right-handed helix composed of two complementary chains twisted around each other. Each chain is a linear polynucleotide consisting of four nucleotides, two purines: adenine (A) and guanine (G), and two pyrimidines: cytosine (C) and thymine (T).

The chemical structure of RNA is very similar to that of DNA, but RNA contains uracil (U) instead of thymine (T); i.e. RNA is made up of a long chain of nucleotides from the set $\{A, U, G, C\}$. In addition, most RNA molecules are single-stranded and can adopt very complex three-dimensional structures.

3.2.2 Aligned Data via Multiple Sequence Alignments

The common methods of reconstructing phylogenetic trees are based on a fixed single multiple alignment, in which the homologous nucleotides aligned in columns using multiple sequence alignment (MSA) algorithms, e.g. ClustalW [44, 68] and T-coffee [12].

We use $\mathcal{Y}$ to denote the set of observed molecular sequences for an X-tree. For simplicity, we will assume that the observation, $\mathcal{Y}$, takes the form of a matrix $y_{x,s}$, where $x \in X$ and s denotes an aligned position on the genomes, called a site. In this dissertation, we assume the sites are independent.

3.2.3 Datasets

The DNA sequences we will analyze are aligned protein coding mitochondrial DNA sequences obtained from 32 species of cichlid fishes [17, 64]. Each DNA sequence consists of 1047 sites. Identical nucleotides are observed on 569 sites, and we use the nucleotides on the remaining 478 sites for phylogenetic tree construction.

In this dissertation, we will use *ribosomal RNA* (rRNA) sequence data arbitrarily chosen from the Comparative RNA Web (CRW) Site (http://www.rna.ccbb.utexas.edu) which contains RNA sequences and structure information, results of data analysis, interpretation, etc [13].

3.3 Score-based Methods for Phylogeny Reconstruction

During the past several decades, researchers have proposed many methodologies to reconstruct evolutionary histories using molecular sequence data. [52] provided a good review of these methods. In this section, we will briefly introduce two score-based methods, neighbor joining (NJ) [37, 92, 98] and maximum parsimony [36, 75, 93]. We will introduce the model-based methods for phylogeny reconstruction

in Section 3.4.

3.3.1 Neighbour-joining Method

The neighbor joining (NJ) method is a bottom-up clustering method used for reconstructing phylogenetic trees using the distance between each pair of taxa in the tree [52, 92]. The evolutionary distance represents the number of changes, e.g. A→G, that have occurred along the branches between two sequences.

The main advantage of NJ is that it can be implemented in a polynomial time, making it capable of handling massive datasets. The main drawbacks of NJ are rooted in the inaccuracy of approximating evolutionary distances by observed differences between sequences. Please refer to [52] for details.

3.3.2 Maximum Parsimony

The *maximum parsimony method* searches all possible tree topologies to choose the tree that requires the fewest possible mutations to explain the data. The classical parsimony problem has direct connections with graph theory and enumerative combinatorics. To find the most parsimonious tree, we must have a way of calculating how many changes of state are needed on a given tree. The most famous algorithms for Maximum Parsimony are Fitch's algorithm [36] and Sankoff's algorithm [93].

The advantages of the maximum parsimony method include that it is fast enough for the analysis of hundreds of sequences and it is robust if branches are short [52]. The main disadvantage is that maximum parsimony can be inconsistent in the case of 'long-branch attraction' [8, 32]. In other words, two long branches that are not adjacent on the true tree are inferred to be the closest relatives of each other by parsimony. Another drawback is that it makes unrealistic assumption that the number of mutations is to be equal on all branches of a tree.

3.4 Model-based Methods for Phylogeny Reconstruction

Because the weight or cost parameters in score-based methods cannot be easily estimated and interpreted, they are replaced by stochastic modelling of nucleotide, codon, or amino acid evolution. The advantages of stochastic modelling include

explicit models of evolution, meaningful parameters, and allowance for hypothesis testing and model comparison. The parameters can be estimated by maximum likelihood [33, 46, 62] or Bayesian techniques [67, 70, 77, 82, 113]. Both the maximum likelihood method and Bayesian phylogenetics are heavily dependent on the probabilistic model of evolution. We will first introduce stochastic evolutionary models and then provide more details on Bayesian phylogenetics.

3.4.1 Stochastic Evolutionary Model

continuous time Markov chain (CTMC) over characters

The standard technique to specify the transition probability of a node v in a graphical model given its parent $\varrho(v)$ is to use a *stochastic process*. A stochastic process is a collection of random variables indexed by a set $S, \{Y_s, s \in S\}$. In phylogenetics, we use a stochastic process called a CTMC, where $S = [0, T]$ represents the time interval for a large number of successive generations and the individual variables Y_s are assumed to have a countable domain Σ, e.g. $\Sigma = \{A, C, G, T\}$ for DNA data.

The probability of transitioning from $Y_0 = i$ to $Y_T = j$ along a branch of length T is denoted by the (i, j)-th element, $P_{i,j}(T)$, of a matrix $P(T)$. The matrix of transition probabilities P satisfies the matrix differential equation $P' = PQ$ with initial condition $P(0) = I$, where I is the identity matrix, and Q is called the *rate matrix*.

Under the Markov assumption, $P(T)$ must satisfy $P(T + T^*) = P(T)P(T^*)$. This implies the solution of $P' = PQ$ is

$$P(T) = \exp(QT) = I + \sum_{i=1}^{\infty} Q^i T^i / i!, \tag{3.1}$$

where exp is the *matrix exponential*, and I denotes the $|\Sigma| \times |\Sigma|$ identity matrix.

In practice, the matrix exponential is computed by diagonalization. Denote $\lambda_1, \cdots, \lambda_{|\Sigma|}$ as the eigenvalues of Q, and denote U as the orthogonal matrix in which the columns are composed of the corresponding eigenvectors. Using the diagonal-

ization $Q = U\Lambda U^{-1}$, Equation (3.1) becomes

$$P(T) = \exp(QT) = U\text{diag}(\exp(\lambda_1 T, \lambda_2 T, \cdots, \lambda_{|\Sigma|} T))U^{-1}.$$

There are many evolutionary models for the rate matrix of DNA or RNA sequences. The HKY85 model introduced by [47] is one of the popular ones. In HKY85, the rate matrix is modelled by

$$Q = \begin{bmatrix} - & \pi_C r_{tv} & \pi_G r_{ti} & \pi_T r_{tv} \\ \pi_A r_{tv} & - & \pi_G r_{tv} & \pi_T r_{ti} \\ \pi_A r_{ti} & \pi_C r_{tv} & - & \pi_T r_{tv} \\ \pi_A r_{tv} & \pi_C r_{ti} & \pi_G r_{tv} & - \end{bmatrix},$$

where r_{tv} is the transversion[1] rate, r_{ti} is the transition[2] rate, and $(\pi_A, \pi_C, \pi_G, \pi_T)'$ is the stationary distribution of CTMC. The notation $-$ is short for minus the sum of all the other entries in the same row. For example, in the first row of Q, it represents $-(\pi_C r_{tv} + \pi_G \pi_{ti} + \pi_T r_{tv})$. The transition probabilities, $P(T)$, of the HKY85 model is calculated using Equation (3.1); The explicit expression can be found in [70].

The HKY85 model is often reparameterized by the transition to transversion ratio, i.e. $\kappa = r_{ti}/r_{tv}$, representing a relative bias toward the occurrence of transitional events over transversional events. More precisely,

$$Q = \alpha \begin{bmatrix} - & \pi_C & \kappa\pi_G & \pi_T \\ \pi_A & - & \pi_G & \kappa\pi_T \\ \kappa\pi_A & \pi_C & - & \pi_T \\ \pi_A & \kappa\pi_C & \pi_G & - \end{bmatrix},$$

where the parameter α represents an overall transversion rate.

A special case of the HKY85 model is the Kimura two-parameter model (K2P) model [60], in which the stationary state frequencies are fixed to be equal, i.e. $\pi_A = \pi_C = \pi_G = \pi_T = 0.25$. Using the K2P model for phylogenetic reconstruction, the parameter α and branch lengths are unidentifiable. Without loss of generality,

[1] Transition refers to a mutation between two pyrimidines (T↔C) or two purines (A↔G).

[2] Transversion refers to a mutation between a pyrimidine and a purine (A↔C, A↔T, G↔C, or G↔T).

we assume $\alpha = 1$. In this dissertation, we use the K2P model with parameter $\kappa = 2$ as the default model.

Let θ denote the vector of parameters of the evolutionary model; $\theta \in \Theta$. We call θ *evolutionary parameters*. In the case of the K2P model, $\theta \equiv \kappa$. Recall that we use $\mathcal{Y}$ to denote the observed biological sequences. From now on, we use t to denote a phylogenetic tree in a tree space $\mathcal{X}$.

The likelihood, $L(\theta, t|\mathcal{Y})$, of the evolutionary parameter θ and the phylogenetic tree t is calculated by the sum-product algorithm, which can be easily expressed as a recursion. Let $L_v(j)$ denote the probability of everything that is observed below the internal node v conditional on v having character j. We start the recursion from calculating the likelihoods of leaves. If the leaf v has an observation i, let $L_v(i) = 1$, and $L_v(j) = 0$ for all $j \in \Sigma, j \neq i$. Then the sum-product algorithm computes $L_v(i)$ for each internal node v on the tree from the conditional likelihood function of its immediate descendant nodes, $C(v)$. Mathematically, this recursion can be expressed as follows:

$$L_v(i) = \begin{cases} 1, & \text{if } v \in X \text{ and } \mathcal{Y}(v) = i \\ 0, & \text{if } v \in X \text{ and } \mathcal{Y}(v) \neq i \\ \prod_{v \in C(v)} \left\{ \sum_j P_{ij}(b_v) L_v(j) \right\}, & \text{if } v \text{ is an internal node} \end{cases} \tag{3.2}$$

where $P_{ij}(b_v)$ is the transition probability of changing from i to j along the branch of length b_v, calculated by Equation (3.1). The sum-product algorithm continues the recursion until it computes $L_0(i)$, the probability of the tree with the root having character i. Finally, the likelihood is computed by

$$L(\theta, t|\mathcal{Y}) = \sum_{i \in \Sigma} \pi_i L_0(i). \tag{3.3}$$

String-valued Continuous Time Markov Chain (SCTMC)

The most popular SCTMC is the TKF91 model proposed in the pioneering paper of [106]. TKF91 is a time-reversible Markov model for single nucleotide insertions or deletions (indels). TKF91 does proper statistical analysis for two sequences by obtaining pairwise maximum likelihood of sequence alignments and estimating

the evolutionary distance between two sequences. Its main advantage is that its marginals $P(Y_T = s|Y_0)$ can be expressed as a string transducer with a set of weights obtained from closed-form functions of T such that it is as fast as score-based approaches. TKF92 [107] was developed based on TKF91 to deal with arbitrary-length non-overlapping indels by breaking one biological sequence into indivisible fragments. The math for TKF92 is similar to the TKF91 model, but with "residues" replaced by "fragments". Furthermore, [79] presented a probabilistic "long indel" model of sequence evolution, allowing for overlapping indels of arbitrary length. For more backgrounds and references, see http://biowiki.org/.

Researchers have used SCTMCs to compute the marginals of the process in order to obtain the probability of trees (and alignments) [54, 73, 106]. However, the computational bottleneck has prevented a broader applications based on SCTMCs. We will mainly focus on the CTMC over characters in this dissertation, and explore a computational method for a general SCTMC in Section 6.

3.4.2 Maximum Likelihood

In maximum likelihood inference, trees and evolutionary parameters are estimated by maximizing the likelihood function in Equation (3.3) that fully captures what the data tell us about the phylogeny under a given model [46]. Finding the maximum of the likelihood function involves searching through a multidimensional parameter space. Whether maximum likelihood is computationally expensive depends on the thoroughness of the search.

Although maximum likelihood [33] can be efficient for obtaining point estimates of phylogenies, it is difficult to determine statistical confidence [40]. The most commonly used procedure for assessing phylogenetic uncertainty is using nonparametric bootstrap [34]. Many studies have focused on comparing nonparametric bootstrap and posterior probabilities with regard to estimating the tree uncertainty [1, 20, 29, 96, 102]. These studies show that bootstrap tends to be too conservative.

3.4.3 Framework of Bayesian Phylogenetics

Bayesian phylogenetics has a strong connection to the maximum likelihood method. The same evolutionary models used in maximum likelihood can be used in Bayesian methods simply by specifying the prior distributions for the tree and evolutionary parameters. Priors incorporate previous knowledge from sources other than the data at hand. Bayesian methods estimate the posterior distribution which is a function of both the likelihood and the prior.

In this dissertation, we will focus on Bayesian phylogenetics methods because of several advantages. First, Bayesian methods allow complex models of sequence evolution to be implemented. Second, Bayesian phylogenetics produces both a tree estimate and measures of uncertainty for the groups on the tree. Third, Bayesian methods might be a faster way to assess support for trees than maximum likelihood bootstrapping [25].

Recall that $\mathcal{Y}$ is a set of observations on the leaves of a phylogenetic X-tree. For $X' \subset X$, we use the notation $\mathcal{Y}(X')$ for the subset of observations corresponding to a subset X' of the leaves.

Our objective is to use n observed biological sequences, $\mathcal{Y}$, to estimate a phylogenetic tree using Bayesian approaches. Let $\theta \in \Theta$ denote the vector of parameters of the evolutionary model. Let $p(\theta)$ denote the prior for θ. For a tree $t \in \mathcal{X}$, the density of the prior distribution given θ is denoted by $p(t|\theta)$. The probability model for observed data $\mathcal{Y}$ conditioning on a specific choice of parameters θ and tree t is $\mathbb{P}(\mathcal{Y}|\theta, t)$. Bayesian inference is based on the posterior density

$$p(\theta, t|\mathcal{Y}) = \frac{\mathbb{P}(\mathcal{Y}|\theta, t)p(t|\theta)p(\theta)}{\int_\theta \int_t \mathbb{P}(\mathcal{Y}|\theta, t)p(t|\theta)p(\theta)dtd\theta} \tag{3.4}$$

where $\mathbb{P}(\mathcal{Y}|\theta, t)$ is the likelihood function $L(\theta, t|\mathcal{Y})$ in Equation (3.3). All statistical inferences are based on the sample trees and model parameter values form their joint posterior distribution.

The total number of distinct labelled topologies of a rooted tree of n leaves is $(2n-3)!!$ [95], which increases at a super-exponential rate as the number of taxa increases. For example, when $n = 10$, the possible tree topologies are greater than 34 millions. Therefore, the denominator of the posterior density in Equation (3.4)

involves high-dimensional summations and integrals for which there is no closed-form solution except for very small trees.

Since typically the likelihood function for phylogenetic models are too complex to integrate analytically, we use Monte Carlo methods to approximate the posterior distribution of phylogenetic trees and evolutionary parameters. Conventional Bayesian phylogenetic inference uses MCMC to approximate the posterior probabilities of trees, which will be the focus of next subsection.

3.4.4 Bayesian Phylogenetics via MCMC

MCMC is conventionally used for approximating posterior distribution of phylogenetic trees [50–52, 66, 67, 70, 77, 82, 113]. Many user-friendly software packages have been developed for implementing MCMC for phylogenetics. The most popular packages include BAMBE [97], MrBayes [55, 90, 91], and BEAST [28].

The space under consideration is a joint space of all the possible trees and all the evolutionary parameters, denoted $\Omega = \Theta \times \mathcal{X}$. An MCMC algorithm samples a dependent sequence of points from the space Ω such that after some point in the sequence, all subsequent sampled points are distributed approximately according to the posterior distribution. Each iteration of the MH algorithm, described in Algorithm 2.1, proposes a new point of specifications of a phylogenetic tree and evolutionary parameters, usually similar to the present one, and we use the MH ratio to accept or reject it.

The main difficulty in Bayesian phylogenetic inference with MCMC lies in the efficiency with which topology proposals sample tree space. Ideally, a well functioned proposal distribution can lead to a well-mixing Markov chain that can rapidly traverse the posterior distribution such that inferences will be sufficiently accurate based on a computationally feasible sample. However, MCMC imposes relatively strict constraints on the types of proposals that can be used, making it is challenging to design fast mixing proposal distributions. Due to combinatorial constraints, the distribution on tree space is often a complex multimodal distribution, pronounced moves are likely to result in sample points with low posterior probabilities because the phylogenetic tree space is so huge that the posterior density is usually low for most places in the tree space. Design of the currently used

proposal distributions leads to a low acceptance rate of transitions unless a small neighborhood is used [51, 66]. Therefore, the proposed point is usually close to the present point of the chain, requiring the MCMC chain to run for a long time to explore the full possibilities of 'tree space'.

With a few exceptions [50, 51, 66], the proposals used by current phylogenetic MCMC sampler [67] have remained unchanged in the past decade. [67] discussed two types of MCMC algorithms, the GLOBAL algorithm and LOCAL algorithm, programmed in their software package BAMBE. The version of the GLOBAL algorithm modifies all branch lengths and potentially changes the tree topology simultaneously. In contrast, the LOCAL algorithm proposes changes to only small portions of the tree. Each algorithm has two versions, one assumes a molecular clock and one assumes a non-clock tree. A composition of two different basic update mechanisms are used to traverse the space of phylogenetic trees and evolutionary parameters. One cycle of MCMC iteration consists of two stages: in the first stage, keep the current tree fixed, propose the new evolutionary parameters, and either accepted it or rejected it according to the MH ratio; the second stage is to modifies the current tree while holding the evolutionary parameters fixed.

[66] evaluated two classes of proposal distributions to change topology and branch lengths simultaneously in phylogenetic inference for unrooted trees. More precisely, branch-change proposals modify branch lengths or branch attachment points continuously in a way that produces topology changes in some cases; branch-rearrangement proposals use the pruning-regrafting moves to prune a subtree from the rest of the tree and regraft it somewhere else or use swapping moves to simply trade places of two subtrees. [51] studied different proposal distributions for constrained rooted clock trees. [50] explored new MCMC proposal distributions to find more efficient MCMC algorithms.

In this dissertation, we will use the nearest neighbor interchange (NNI) [58] to propose a new tree topology, illustrated in Figure 3.3. The edge e has four nearest neighbors, A, B, C, and D, each of which can be a subtree. By interchanging the four nearest neighbors, the NNI proposal changes the first topology to one of the second and third topologies randomly.

One proposal for branch lengths, *multiplicative branch proposal*, picks one edge at random and multiply its current value by a random number distributed

uniformly in $[1/a, a]$ for some fixed parameter $a > 1$ (controlling how bold the move is) [66].

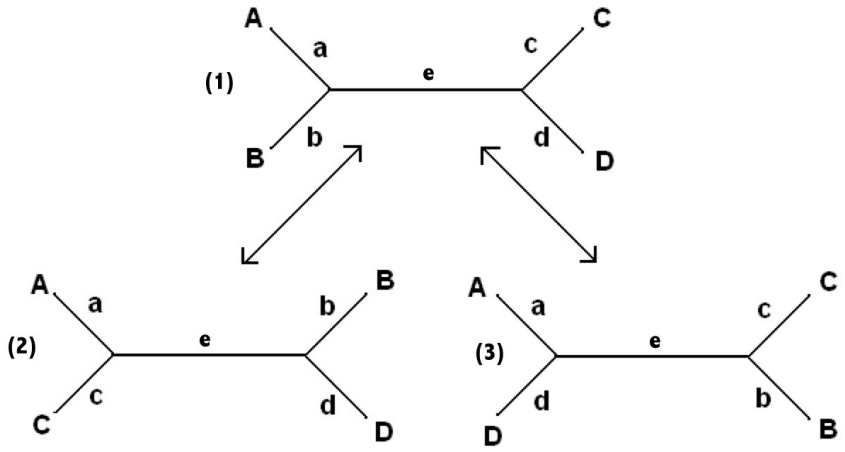

Figure 3.3: The Nearest Neighbor Interchange (NNI) Proposal [58].

In addition to the difficulty in designing proposal distributions, assessing whether the chain has converged is a crucial issue for using MCMC. Moreover, existing MCMC proposals are computationally expensive because they require recalculating the likelihood of the full tree from scratch, even when only a small perturbation is made on the current tree. In order to speed up computation, next chapter of this dissertation is focused on our work of exploring SMC algorithms for reconstructing general phylogenetic trees.

3.5 Consensus Tree and Tree Distances

In this section, we address the problem of summarizing a sample of phylogenetic trees, e.g. from MCMC algorithms. Moreover, we introduce the tree distance between two trees for the purpose of evaluating the estimated tree.

Contrary to summarizing a sample of real numbers, it is not a trivial problem for trees that are composed of tree topology and branch lengths. The summary tree usually refers to the maximum *a posteriori* (MAP) tree or the *consensus tree* [67]. Consensus trees are trees that summarize the information contained in a set

of trees having the same species on their leaves. In this dissertation, we use the *majority-rule consensus tree* which consists of those groups that are present in no less than a half of the trees [33].

We measure the distance between a reference tree t and an estimated consensus tree t' using three types of tree distance metrics: Robinson-Foulds (RF) [89], the partition metric (PM) [35], and Kuhner Felsenstein (KF) [65]. We first discard the edge directions from rooted trees to get unrooted trees. Each branch on an unrooted tree can partition the whole set of leaves into two unordered subsets, called one bipartition. We use $S(t)$ to denote the set of all the bipartitions of t: $S(t) = \{B_i, i = 1, \cdots, n_e\}$, where B_i is the bipartition resulting from the i-th edge. The set of different bipartitions of t and t' is denoted by $D(t,t') = S(t) \triangle S(t')$, where $A_1 \triangle A_2$ denotes the symmetric difference of sets A_1 and A_2. The partition metric of t and t' is defined as the number of their different bipartitions, denoted $|D(t,t')|$. The RF metric of t and t' is defined as $\sum_{B \in D(t,t')} |b(B;t) - b(B;t')|$, where $b(B;t)$ denotes the length of the branch corresponding to the bipartition B on tree t. The KF metric is defined as $\sum_{B \in D(t,t')} (b(B;t) - b(B;t'))^2$.

To make these distance metric more comparable for different trees, we normalize the partition metric of t and t':

$$\text{Normalized partition} = \frac{|D(t,t')|}{(|S(t)| + |S(t')|)}, \tag{3.5}$$

and we normalize the RF metric of t and t':

$$\text{Normalized RF} = \frac{\sum_{B \in D(t,t')} |b(B;t) - b(B;t')|}{(\sum_i^{n_e} b(B_i;t) + \sum_j^{n'_e} b(B_j;t'))}. \tag{3.6}$$

Chapter 4

Combinatorial SMC

sequential Monte Carlo (SMC) is another approach to approximation of posterior distributions, which has been very successful in state-space models ([15, 26, 27, 71]), and more recently, to more general settings [22, 23]. However, because of the intricacies of phylogenetic tree spaces, it is non-trivial to directly apply these general frameworks to posterior tree inference.

Previous work on applying SMC to phylogenetics has been limited in two important ways. First, it has relied on more restrictive SMC algorithms [9, 43, 105], and as a consequence, they are limited in the types of phylogenetic proposals they can use. Second, no previous results have been published on applying SMC to non-clock trees. This is an important limitation, as most current work in phylogenetics depends on non-clock tree models.

Our contribution is to show how both of these limitations can be addressed using a new framework for building SMC phylogenetic tree inference algorithms. In our combinatorial sequential Monte Carlo (CSMC) framework, the flexibility on the proposal distributions generalizes both MCMC methods and previous work on phylogenetic SMC. In particular, this flexibility makes it possible to construct non-clock tree proposals that do not require recalculating the likelihood of the full tree at each proposal step.

The proposed SMC algorithm is motivated by a certain over-counting problem in sequentially constructing a non-clock phylogenetic tree. A conventional SMC algorithm will favour the trees which can be constructed in multiple ways, whereas

in our algorithm, a graded poset on an extended combinatorial space is used to compute correction terms.

The remainder of this chapter is organized as follows. Section 4.1 introduces some notations used in this chapter. Section 4.2 describes the CSMC algorithm, its asymptotic consistency properties and its application to phylogenetics. In Section 4.3, we show some results of simulation studies and apply the proposed SMC methods to real RNA datasets. Section 4.4 concludes this chapter and provides some discussions.

4.1 Notation

Recall that we use π to denote the normalized measure of interest, which can be expressed as $\pi = \gamma/Z$ using the unnormalized measure γ and the normalizing constant Z. In this chapter, we introduce another notation, $||\gamma||$, to denote the normalizing constant, i.e. $Z \equiv ||\gamma||$, for the purpose of describing the SMC algorithm in a compact form.

In Bayesian phylogenetics, π is the normalized posterior over trees, i.e. $\pi = p(\theta, t|\mathcal{Y})$; the unnormalized posterior γ is a measure over trees t with density

$$p(\theta, t, \mathcal{Y}) = \mathbb{P}(\mathcal{Y}|\theta, t)p(t|\theta)p(\theta);$$

the normalizing constant is the data likelihood in the denominator of (3.4), i.e. $||\gamma|| = \mathbb{P}(\mathcal{Y})$. In this chapter, we assume that the evolutionary parameters θ is fixed and we focus on the phylogenetic tree reconstruction. We can omit θ from the notation for simplicity.

We are interested in approximate expectations of a function ϕ with respect to π. Denote $\pi\phi$ as a short-hand for integration of ϕ with respect to π, i.e. $\pi\phi = \int \pi(dt)\phi(t)$. As a concrete example of the posterior expectation of a function ϕ in phylogenetics, we define ϕ as an indicator of whether a clade, a group consisting of a species and all its descendants, is in a phylogenetic tree t or not. Let clades(t) denote the set of all clades in t. For a clade a, define $\phi(t) = \mathbf{1}(a \in \text{clades}(t))$, where $\mathbf{1}$ is an indicator function.

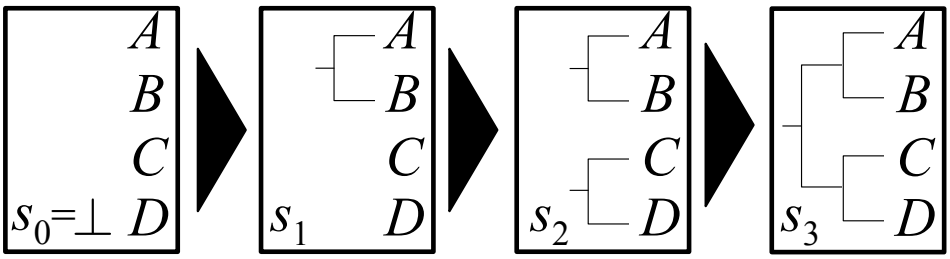

Figure 4.1: An example of the partial states of a discrete rooted X-tree.

4.2 Method

In this section, we describe CSMC, an algorithm to approximate expectations $\pi\phi$ and find the normalization of an unnormalized positive measure γ on a target combinatorial space $\mathcal{X}$. To simplify the presentation, we first introduce the algorithm under the assumption that $\mathcal{X}$ is a finite but large combinatorial set, e.g. a set of tree topologies. We will show in Section 4.2.5 that this assumption can be relaxed, e.g. to accommodate branch lengths in phylogenetics.

4.2.1 Discrete Setup

At a high level, the main assumption on which our algorithm depends is that any object t in the target sample space $\mathcal{X}$ can be constructed incrementally using a sequence of intermediate states $s_0, s_1, \ldots, s_R = t$. Here s_r will be called a *partial state of rank r*, the set of all partial states of rank r will be denoted by $\mathcal{S}_r$, and the set of partial states across all ranks will be denoted by $\mathcal{S} = \bigcup_r \mathcal{S}_r$. Our terminology (partial states, rank) has an order theoretic motivation that will be described in more details shortly. Note that we consider phylogenies without branch lengths in this subsection.

In phylogenetics for example, the partial states we will use are based on *forests*. Consider for example a discrete rooted X-tree as in Figure 4.1. In this section, we use the term *tree* as a short-hand for *tree topology*. Such a tree can be constructed by starting from the disconnected graph with vertices X, and by adding, at each step, one internal node $v \notin X$, and a pair of edges connected to v such that

the number of connected components decreases by one at every step. More precisely, we will build each rooted X-tree t by proposing a sequence of X-forests $s_0, s_1, \ldots, s_R = t$, where an X-forest $s_r = \{(t_i, X_i)\}$ is a collection of rooted X_i-trees t_i such that the disjoint union of leaves of the trees in the forest is equal to the original set of leaves, $\bigcup_i X_i = X$. Note that with this specific construction, a forest of rank r has $|X| - r$ trees.

The sets of partial states considered in this section are assumed to satisfy the following three conditions:

1. The sets of partial states of different ranks should be disjoint, i.e. $\mathcal{S}_r \cap \mathcal{S}_s$ for all $r \neq s$ (in phylogenetics, this holds since a forest with r trees cannot be a forest with s trees when $r \neq s$).

2. The set of partial state of smallest rank should have a single element denoted by $\perp$, $\mathcal{S}_0 = \{\perp\}$ (in phylogenetics, $\perp$ is the disconnected graph on X).

3. The set of partial states of rank R should coincide with the target space, i.e. $\mathcal{S}_R = \mathcal{X}$ (in phylogenetics, at rank $R = |X| - 1$, forests have a single tree and are members of the target space $\mathcal{X}$).

These conditions will be subsumed by the more general framework of Section 4.2.5, but the more concrete conditions above help understanding the poset framework.

In order to put the CSMC algorithm under service, the user first needs to specify an *extension* of the measure γ (the corresponding normalized measure is denoted by π), defined only on the target space $\mathcal{X}$, into a measure over the larger space $\mathcal{S}$. The restriction of this extended measure on $\mathcal{X}$ should coincide with the target measure γ. We abuse notation and use γ and π for both the extended and the target measures.

In phylogenetics for example, a measure over X-trees is usually specified by the user: in the Bayesian setup of Section 3.4.3 it is given by a prior times a likelihood:[1]

$$\gamma(t) = \gamma_{\mathcal{Y}}(t) = \mathbb{P}(\mathcal{Y}|t)p(t),$$

[1] Since we assume a discrete space in this section, we write $\gamma(t) = \gamma(\{t\})$ and denote the probability mass function of the prior by $p(t)$.

where we are assuming fixed parameters θ, an assumption we will relax in Chapter 5. A natural choice in non-clock models to obtain an extension of γ into forests is to take a product over the trees in the forest s as follows:

$$\gamma(s) = \prod_{(t_i, X_i) \in s} \gamma_{\mathcal{Y}(X_i)}(t_i).$$

We call this choice of extension the *natural forest extension*, and note that a range of other choices are possible (see Section 4.4 for more examples).

4.2.2 Algorithm

Algorithm 4.1 The Combinatorial Sequential Monte Carlo Algorithm

1. Step 1: initialization
   (a) for all $k \in \{1, \ldots, K\}$,
      i. set $s_{0,k}$ to $\perp$
      ii. set $w_{0,k}$ to $1/K$
   (b) construct $\gamma_{0,K}$ using Equation (4.1)
2. Step 2: for iteration $r = 1, 2, \ldots, R$,
   (a) for all $k \in \{1, \ldots, K\}$:
      i. sample $\tilde{s}_{r-1,k} \sim \pi_{r-1,K}$
      ii. sample $s_{r,k} \sim \nu^{+}_{\tilde{s}_{r-1,k}}$
      iii. compute $w_{r,k} = w(\tilde{s}_{r-1,k}, s_{r,k})$ using Equation (4.2)
   (b) construct $\gamma_{r,K}$ using Equation (4.1)
3. Step 3: return $\gamma_{R,K}$

In this section, we introduce the CSMC algorithm, an iterative procedure that approximates the target measure π in R iterations. At each iteration r, a list of K partial states is kept in memory. Each element of this list is called a *particle*, denoted $s_{r,1}, s_{r,2}, \ldots, s_{r,K} \in \mathcal{S}_r$. We also assume there is a positive weight $w_{r,k}$ associated with each particle $s_{r,k}$. Refer to Algorithm 4.1 for an overview of the steps

described in more details in the following.

The general form of the algorithm has similarities with standard SMC algorithms, with the important exception of the weight updates, which needs to be altered to accommodate general combinatorial structures.

Given a list of weighted particles, we construct a discrete positive measure as follows:

$$\gamma_{r,K}(A) = \|\gamma_{r-1,K}\| \frac{1}{K} \sum_{k=1}^{K} w_{r,k} \delta_{s_{r,k}}(A), \quad \text{for all } A \subset \mathcal{S}. \tag{4.1}$$

The algorithm constructs these discrete measures recursively, and returns the measure $\gamma_{R,K}$ at termination. Note that in most descriptions of standard SMC algorithms, the algorithm is described in terms of a target *probability distributions* π. Allowing γ and the intermediate particle populations to be unnormalized has the advantage of revealing more explicitly the connexion between the estimators for the evidence $\mathbb{P}(\mathcal{Y})$ and posterior distribution $p(t|\mathcal{Y})$.

The algorithm is initialized with the rank $r = 0$ populated with K copies of the least partial state $\perp$ described in the previous section. Each of these copies is associated with a uniform initial weight, $1/K$.

Given the empirical measure $\gamma_{r-1,K}$ from the previous population of particles, a new list of particles and weights is created in three steps. Figure 4.2 provides a graphical illustration of the three steps of an SMC algorithm.

The first step is to select a promising list of particles while preserving the correct asymptotic distributions. In this section we use the simplest scheme, multinomial resampling, to accomplish this, but other schemes are possible and are discussed in Section 2.2.2. In multinomial resampling, this first step is simply done by resampling K times from the normalized discrete measure $\pi_{r-1,K}$. We denote these sampled particles by $\tilde{s}_{r-1,1}, \tilde{s}_{r-1,2}, \ldots, \tilde{s}_{r-1,K}$.

The second step is, for each particle $\tilde{s}_{r-1,k}$ in this list, to use the proposal $\nu^+_{\tilde{s}_{r-1,k}}$ to grow each of the resampled particle into a new particle of rank r, denoted by $s_{r,k}$. In order to do this, the user needs to specify a *proposal* probability kernel $\nu^+_s : \mathcal{S} \to [0,1]$ for all $s \in \mathcal{S}$. Given an initial partial state s and destination partial state s', we denote the probability of proposing s' from s by $\nu^+_s(s')$ or by a more readable

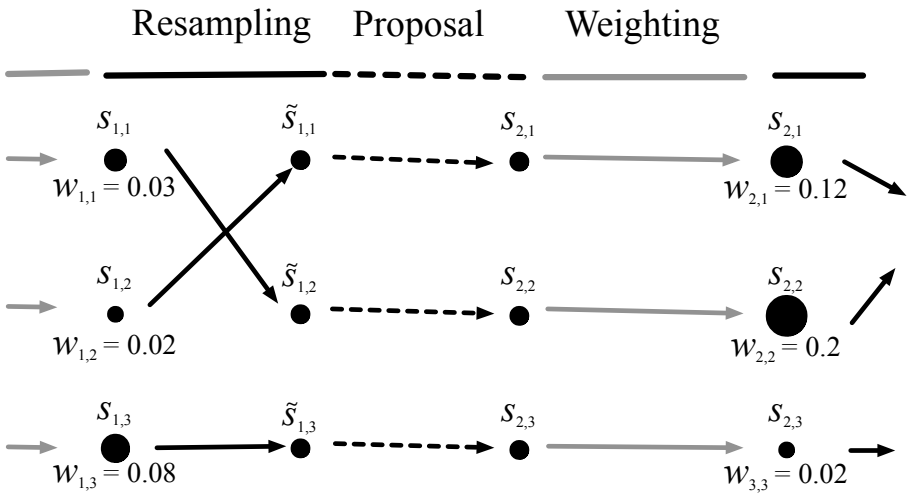

Figure 4.2: A graphical illustration of the SMC algorithm.

notation $\nu^+(s \to s')$ especially when subscripts are involved. We assume that the successors proposed from a partial state of rank r will always have rank $r+1$, i.e. if $s \in \mathcal{S}_r$ and $\nu_s^+(s') > 0$, then $s' \in \mathcal{S}_{r+1}$. This proposal is applied independently to sample a successor to each particle $\tilde{s}_{r-1,k}$: $s_{r,k} \sim \nu^+_{\tilde{s}_{r-1,k}}$.

For example, when building discrete rooted X-trees, the proposal needs to select a pair of trees to merge. One simple choice is to pick a pair uniformly at random among the $\binom{|X|-r}{2}$ pairs; other choices are discussed in Section 4.4.

The third step is to compute a weight for each of these new particles. This is done using the following formula:

$$\begin{aligned} w_{r,k} &= w(\tilde{s}_{r-1,k}, s_{r,k}) \\ &= \frac{\gamma(s_{r,k})}{\gamma(\tilde{s}_{r-1,k})} \cdot \frac{\nu^-(s_{r,k} \to \tilde{s}_{r-1,k})}{\nu^+(\tilde{s}_{r-1,k} \to s_{r,k})}, \end{aligned} \tag{4.2}$$

where we call the function ν^- an *overcounting function.* Note that while this weight update looks like a Metropolis-Hastings ratio, it has the fundamental difference that $\nu^+ \neq \nu^-$ in general. The overcounting correction is more closely related to the backward kernels of [22, 23], but because of the combinatorial nature of the space,

the poset framework plays an instrumental role in constructing ν^- in the types of spaces we are interested in.

After running this algorithm, the discrete positive measure $\gamma_{R,K}$ can be used to approximate γ, where $\gamma_{R,K}$ is obtained from applying Equation (4.1) to the particles output at the last generation. Concretely, this means that $\|\gamma_{R,K}\|$ can be used to estimate $\mathbb{P}(\mathcal{Y})$, and for any statistic of interest ϕ, its expectation under the posterior can be estimated by $\pi_{R,K}\phi$.

In the next section, we show how ν^- can be selected to guarantee convergence of the algorithm to the target distribution π when $K \to \infty$. The precise meaning of convergence will be discussed in Section 4.2.4 and 4.2.5.

4.2.3 Overcounting Correction

In the previous section, we have described an algorithm similar to standard SMC, with the exception of the overcounting correction, which has an extra factor ν^- in the weight update. Before explaining how to compute this correction in general, we first describe the problem that would arise if we were to omit the correction in the specific case of phylogenetic non-clock inference. To simplify the discussion, we start by considering a model where there is no observation (in practice, sampling from the prior would not require an approximate algorithm, but it is instructive to start by studying this simpler example).

For this example, refer to Figure 4.3 (a), where we show that all the different sequences of partial states (forests) leading to one of the $1 \cdot 3 \cdot 5 = 15$ fully specified states (rooted X-trees). An arrow between partial states s and s' in this figure means that s' can be obtained from s by one application of the proposal, i.e. that $\nu^+(s \to s') > 0$.

It can be seen in the figure that a balanced binary tree on four taxa, for example one with rooted clades $\{A, B\}, \{C, D\}$, can be constructed in two distinct ways: either by first merging A and B, then C and D, or by first merging C and D, then A and B. On the other hand, an unbalanced tree on the same set of tax can be constructed in only one way, for example the tree with clades $\{A, B\}, \{A, B, C\}$ can only be constructed by first proposing to merge A and B, then C and $\{A, B\}$. A consequence of this dichotomy is that under the uniform proposal, the expected fraction of parti-

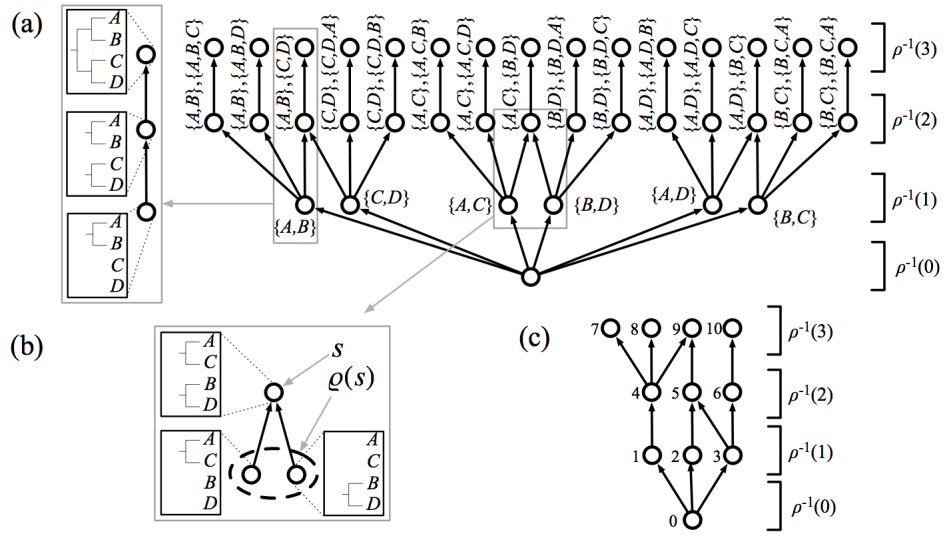

Figure 4.3: (a) All the different sequences of partial states (forests) leading to fully specified states (rooted X-trees). (b) An example of the set of parents $\varrho(s)$ of a partial state s. (c) An example of a simple cyclic poset.

cles with a balanced topology of each type is $2/21$, while it is $1/21$ for unbalanced topologies (since there are 21 proposal paths, 2 for each balanced topologies, 1 for each unbalanced one). Since we would like the probability π of each topology to be close to 1/5, the naive estimate is therefore inconsistent.

In order to resolve this issue (by defining an appropriate overcounting function), it will be useful to formalize the graph shown in Figure 4.3 (c). This can be done by defining a *partial order* $\leq$ on $\mathcal{S}$. Recall that $(\mathcal{S}, \leq)$ is called a *partially ordered set*, or briefly a *poset*, if $\leq$ is a binary relation on $\mathcal{S}$ that is *reflexive, antisymmetric*, and *transitive* [99]. Also, for $s, s' \in \mathcal{S}$, we say that s *is covered by* s', written $s \prec s'$, if $s \leq s'$ and there is no $z \in \mathcal{S}$ between s and s'. Since the covering relation determines the partial order in a finite ordered set, this means that in our combinatorial setup we can induce a poset on the extended space $\mathcal{S}$ by deeming that s' covers s if and only if $\nu^+(s \to s') > 0$.

By the way we defined $\mathcal{S} = \bigcup_r \mathcal{S}_r$, we get that $\leq$ actually has an extra structure called a *rank* ρ: a function from $\mathcal{S}$ to $\{0, 1, \cdots, R\}$ such that $\rho(s_0) = 0$ if s_0 is a

minimal element of the poset, and $\rho(s') = \rho(s) + 1$ if s' covers s in $\mathcal{S}$.

With these definitions, graphs such as Figure 4.4 (left) can then be understood as the *Hasse diagram* corresponding to this induced poset, namely a graph where the set of vertices is $\mathcal{S}$, and there is an edge between s and s' whenever s covers s'.

In previous work [9], the overcounting problem has been avoided by forbidding proposals ν^+ that induce a cyclic Hasse diagram. In the CSMC algorithm, ν^- is used to avoid this artificial restriction. We present the solution in discrete spaces $\mathcal{S}$ in this section. The discrete assumption is lifted in Section 4.2.5.

Generally, in order to have consistency as in Equation (4.6), the only requirement on ν^- is:

Assumption 3. *For all* $s, s' \in \mathcal{S}$, $\nu^+(s \to s') = 0$ *implies* $\nu^-(s' \to s) = 0$.

Let $\varrho : \mathcal{S} \to \mathcal{F}_{\mathcal{S}}$ denote the set of parents $\varrho(s)$ of a partial state s. In phylogenetics for example, the number of parents is equal to the number of nontrivial trees in the forest, where a tree is said to be trivial if it has a single leaf in it (see Figure 4.3 (b) for an example).

A simple choice of ν^- that satisfies this condition is

$$\nu^-(s' \to s) = |\varrho(s')|^{-1} \times \mathbf{1}[\nu_s^+(s') > 0], \tag{4.3}$$

when $|\varrho(s')|$ is finite. Correspondingly, we get the following formula for the case of rooted non-clock tree inference:

$$\nu^-(s' \to s) = \left(\sum_{(t_i, X_i) \in s'} \mathbf{1}[|X_i| > 1] \right)^{-1}. \tag{4.4}$$

While Assumption 3 is very weak, it is of interest to select ν^- so as to minimize the variance of the weights appearing in the CSMC algorithm. By generalizing Proposition 3.1 in [22], one can show that the optimal choice for ν^- is

$$\nu^{-opt}(s' \to s) = \frac{\pi(s)}{\sum_{s'' \prec s'} \pi(s'')}. \tag{4.5}$$

4.2.4 Analysis in the Discrete Case

To motivate the overcounting correction, we sketch in this section an elementary proof that CSMC converges to the correct value as the number of particles goes to infinity. Again, we restrict our attention to the discrete case for now ($|\mathcal{S}| < \infty$), where the proof is transparent. In Section 4.2.5, we present a general result that do not make this assumption.

Formally, we will prove the following result, where for simplicity, in this chapter $\rightarrow$ denotes convergence as the number of particles $K \rightarrow \infty$ in L^2 unless stated otherwise.

Proposition 4. *Under Assumption 3, we have for all test functions $\phi : \mathcal{X} \rightarrow \mathbb{R}$*

$$\gamma_{R,K}\phi \rightarrow \gamma\phi. \tag{4.6}$$

There are two important corollaries of consistency results of the form of Equation (4.6):

1. The normalized weighted particles obtained at iteration R of the SMC algorithm, $\pi_{R,K}$, form a consistent estimator for computing expectations under π:

   $$\pi_{r,K}\phi \rightarrow \pi\phi.$$

   This corollary is important because π corresponds to a posterior expectation in the Bayesian framework.

2. The normalization of γ can be consistently estimated as follows:

   $$\|\gamma_{r,K}\| \rightarrow \|\gamma\|. \tag{4.7}$$

   This corollary is important because $\|\gamma\|$ corresponds to a marginal likelihood in the Bayesian framework. Note that the left-hand side of Equation (4.7) is equal to the product of all the weight normalizations divided by K^R (which

is the way this result is usually presented), i.e.

$$Z_{r,K} \equiv \|\gamma_{r,K}\| = \prod_{j=1}^{r} \frac{1}{K} \sum_{k=1}^{K} w_{j,k}.$$

Proposition 4 can be established in two steps. First, we note that when the induced Hasse diagram is acyclic, previous SMC consistency proofs apply directly. Second, we show that in the cyclic case, we can construct a certain distribution $\check{\pi}$ and proposal $\check{q}^+$ on a larger space $\check{\mathcal{S}}$ with the following properties.

1. The target distribution π can be obtained from $\check{\pi}$ by straightforward marginalisation of the samples.

2. The induced Hasse diagram is acyclic, so the algorithm on $\check{\mathcal{S}}$ is consistent by the first step of the proof.

3. The proposal steps and weight updates in the algorithm on $\check{\mathcal{S}}$ can be shown to be equivalent to those of the original algorithm on $\mathcal{S}$. This shows that the algorithm on $\mathcal{S}$ is also consistent.

As described above, let us start by assuming the poset is acyclic. In this case, we claim that we can invoke Proposition 4 of [9]. First, the boundedness assumptions required in Proposition 4 are automatically satisfied since here $|\mathcal{S}| < \infty$. Second, the connectedness assumption made in this previous work, Assumption 2b, can be shown to hold using the following argument: assume on the contrary that there is a connected component $C \subset \mathcal{S}$ in the Hasse diagram that does not contain $\perp$, and let s be a minimal element, where $s \in C$ by finiteness. Since $\{s' : \nu^-(s \to s') > 0\} \subset \rho^{-1}(\rho(s) - 1)$, we have a contradiction. Therefore there can be only one connected component.

If the poset is not acyclic, we now present the reduction to the acyclic case. Let $\mathcal{S}_{0:r} = \mathcal{S}_0 \times \mathcal{S}_1 \times \cdots \times \mathcal{S}_r$, the set of paths of length r in $\mathcal{S}$. We will view the algorithm as incrementally building partial states over a larger space, with $\check{s}_0 \in \mathcal{S}_0, \check{s}_1 \in \mathcal{S}_{0:1}, \check{s}_2 \in \mathcal{S}_{0:2}, \ldots, \check{s}_R \in \mathcal{S}_{0:R}$. In other words, instead of viewing the algorithm as operating over $\mathcal{S} = \bigcup_{r=0}^{R} \mathcal{S}_r$, we will view it as operating over $\check{\mathcal{S}} = \bigcup_{r=0}^{R} \mathcal{S}_{0:r}$.

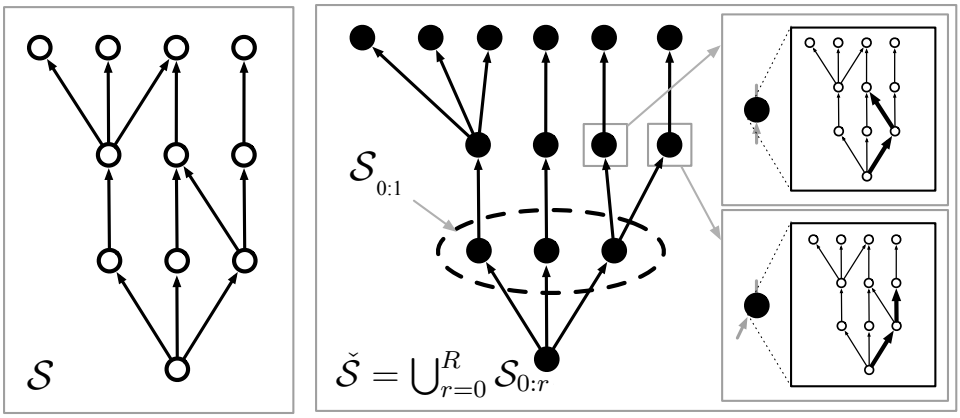

Figure 4.4: An example of changing a simple cyclic poset to an acyclic case.

Let us start by introducing a new measure $\check{\pi}$ on $\check{\mathcal{S}}$. Let $\check{s}$ be an element in $\check{\mathcal{S}}$, i.e. a sequence of forests, say of length r, $\check{s} = \check{s}_r = (s_0, s_1, \ldots, s_r) \in \mathcal{S}_{0:r}$. Following [22, 23], we define the new measure by a product $\check{\gamma}(\check{s}_r) = \gamma(s_r) \prod_{j=1}^{r-1} \nu^-(s_j \to s_{j-1})$. Note that since the $\nu^-(s \to \cdot)$ are assumed to be normalized probability densities, marginalization over $s_0, s_1, \ldots, s_{r-1}$ recovers the original extended measure π.

The proposal over $\check{\mathcal{S}}$ creates an identical copy of the sequence of forests, and adds to it a new elements by sampling from the original proposal density ν^+. Note that with this definition, given an element $\check{s} \in \check{\mathcal{S}}$, there can be only one predecessor $\varrho(\check{s})$, namely the prefix of the sequence with the last element removed. As a simple example, Figure 4.4 shows the picture of a simple finite cyclic poset and its acyclic poset over the extended space.

Finally, standard SMC operating on this extending space can be seen to be equivalent to CSMC, since the weight updates simplify to:

$$\begin{aligned}\frac{\check{\gamma}(\check{s}_r)}{\nu^+(\check{s}_{r-1} \to \check{s}_r)\check{\gamma}(\check{s}_{r-1})} &= \frac{\gamma(s_r) \prod_{j=1}^{r} \nu^-(s_j \to s_{j-1})}{\nu^+(\check{s}_{r-1} \to \check{s}_r)\gamma(s_{r-1}) \prod_{j=1}^{r} \nu^-(s_j \to s_{j-1})} \\ &= \frac{\gamma(s_r)\nu^-(s_r \to s_{r-1})}{\gamma(s_{r-1})\nu^+(s_{r-1} \to s_r)}\end{aligned}$$

This completes the proof in the discrete cyclic case.

4.2.5 General Setup

As in the previous section, γ denotes the target positive measure, but in this section we do not restrict γ to be defined over a discrete space. More precisely, let $\mathcal{F}_\mathcal{X}$ denote a sigma-algebra on $\mathcal{X}$, and let $\gamma : \mathcal{F}_\mathcal{X} \to [0, \infty)$. We assume that the user has provided an extension $\gamma : \mathcal{F}_\mathcal{S} \to [0, \infty)$, and a pair of forward and backward proposal probability kernels $\nu^+, \nu^- : \mathcal{S} \times \mathcal{F}_\mathcal{S} \to [0, 1]$.

Next, we define the following measures on the product space $\mathcal{S} \times \mathcal{S}$:

$$\tau^+(A \times B) = \gamma_A(\nu^+(B)) = \int \gamma_A(dx)\nu^+_x(B) \tag{4.8}$$

$$\tau^-(A \times B) = \gamma_B(\nu^-(A)) = \int \gamma_B(dx)\nu^-_x(A), \tag{4.9}$$

where $A, B \in \mathcal{F}_\mathcal{S}$, and for any measure λ and measurable A, $\lambda_A(B)$ denotes $\lambda(A \cap B)$. Informally, and up to normalization, the measure in Equation (4.8) corresponds to picking A according to γ, and then propagating the points in A using the forward proposal. The measure in Equation (4.9) corresponds to the inverse situation of picking B according to γ and then propagating these points using the backward proposal (this second situation is hypothetical since the algorithm does not sample from ν^-, but is useful in the analysis).

We make the following assumptions:

Assumption 5. *We have $\tau^- \ll \tau^+$,*

which implies by the Radon-Nikodym theorem the existence of a derivative $\tau^-/\tau^+ : \mathcal{S} \times \mathcal{S} \to [0, \infty)$. We also assume that there is a version $w = \tau^-/\tau^+$ with the following properties:

Assumption 6. *There is a ranked poset $(\mathcal{S}, \prec, \rho)$ such that s' covers s iff $w(s, s') > 0$, $\rho : \mathcal{S} \to \mathbb{Z}$. Using the notation γ_r as a short hand for $\gamma_{\rho^{-1}(r)}$, we assume that the Hasse diagram induced by $\prec$ is (a) connected and (b) that there is an R such that $\gamma_r = \gamma$ for $r \geq R$, and that γ_r is a Dirac delta for $r \leq 0$.*

To get a compact notation for CSMC, we introduce the following Monte Carlo

propagation operator:

$$(\text{prop}_K\,\lambda)\phi = \|\lambda\|\left(\frac{1}{K}\sum_{k=1}^{K} w(S_k, S'_k)\phi(S'_k)\right),$$

where $S_k \sim \bar{\lambda}$ ($\bar{\lambda}$ denotes the normalized measure of λ), $S'_k|S_k \sim \nu^+(S_k \to \cdot)$, independently, $\lambda : \mathcal{F}_\mathcal{X} \to [0,\infty)$ is an arbitrary measure on the poset, and $\phi : \mathcal{S} \to \mathbb{R}$ is a test function.

Note that the propagation operator incorporates both a multinomial resampling step and a proposal step in one equation. If we denote the composition of these operators by $\text{prop}_K^2\,\gamma_0 = (\text{prop}_K(\text{prop}_K\,\gamma_0))$, then the full CSMC algorithm can be summarized by $\gamma_{R,K} = \text{prop}_K^R\,\gamma_0$.

With this notation, we can write the main result as:

Proposition 7. *If $\phi : \mathcal{X} \to \mathbb{R}$ is measurable and such that $\phi \le C_1, w \le C_2$ for some C_1 and C_2, then*

$$\gamma_{R,K}\phi \to \gamma\phi.$$

The proof and supporting lemmas can be found in the appendix. There are many similarities between our proof and that for consistency in general state space models, however the main point of the appendix is to illustrate how the conditions on the poset are used in the proof.

In the next section, we show how this result can be specialized to various phylogenetics setups.

4.2.6 Connection to Phylogenetics with Branch Lengths

In an ultrametric setup, the poset $\mathcal{S}$ is defined as the set of *ultrametric forests* over X. An ultrametric forest $s = \{(t_i, X_i)\}$ is a set of ultrametric X_i-trees t_i such that the disjoint union of the leaves yields the set of observed taxa. Defining the *height* of an ultrametric forest as the height of the tallest tree in the forest. Heights of subtrees on a clock tree has been shown in Figure 3.2. The height increment, denoted Δ_r where $r = \rho(s)$, of the forest s represents the waiting time between two coalescent events. A common prior for ultrametric trees is the coalescent [61]. We assume

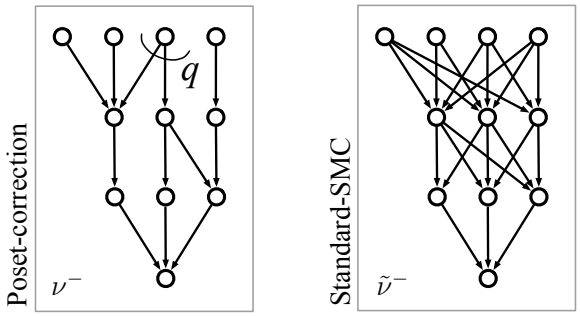

Figure 4.5: The support of the backward kernels used in the synthetic poset experiments.

the unconditional distribution of the topology is the uniform distribution over all possible topologies. The prior for Δ_r is $Exp\left(\mu_0\binom{|s|}{2}\right)$.

In a non-clock setup, the poset $\mathcal{S}$ is defined as the set of *non-clock forests* over X. A non-clock forest $s = \{(t_i, X_i)\}$ is a set of X_i-trees t_i in which the root is the newly formed internal node when s is a partial state and the root is an arbitrary point on the new branch when s is a full state. A common prior on a branch length of an unrooted non-clock phylogenetic tree is $\text{Exp}(\mu_0)$.

When building discrete rooted X-trees, three elements need to be proposed at each step: which pair of trees to merge, what is the length of the added branch, and what is the new position of the root. For efficiency, in a non-clock setup, we assume in most of what follows that the pair is picked uniformly at random among the $\binom{|X|-r}{2}$ pairs, that the branch length is sampled according to the prior, and that the position of the root is sampled uniformly on the newly inserted branch. An ultrametric setup can be regarded as a special case of a non-clock setup with some constraint: the unique position of the root is selected to ensure the newly formed tree is ultrametric. In practice, this constraint is satisfied by sampling the height increment rather than the branch length directly.

4.3 Numerical Examples

4.3.1 Synthetic Posets

We start with an illustration of the effect that a lack of appropriate correction can have on the approximation in cyclic posets. We use a synthetic poset with a small cardinality $|\mathcal{S}| < \infty$ so that the exact target distribution can be computed easily. The poset has a support as in Figure 4.4 (left) [2]. Here both the exact solution and the approximations are low-dimensional multinomial distributions, so the total variation distance can be computed efficiently.

First, we use this simple test case to show that Assumption 5 on ν^- is necessary to have consistency (Proposition 7). Two versions of the particle algorithm presented in the previous section are compared: the first one, 'Poset-correction' uses a backward kernel satisfying Assumption 5, while the second one, 'Standard-SMC', does not. The latter uses a backward proposal proportional to $\tilde{\nu}^- \equiv 1$, in which case the weight update reduces to the weight update found in state-space SMC algorithm. It can be checked easily that this choice violates Assumption 5. The support of the two backward kernels are shown in Figure 4.5.

In Figure 4.6 (left), we compare the performance of the two algorithms as the number of particles increases. The performance of the algorithms is measured using the total variation distance, and the experiment for each number of particle is repeated 1000 times using different random seeds. The results show that only the algorithm satisfying Assumption 5 gives an approximation with a total variation distance going to zero as the number of particles increases.

We did a second experiment to give an example where cycles in posets are beneficial. In this experiment, we fix the structure of the backward kernel as in the 'Poset-correction', with the exception of one of the backward transition, which we parameterize with a number $q \in [0, 1]$ shown in Figure 4.5 (left). When $q \in \{0, 1\}$, this effectively corresponds to removing a cycle in the Hasse diagram of the poset. We show in Figure 4.6 (right) the total variation distance as a function of this parameter q for a fixed number of particles (10). It can be seen that the

[2] The values of π, ν^+, ν^- can be downloaded from http://www.stat.ubc.ca/~l.wang/phylo/syntheticPoset/.

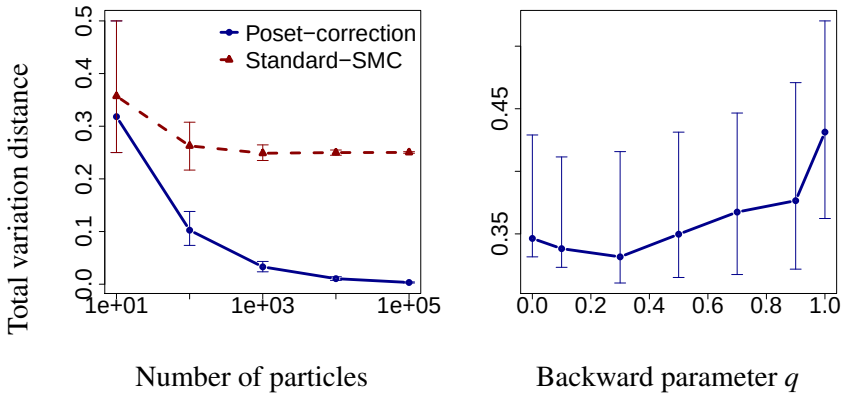

Figure 4.6: Total variation distance of the particle approximation to the exact target distribution. Left: the performance of the two algorithms as the number of particles increases. Right: the total variation distance as a function of the parameter q for a fixed number of particles (10).

best performance is attained away from the points $\{0, 1\}$, showing that the cycle is indeed beneficial.

4.3.2 Synthetic Phylogenies

In this study, we evaluate the proposed CSMC method on estimating phylogenetic trees. We summarize the set of the random trees obtained from the CSMC algorithm using the consensus tree [67]. We calculate the tree distance between estimated consensus trees and true trees using the RF metric and the PM. The smaller the tree distance, the better the performance. See Section 3.5 for an introduction to consensus trees and distance metrics for phylogenetic trees.

We simulated 100 ultrametric trees of 10 taxa with $\mu_0 = 10$. The non-clock trees were obtained by perturbing the branch lengths of ultrametric trees. More precisely, we modified a branch of length b by adding a noise randomly sampled from Unif$(-.3b, .3b)$. We generated 10 datasets for each tree. In each dataset, a total of 10 DNA sequences were generated given a tree and the K2P model with the parameter $\kappa = 2$. The length of each sequence is 1000. We generated the

sequence on the root by randomly sampling from the stationary distribution of the CTMC; then we generated each of its children using the transition probability that was computed with κ and the branch length by Equation 3.1. This procedure was recursively implemented until reaching leaves. The sequences of the internal nodes were discarded and those on leaves were taken as the observations.

CSMC was applied to each of these datasets. For comparison, we also used one widely-used Bayesian phylogeny software, MrBayes (MB), which implements MCMC for a range of tree models [55]. For both software, we fixed the parameter, κ, to be the true value and focused on comparing their performance in phylogenetic tree estimation. Priors for the branch lengths were exponential distributions with rate 10. The initial tree for MB was of an arbitrary topology and branch lengths sampled from exponential distribution with the true tree rate.

In both the CSMC and MCMC algorithms, the computational bottleneck is the peeling recurrence, which needs to be computed at each speciation event in order to evaluate $\pi(t)$. Therefore, we report running times as the number of times the peeling recurrence is calculated. Figure 4.7 shows the average Robinson Foulds metric and partition metric versus the running times in log scale using 1000 datasets simulated from ultrametric trees and non-clock trees, respectively. On average, CSMC outperforms MB in terms of obtaining a smaller tree distance with a smaller number of steps. Regarding the partition metric, our SMC algorithm is 100 times, or 2 orders of magnitude, faster than MCMC to reach 0. The vertical line shows the 50% credible intervals of the estimates. For ultrametric trees, the variance of estimates using SMC is smaller than or equal to those from MB. For non-clock trees, with a small number of particles, SMC is obviously better. After reaching convergence, SMC and MB performs similarly.

4.3.3 RNA Data

We analyzed aligned curated ribosomal RNA (rRNA) sequences obtained from 199 species of Chloroplast [13] [3]. We randomly selected four sub-datasets of 5, 10, 20, and 30 taxa, respectively, from the whole dataset. Numbers of sites are

[3]The dataset was downloaded from http://www.rna.ccbb.utexas.edu/DAT/3C/Alignment/ on 08/09/2011

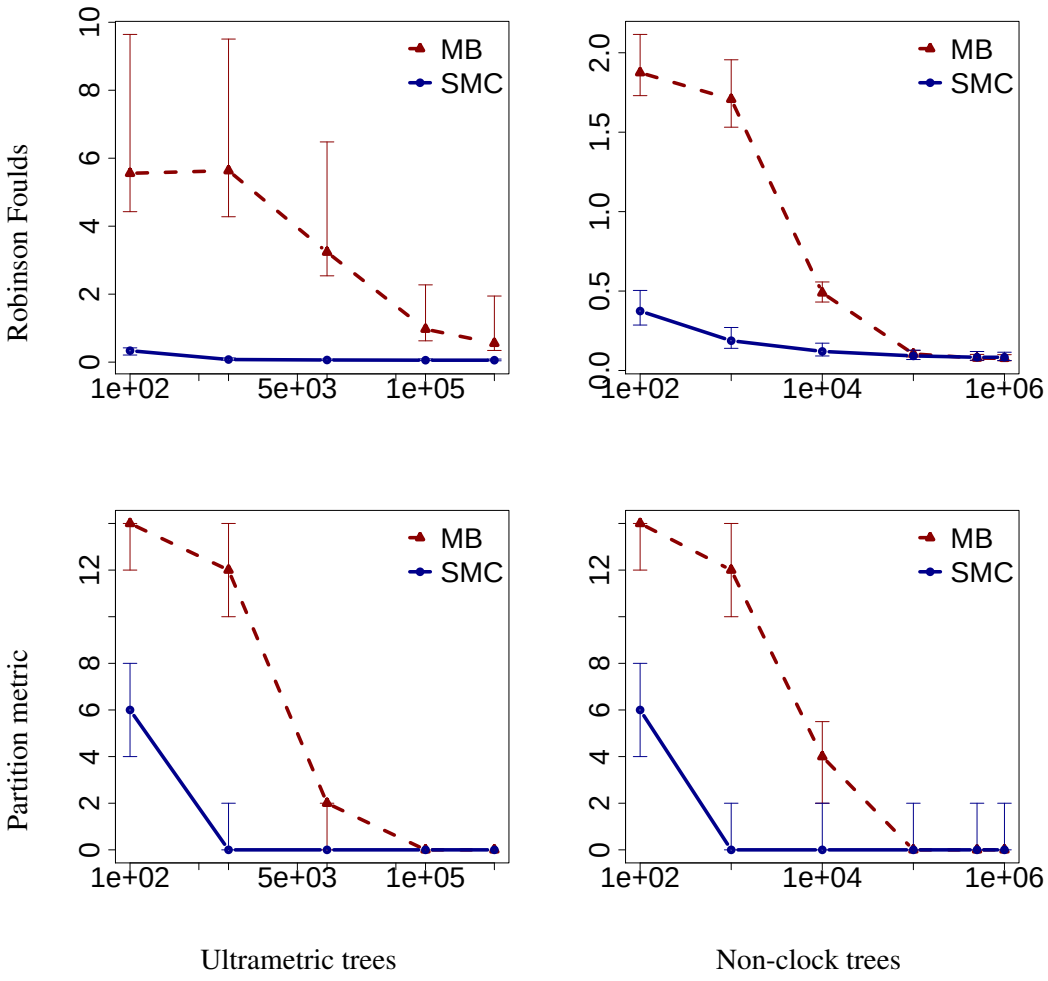

Figure 4.7: The average Robinson Foulds metric and partition metric versus the running times (the number of times the peeling recurrence is calculated) in log scale using 1000 datasets simulated from ultrametric trees (left) and non-clock trees (right).

121, 121, 126, and 140, respectively. Using a K2P model with $\kappa = 2$, both SMC and MrBayes were applied to these datasets. Figure 4.8 shows the log-likelihood of the consensus trees of using CSMC and MrBayes versus the running times (divided by 1000). We can see that CSMC can obtain higher log-likelihood than MB using a smaller running time. For large numbers of iterations and particles, the MCMC sampler slightly outperformed CSMC on the real data of 30 taxa.

4.4 Discussion

We have presented a general class of SMC methods that provide an alternative to MCMC for Bayesian phylogenetic inference. Our proposed method, CSMC, can efficiently explore the huge phylogenetic tree space by making use of the order-theoretic notion of a poset. Our CSMC methods can do inference for not only ultrametric trees but also non-clock trees which have not been approached by SMC previously. Both theoretical and experimental results show that CSMC can obtain a consistency estimate of the expectation of a function with respect to a target measure. In addition, extensive numerical simulations show that CSMC can obtain comparable results to the MCMC method in terms of computation speed and estimation accuracy.

4.4.1 Choices of the Partial States and Their Distributions

Besides the natural forest extension, described in Section 4.2.1, there are other potential extensions of π into forests. As an example, we can define $\pi(s_r) = p(\mathcal{Y}, NJ(s_r))$, where $NJ(s_r)$ refers to a complete phylogenetic tree that is obtained by fixing the subtrees in the current forest s_r and applying the neighbor joining (NJ) method, briefly reviewed in Section 3.3.1, to the roots of s_r.

Moreover, there are more than one way to define the sequence of partial states as long as the objects obtained at the last iteration of CSMC are in the target space, i.e. a phylogenetic tree in our example. The method we have used is based on building a complete tree by starting from the disconnected graph and by connecting a pair of subtrees in a forest at every step. An alternative, called 'star-shaped decomposition', is to build a complete tree by starting with all leaves of molecular

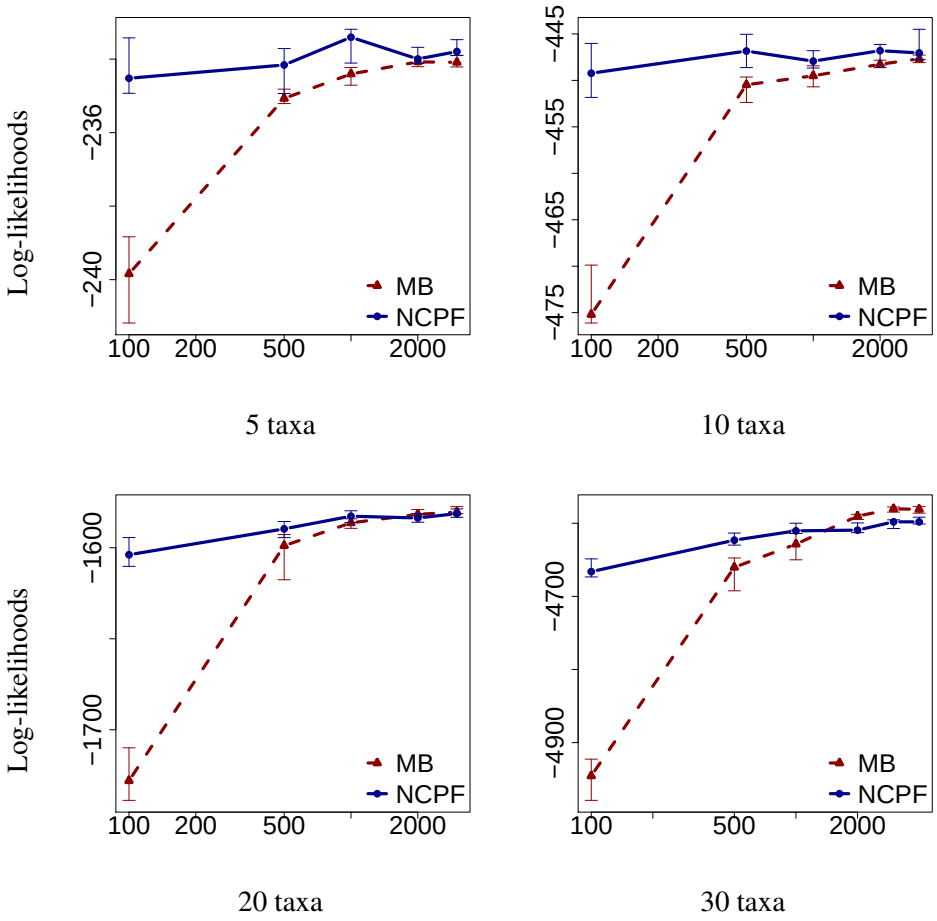

Figure 4.8: Log-likelihoods of the consensus trees using CSMC and MB versus the running times (divided by 1000) using RNA datasets of 5, 10, 20, and 30 taxa, respectively. Note that 'NCPF' on the plot represents 'particle filter for non-clock trees', i.e. CSMC for non-clock trees.

sequences connected to a single centre node and separating a pair of nodes attached to the centre node at each step. The likelihood of a star-shaped partial state can be computed in a similar way as a binary tree using Equation (3.3).

4.4.2 Choices of the Proposal Distributions

The design of the proposal distributions has significant impact on the performance of the CSMC algorithm. We have the flexibility to choose forward proposal distributions as long as it induces a poset on all possible partial states. In addition to the simple proposal distribution used in Section 4.2.2, an alternative proposal distribution is to consider all possible pairs of subtrees to merge, i.e.

$$\nu^{+}(s_{r-1} \to s_r) = \frac{\pi(s_r)}{\sum_{s'_r : s_{r-1} < s'_r} \pi(s'_r)}.$$

This is called Prior-Post proposal in [43, 105].

For a partial state s_{r-1} of m subtrees, the Prior-Post proposal considers $m(m-1)/2$ possible pairs of subtrees, which is computationally expensive for a large number m. From our simulation studies, intensive computing for each particle allows us to use a smaller number of particles to obtain the same estimation accuracy. However, there is no obvious benefits in terms of the total computing time. Therefore, we choose to use the simple proposal distribution in this dissertation.

4.4.3 Resampling Strategies

CSMC algorithms can be improved by using more sophisticated resampling strategies that have been developed for other SMC algorithms, for example, stratified sampling [63] and adaptive resampling [24]. Other potential improvements are reviewed in [14] and [26].

4.4.4 Another Application: Self-avoiding Walk

Although the proposed method is motivated by Bayesian phylogenetic inference, CSMC is general enough to be applied to many other applications. Bayesian inference problems in computational biology, e.g., statistical alignment for proteins and DNA and grammatical inference for RNA, may benefit from the CSMC framework.

One of other applications is to compute the total number of different self-avoiding walks (SAWs) of a fixed length.

SAWs in a two- or three-dimensional lattice space have been studied as the most basic model for chain polymers in polymer physics [85, 110, 111]. In a two-dimensional lattice model, a chain polymer of length N is fully characterized by the positions of all of its molecules, $\boldsymbol{x} = (x_1, \cdots, x_N)$, where $x_i = (a, b)$, and a and b are integers. The distance between x_i and x_{i+1} has to be exactly 1, and $x_{i+1} \neq x_k$ for all $k \leq i$.

The target distribution of interest is $\pi(\boldsymbol{x}) = 1/Z_N$, where the space of $\boldsymbol{x}$ is the set of all SAWs of length N and Z_N is a normalizing constant, which is equal to the total number of different SAWs of length N.

Let s_0 be the walk of length 0 as well as the initial state. We sequentially generate partial states $s_1, s_2, \cdots$, by using a proposal distribution with density a $\nu^+(s_{r-1} \to s_r)$: for the partial state s_{r-1}, we have a walk of length $r-1$ and we generate a partial state s_r of length r by uniformly inserting one step to the walk s_{r-1}. We require $s_N \equiv \boldsymbol{x}$.

Previous work [110] can be shown to be a special case of our framework where the backward proposal distribution has the following density $\nu^-(s_r \to s_{r-1}) = 1/r$. After running the CSMC algorithm, Z_N can be estimated by $\|\gamma_{r,K}\| = \prod_r \frac{1}{K} \sum_k w_{r,k}$.

Chapter 5

Combining MCMC and SMC

In previous chapters, we have introduced the applications of MCMC and SMC in Bayesian phylogenetic tree inference. As the two important types of Monte Carlo methods, they have their own strengths and drawbacks. MCMC is a class of very powerful methods that have a rich literature. The main problem with MCMC is that it is typically difficult to design appropriate proposal distributions for high-dimensional and multimodal target distributions. Consequently, the chain often mixes poorly and it takes an extremely long time to reach convergence. On the contrary, our proposed CSMC is fast in computation; but it requires to run a large number of particles simultaneously.

It is tempting to study combinations of MCMC and SMC to utilize their advantages and circumvent their limitations. Since the main limitation of using MCMC is unrealistic running time while the main constraint of using SMC is computer memory, an ideal framework of their combination should provide a flexible way of allocating the available total budget, including time and computing resources.

In this chapter, we will investigate two frameworks for combining SMC and MCMC in applications to phylogenetics. The first one is a set of particle Markov chain Monte Carlo (PMCMC) algorithms [3, 53], in which each MCMC iteration uses our proposed CSMC algorithm to approximate the posterior distribution of the phylogenetic trees and the marginal likelihoods of data, given parameters in the evolutionary model. Using PMCMC, we can incorporate global uncertainty of

hyperparameters, e.g. random nucleotide evolutionary models.[1] This is of concern since modelling uncertainty over evolutionary processes is one of the key selling point of Bayesian phylogenetic methods [25]. The second framework is to embed the standard MCMC algorithms in phylogenetics literature [52, 66, 67, 70], reviewed in Section 3.4.4, into an SMC sampler proposed by [22, 23], reviewed in Section 2.2.3.

5.1 Particle MCMC for Phylogenetics

In phylogenetics, our goal is to estimate the phylogenetic tree t and evolutionary parameters θ based on n observed biological sequences $\mathcal{Y}$. Using Bayesian approaches, the target distribution is the posterior distribution of t and θ, i.e. $\pi(t,\theta) \equiv p(t,\theta|\mathcal{Y})$. When the evolutionary parameters, θ, are assumed to be known or fixed, the target distribution is denoted $\pi_\theta(t) \equiv p_\theta(t|\mathcal{Y})$.

In Section 2.3, we have reviewed the generic PMCMC algorithms, including PIMH, PMMH, and PGS. In this section, we will focus on practical details when applying these algorithms to estimate the phylogenetic trees and evolutionary parameters using our proposed CSMC algorithm described in the previous chapter.

5.1.1 The Particle Independent Metropolis-Hastings (PIMH)

If estimating the evolutionary parameters is not of interest, we can use the PIMH, described in Algorithm 5.1, to approximate the posterior of phylogenetic tree t, i.e. $p_\theta(t|\mathcal{Y})$. Algorithm 5.1 is a special case of the generic PIMH (Algorithm 2.6), where the CSMC algorithm (Algorithm 4.1) is used in each iteration of the PIMH.

Although the PIMH algorithm and the SMC algorithm perform similarly in terms of estimation accuracy with the same total number of particles [3, 53], it is quite useful to have this alternative for Bayesian phylogenetics because of a practical reason. The implementation of an SMC algorithm is often limited by the computer memory capacity due to the fact that a large number of particles are required simultaneously. It may not be a problem when each particle is small. However, this is not the case in phylogenetics because typically each particle is a

[1] We use the 'hyperparameter' terminology because we view the phylogenetic tree as a parameter itself.

partial or a complete phylogenetic tree of many biological sequences with hundreds and thousands of sites. Contrary to SMC, the PIMH algorithm allows us to run a smaller number of particles in each iteration and run it for any specified amount of time until a sufficient number of particles have been generated. We will use an example to demonstrate the similarity in performance between PIMH and SMC in Section 5.1.4.

Algorithm 5.1 The PIMH Algorithm for Phylogenetics

1. Initialization, $i = 0$,
   (a) run the CSMC algorithm (Algorithm 4.1) targeting $p_\theta(t|\mathcal{Y})$,
   (b) sample $t(0) \sim \hat{p}_\theta(\cdot|\mathcal{Y})$ and compute $\hat{p}_\theta(\mathcal{Y})(0)$.
2. For iteration $i \geq 1$,
   (a) run the CSMC algorithm (Algorithm 4.1) targeting $p_\theta(t|\mathcal{Y})$,
   (b) sample $t^* \sim \hat{p}_\theta(\cdot|\mathcal{Y})$ and calculate $\hat{p}_\theta(\mathcal{Y})^*$ using (2.3),
   (c) with probability

$$\min\left\{1, \frac{\hat{p}_\theta(\mathcal{Y})^*}{\hat{p}_\theta(\mathcal{Y})(i-1)}\right\},$$

   set $t(i) = t^*$ and $\hat{p}_\theta(\mathcal{Y})(i) = \hat{p}_\theta(\mathcal{Y})^*$; otherwise set $t(i) = t(i-1)$ and $\hat{p}_\theta(\mathcal{Y})(i) = \hat{p}_\theta(\mathcal{Y})(i-1)$.

5.1.2 Particle Marginal Metropolis-Hastings (PMMH)

The PMMH algorithm approximates a 'marginal Metropolis-Hastings' (MMH) update targeting directly the marginal density $p_\theta(\mathcal{Y})$ that is estimated as a byproduct of the CSMC algorithm using

$$Z_{R,K} = \|\gamma_{R,K}\| = \prod_{r=1}^{R}\left(\frac{1}{K}\sum_{k=1}^{K} w_{r,k}\right).$$

Algorithm 5.2 explicitly rewrite the PMMH algorithm for phylogenetics with our CSMC Algorithm 4.1 in each iteration.

Algorithm 5.2 The PMMH for Phylogenetics

1. Initialization, $i = 0$,
   (a) set $\theta(0)$ arbitrarily and
   (b) run the CSMC algorithm targeting $p_{\theta(0)}(t|\mathcal{Y})$, sample $t(0) \sim \hat{p}_{\theta(0)}(\cdot|\mathcal{Y})$ and let $\hat{p}_{\theta(0)}(\mathcal{Y})$ denote the marginal likelihood estimate.
2. For iteration $i \geq 1$,
   (a) sample $\theta^* \sim q(\theta(i-1) \to \cdot)$,
   (b) run the CSMC algorithm targeting $p_{\theta^*}(t|\mathcal{Y})$, sample $t^* \sim \hat{p}_{\theta^*}(\cdot|\mathcal{Y})$ and let $\hat{p}_{\theta^*}(\mathcal{Y})$ denote the marginal likelihood estimate, and
   (c) with probability

$$\min\left(1, \frac{\hat{p}_{\theta^*}(\mathcal{Y})p(\theta^*)}{\hat{p}_{\theta(i-1)}(\mathcal{Y})p(\theta(i-1))}\frac{q\{\theta^* \to \theta(i-1)\}}{q\{\theta(i-1) \to \theta^*\}}\right) \tag{5.1}$$

   set $\theta(i) = \theta^*$, $t(i) = t^*$, and $\hat{p}_{\theta(i)}(\mathcal{Y}) = \hat{p}_{\theta^*}(\mathcal{Y})$; otherwise set $\theta(i) = \theta(i-1)$, $t(i) = t(i-1)$, and $\hat{p}_{\theta(i)}(\mathcal{Y}) = \hat{p}_{\theta(i-1)}(\mathcal{Y})$.

Recall that there is only one evolutionary parameter κ in the K2P evolutionary model, i.e. $\theta = \kappa$. We choose an exponential distribution with rate μ_0 as the prior for κ, i.e. $\kappa \sim \text{Exp}(\mu_0)$. Regarding the proposal distribution $q\{\kappa \to \cdot\}$, one choice is to propose a new κ^* by using a multiplier of κ [66]; i.e. let $\kappa^* = m\kappa$, where $m \sim \text{Unif}(\frac{1}{a}, a)$ with a prefixed tuning parameter $a > 1$. A lower value of a will lead to a higher acceptance rate, which can also mean that the MCMC chain will mix slowly. By [66], the proposal ratio in the MH ratio is

$$\frac{q\{\kappa^* \to \kappa(i-1)\}}{q\{\kappa(i-1) \to \kappa^*\}} = m. \tag{5.2}$$

The prior ratio is

$$\frac{p(\kappa^*)}{p(\kappa)} = \frac{\mu_0 \exp(-\mu_0 \kappa^*)}{\mu_0 \exp(-\mu_0 \kappa)} = \exp(\mu_0(\kappa - \kappa^*)) = \exp(\mu_0 \kappa(1-m)).$$

In each implementation of the CSMC algorithm, we use a total number of K particles. By results of [3, 53], for any fixed number $K \geq 1$ of particles, the transition kernels of PMCMC leave the target density of interest, $p(\theta, t|\mathcal{Y})$, invariant, and under weak assumptions the PMCMC sampler is ergodic. In practice, how to choose the number K is an issue that will affect the performance of a PMCMC algorithm.

5.1.3 The Particle Gibbs sampler (PGS)

Another PMCMC algorithm that can do joint estimation of phylogenetic trees and evolutionary parameters is the particle Gibbs sampler (PGS) described in Algorithm 2.9. Contrary to the PMMH algorithm that bypasses the phylogenetic tree in calculating the MH ratio by marginalizing it, PGS requires a special type of PMCMC update called the *conditional SMC* update. To demonstrate how PGS applies to phylogenetics, we rewrite the two algorithms, the conditional SMC (Algorithm 2.8) and the PGS (Algorithm 2.9), explicitly for phylogenetics in Algorithm 5.3 and Algorithm 5.4, respectively, with more details and explanations.

The introduction of the notation for ancestral lineage in the conditional SMC (Algorithm 2.8) is mainly for the purpose of asymptotic proofs in [3]. In the conditional CSMC algorithm for phylogenetic trees (Algorithm 5.3), we abandon these complicated notations for ancestral lineage. Instead, to simplify notations and implementation, we switch the positions of particles such that the first particle is the frozen one. Obviously, we can switch particles without affecting the particle approximation towards the target distributions. In addition, Algorithm 5.3 is a conditional version of the CSMC algorithm (Algorithm 4.1) whilst Algorithm 2.8 is a conditional version of the generic traditional SMC.

Recall that K is the number of particles in the CSMC algorithm. Let $s'_{1:R} = (s'_1, s'_2, \cdots, s'_R)$ denote the frozen particle path that the conditional CSMC algorithm is conditioned on. In other words, this path always survives whereas the remaining

Algorithm 5.3 Conditional CSMC for Phylogenetic Trees

$s_{0,k} \leftarrow \perp$, $\forall k \in \{1, \cdots, K\}$
$w_{0,k} \leftarrow 1/K$
construct $\gamma_{0,K}$ using Equation (4.1)
for $r \in 1, \ldots, R$, **do**
  $s_{r,1} \leftarrow s'_r$
  $w_{r,1} \leftarrow \frac{\gamma(s_{r,1})}{\gamma(s_{r-1,1})} \cdot \frac{\nu^-(s_{r,1} \to s_{r-1,1})}{\nu^+(s_{r-1,1} \to s_{r,1})}$
  for $k \in 2, \cdots, K$, **do**
    sample $\tilde{s}_{r-1,k} \sim \pi_{r-1,K}$
    $s_{r,k} \sim \nu^+(\tilde{s}_{r-1,k} \to \cdot)$
    $w_{r,k} \leftarrow \frac{\gamma(s_{r,k})}{\gamma(\tilde{s}_{r-1,k})} \cdot \frac{\nu^-(s_{r,k} \to \tilde{s}_{r-1,k})}{\nu^+(\tilde{s}_{r-1,k} \to s_{r,k})}$
  end for
  construct $\gamma_{r,K}$ using Equation (4.1)
end for
return $\gamma_{R,K}$

$K-1$ particles are generated as in the CSMC algorithm.

Now we will apply the conditional CSMC (Algorithm 5.3) in the framework of the generic PGS to estimate the joint posterior of θ and t, summarized in Algorithm 5.4. Since we cannot obtain the closed-form full conditional distribution for θ, i.e. $p(\theta|t, \mathcal{Y})$, we use a step of the Metropolis-Hastings algorithm within Gibbs sampling to sample θ from $p(\theta|t, \mathcal{Y})$, where t is the current phylogenetic tree. The MH ratio for accepting a newly proposed θ^* is

$$\begin{aligned} \alpha(\theta \to \theta^*|t) &= \min\left(1, \frac{p(\theta^*|t, \mathcal{Y})}{p(\theta|t, \mathcal{Y})} \frac{q\{\theta^* \to \theta\}}{q\{\theta \to \theta^*\}}\right) \\ &= \min\left(1, \frac{p(\mathcal{Y}|\theta^*, t)p(\theta^*|t)}{p(\mathcal{Y}|\theta, t)p(\theta|t)} \frac{q\{\theta^* \to \theta\}}{q\{\theta \to \theta^*\}}\right), \end{aligned} \tag{5.3}$$

where the likelihood $p(\mathcal{Y}|\theta, t)$ can be calculated by (3.3). A common assumption is that the priors for θ and t are independent, i.e. $p(\theta|t) = p(\theta)$. In the K2P model, the only parameter is κ, i.e. $\theta = \kappa$. The same prior as described in the PMMH algorithm can be used; i.e. the prior $\kappa \sim \text{Exp}(\mu_0)$, and the proposal $q\{\kappa \to \kappa^*\} = m\kappa$,

where $m \sim \text{Unif}(\frac{1}{a}, a)$ with a prefixed tuning parameter $a > 0$. As a result,

$$\alpha(\kappa \to \kappa^* | t) = \min\left(1, \frac{p(\mathcal{Y}|\theta^*, t)}{p(\mathcal{Y}|\theta, t)} \cdot e^{\mu_0 \kappa(1-m)} \cdot m\right).$$

Algorithm 5.4 The MH within Particle Gibbs Sampler for Phylogenetics

Initialization: $i = 0$
  Sample $\theta(0)$ arbitrarily
  Run the CSMC algorithm targeting $p_{\theta(0)}(t|\mathcal{Y})$
  Sample $t(0) \sim \hat{p}_{\theta(0)}(\cdot|\mathcal{Y})$ and record its ancestral lineage.
for $i \geq 1$ **do**
  Sample $\theta(i) \sim p(\cdot|t(i-1))$
    (a) Draw $\theta^* \sim q(\theta(i-1) \to \cdot)$
    (b) with probability $\alpha(\theta(i-1) \to \theta^*|t(i-1))$ calculated by (5.3), set $\theta(i) = \theta^*$; otherwise set $\theta(i) = \theta(i-1)$.
  Run the conditional CSMC algorithm targeting $p_{\theta(i)}(t|\mathcal{Y})$ conditional on $t(i-1)$ and its ancestral lineage.
  Sample $t(i) \sim \hat{p}_{\theta(i)}(\cdot|\mathcal{Y})$.
end for

5.1.4 Simulation Studies

In this section, we evaluated PIMH and PMMH using some simulated datasets. In order to simulate the datasets, we first generated a set of random trees, including topologies and branch lengths, as the reference trees. Then, for each reference tree, we simulated DNA sequences using the K2P model with parameter $\kappa = 2.0$, described in Section 3.4.1. For each unrooted tree, we chose an arbitrary point on the tree as its root. The data generation started from the root of a tree by randomly sampling from the stationary distribution of the CTMC (see Section 3.4.1). Assuming site independence, we generated the data for the children of the root using the transition probability computed by (3.1). This procedure was recursively implemented until reaching the leaves. We discarded the data at the internal nodes and took the data on leaves as the simulated observations. Table 5.1 summarizes the settings for the simulated datasets that will be used in this section.

Dataset	Tree type	Data type	# Tree	# taxa	# sites	Tree rate
1	clock	DNA	100	5	200	1.0
2	non-clock	DNA	100	10	200	1.0

Table 5.1: Settings for the simulated data.

We constructed a consensus tree using the random trees from the results of applying the PMCMC algorithms for phylogenetics to each simulated dataset. These estimated consensus trees were assessed by their log-likelihoods, and their tree distances from the reference trees measured by PM, RF, and KF tree metrics. Please refer to Section 3.5 for the definition of the consensus tree and tree distance metrics.

PIMH Versus CSMC

In this study, we compared the performance of PIMH and SMC on the same datasets using the same computing cost, i.e. a fixed number of total particles. As mentioned earlier in this chapter, the use of SMC algorithms is limited by computer memory capacity. For example, an "out-of-memory" error occurred when we tried to run 5,000,000 particles simultaneously for a dataset of 10 taxa of 200 sites using a 4-gigabytes memory limit. To avoid such a problem, we used a set of small clock trees of 5 taxa (Dataset 1 of Table 5.1) without loss of generality for the purpose of illustration.

In the CSMC algorithm, all these particles are run simultaneously. In PIMH algorithms, we need to decide the number of particles used at each iteration, which is shown in the first column of Table 5.2. We used a total number of 1,000,000 particles. From the results in Table 5.2, PIMH and SMC performed very similarly in terms of the log-likelihood of the consensus trees and their distances from the reference trees.

PMMH and MrBayes

The purpose of this simulation study is to evaluate PMMH in terms of estimating the evolutionary parameter κ and the phylogenetic trees using a set of simulated

Method	#particles/iter.	# iteration	log-likelihood	RF	PM	KF
CSMC	1000,000	-	-1091.32 (153.59)	0.063 (0.048)	0 (0)	1.38 (3.34)
PIMH	10	100,000	-1091.40 (153.60)	0.062 (0.046)	0 (0)	1.38 (3.33)
PIMH	100	10,000	-1091.40 (153.64)	0.062 (0.046)	0 (0)	1.38 (3.35)
PIMH	1,000	1,000	-1091.39 (153.62)	0.061 (0.046)	0 (0)	1.41 (3.41)
PIMH	10,000	100	-1091.45 (153.65)	0.062 (0.046)	0 (0)	1.34 (3.24)

Table 5.2: Tree estimates using PIMH for clock trees. Log-likelihood refers to that of the consensus tree.

Methods	log-likelihood	κ	RF	PM	KF
PMMH	-1894.84 (162.01)	1.99 (0.25)	1.44 (1.43)	0 (0.45)	0.45 (3.76)
MrBayes	-1898.89 (175.43)	3.72 (0.47)	1.62 (1.13)	0 (1.35)	0.56 (3.15)

Table 5.3: Comparison of PMMH and MrBayes for a set of non-clock trees in terms of estimating the parameter κ (true value 2.0) and the phylogenetic tress. Log-likelihood refers to that of the consensus tree.

non-clock trees (Dataset 2 of Table 5.1). We used 10,000 particles at each iteration of PMMH and ran 5,000 iterations. For the purpose of comparison, we also used MrBayes to run MCMC for 50,000,000 MCMC iterations on the same datasets such that the total computing cost is comparable to that of PMMH.

PMMH outperformed MrBayes from the results for the two algorithms using the 100 simulated non-clock datasets, shown in Table 5.3. For the results of PMMH, we also calibrated the credible intervals using coverage probabilities, shown in Figure 5.1. As expected, the coverage probabilities (circles) are close to the values of credible intervals (solid line). In addition, Figure 5.2 shows the trace plot and histogram of κ estimated by PMMH for three arbitrarily selected non-clock datasets.

5.1.5 Real Data Analysis

We also applied PMMH for non-clock trees to the same datasets in Section 4.3.3 with 1000 particles at each iteration. By estimating κ rather than using a pre-

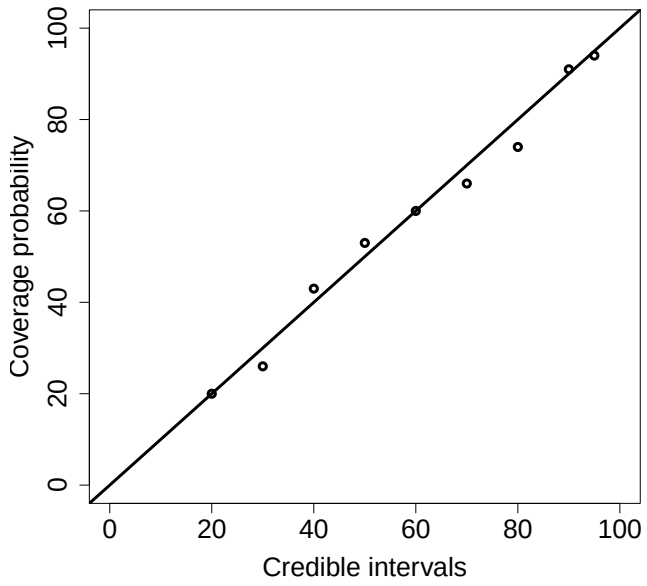

Figure 5.1: Coverage probability versus credible intervals for the estimates of κ using PMMH.

fixed value, the log-likelihoods of the obtained consensus trees increased, shown in Figure 5.3 as an example. The parameter a in PMMH was set to be 1.2. Better performance might be achieved by choosing a more sensible value.

In order to evaluate the influence of the number of particles at each PMMH step, we set this number to be 100, 1000, 10000 and 25000, respectively, and compared the log-likelihoods of the consensus trees using the datasets of 5, 10, and 20 taxa, shown in Table 5.4. We found that the smaller the number of particles, the lower log likelihood and higher variance. Theoretically, we can always achieve higher accuracy by increasing the number of particles. In practice, running a large number of particles can be computationally expensive. Therefore, it is a trade-off between speed and accuracy.

We analyzed the aligned protein coding mitochondrial DNA sequences of 32 species of cichlid fishes [17, 64], described in Section 3.2, using PMMH for non-

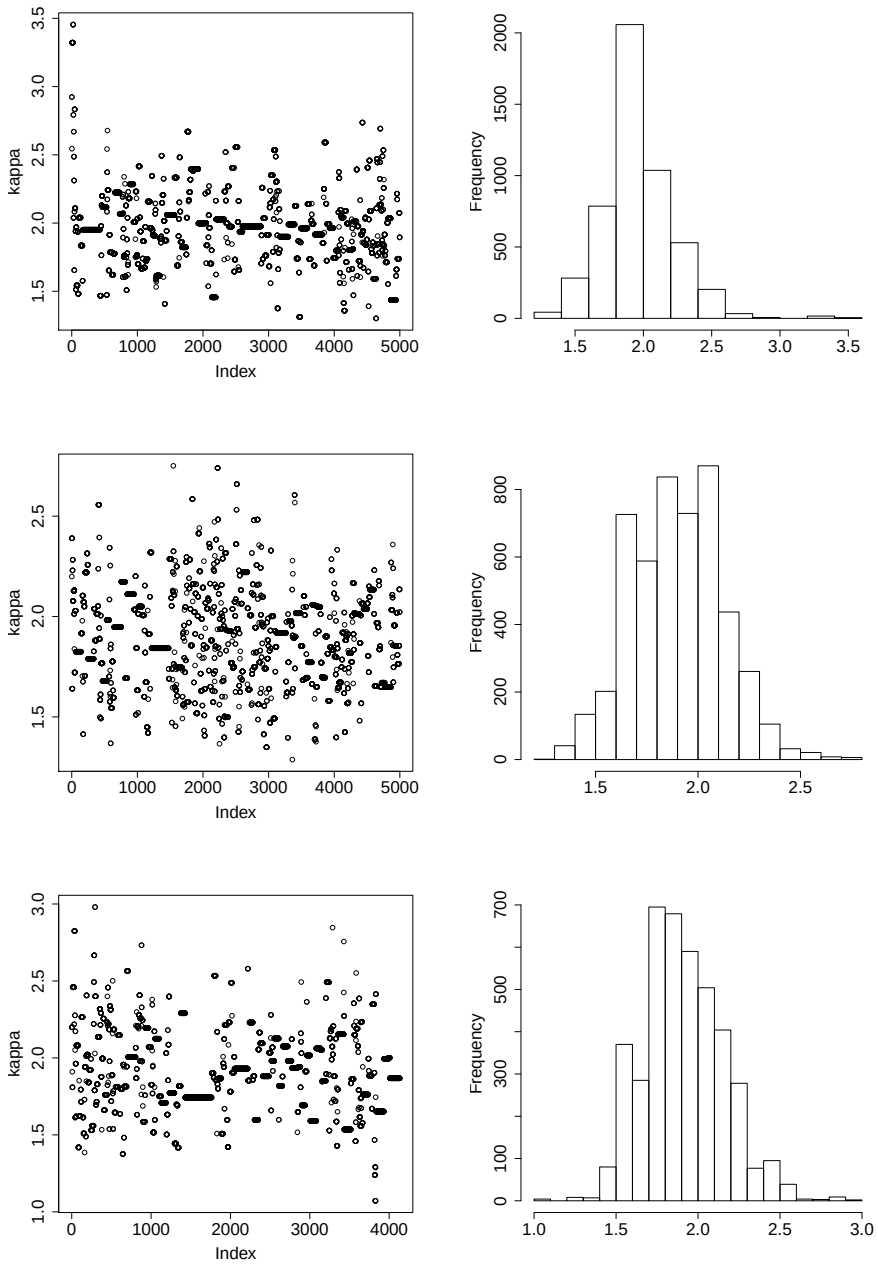

Figure 5.2: Trace plot and histogram of κ estimated by PMMH for three arbitrarily selected non-clock datasets.

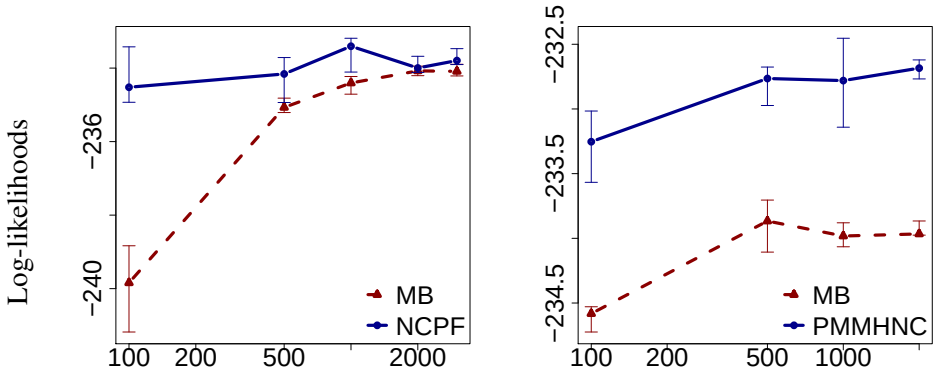

Figure 5.3: Log-likelihoods of the consensus trees versus the running times (divided by 1000) using RNA datasets of 5 taxa. (Left) We only estimated the phylogenetic tree with MB and CSMC. (Right) We estimated both the phylogenetic tree and the parameter κ with MB and PMMH.

# taxa	Number of particles			
	100	1000	10000	25000
5	-231.12 (0.18)	-231.13 (0.059)	-231.12 (0.086)	-231.10 (0.033)
10	-438.99 (2.06)	-436.64 (0.86)	-436.66 (0.62)	-436.28 (0.23)
20	-1608.41 (3.12)	-1581.16 (3.36)	-	-

Table 5.4: Log-likelihoods of the consensus trees obtained using PMMH for 3000 iterations with different numbers of particles at each iteration using non-clock trees.

clock trees. We used 10,000 particles for each MCMC iteration, and ran PMMH for 2,000 iterations. For comparison, we ran MrBayes on the same dataset for 20,000,000 iterations. The log-likelihoods of the consensus trees from PMMH and MrBayes were -6,892.51 and -6,942.20, respectively. Figure 5.4 shows the consensus tree estimated by PMMH, and Figure 5.5 by MrBayes. The result was consistent with previous studies [17, 64]. For example, the top clade of 7 species[2] on our consensus tree are species in the same tribe *Lamprologini* [64].

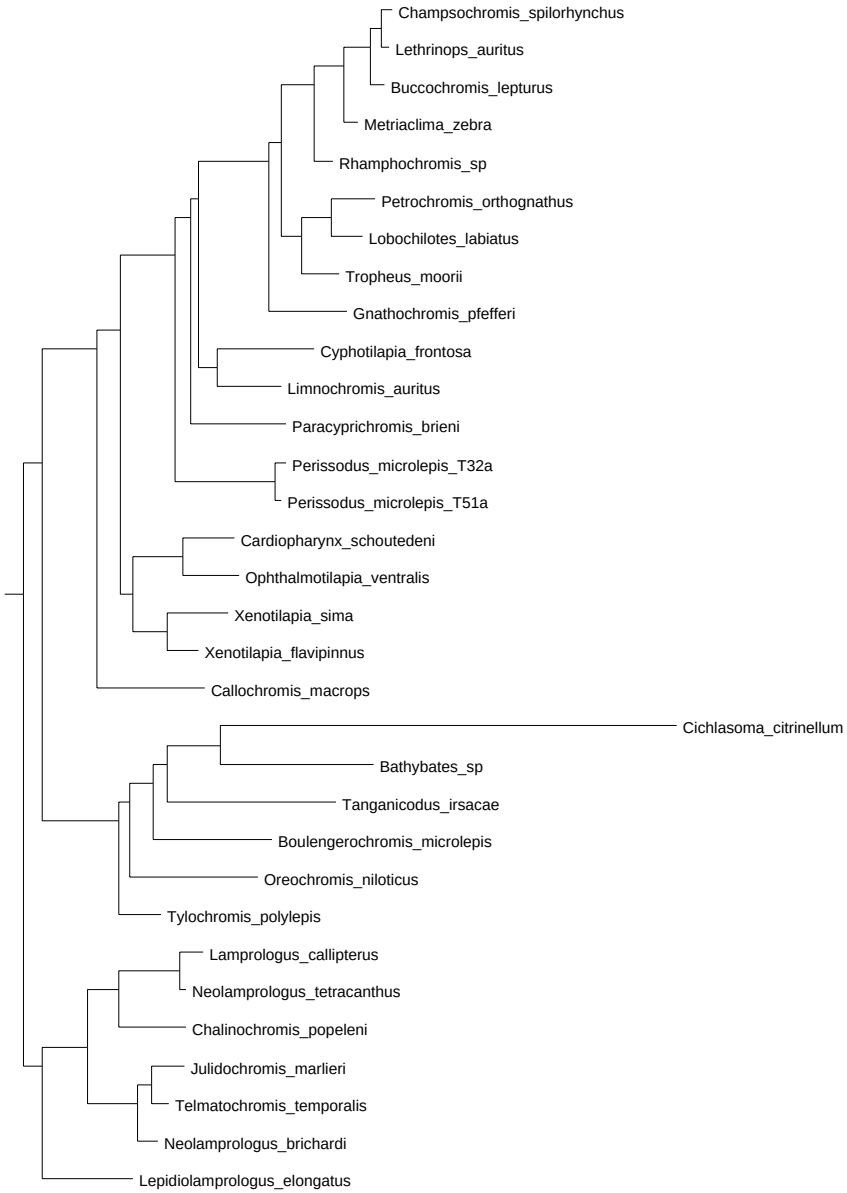

Figure 5.4: The consensus tree for cichlid fishes estimated by PMMH. The species on the root is *Lamprologus callipterus*.

Figure 5.5: Consensus tree for the dataset of Cichlid Fishes using MrBayes. The species on the root is *Lamprologus callipterus*.

5.2 An SMC Sampler with MCMC Kernels

As reviewed in Section 2.2.3, SMC samplers [22, 23] provide a framework of converting an MCMC algorithm for a static distribution π into an SMC algorithm by doing MCMC moves within SMC iterations. In this section, we describe how the standard MCMC algorithms in Bayesian phylogenetics can be used within an SMC algorithm.

In Bayesian phylogenetics, the target distribution of interest is the joint posterior of a phylogenetic tree t and evolutionary parameters θ, i.e. $\pi(t,\theta) \equiv p(t,\theta|\mathcal{Y})$. For simplicity of notation, we denote $x = (t,\theta)$.

We define

$$\pi_r(x) \propto \pi(x)^{\phi_r}, \tag{5.4}$$

where $0 \leq \phi_1 < \cdots < \phi_R = 1$, and $\pi_R(x) = \pi(x) = p(x|\mathcal{Y})$.

We will use the SMC sampler (Algorithm 2.5) with the backward kernel,

$$L_{r-1}(x_r, x_{r-1}) = \pi_r(x_{r-1})K_r(x_{r-1}, x_r)/\pi_r(x_r).$$

As mentioned in Section 2.2.3, with this backward kernel, the incremental importance weight becomes $\gamma_r(x_{r-1})/\gamma_{r-1}(x_{r-1})$. More precisely, using Equation (5.4), we have

$$\gamma_r(x_{r-1})/\gamma_{r-1}(x_{r-1}) = \{\gamma(x_{r-1})\}^{\phi_r-\phi_{r-1}} = \{p(x_{r-1})p(\mathcal{Y}|x_{r-1})\}^{\phi_r-\phi_{r-1}}.$$

A common choice for the Markov kernels, $K_r(x_{r-1},\cdot)$, is to use MCMC kernels [22, 23]. A typical MH kernel used in an SMC sampler is composed of the following steps:

1. Let $q(x_{r-1},\cdot)$ be a proposal distribution. Propose a new tree and new evolutionary parameters, denoted x_r^*, from $q(x_{r-1},\cdot)$.

[2] *Telmatochromis temporalis, Neolamprologus tetracanthus, Neolamprologus brichardi, Lepidiolamprologus elongatus, Lamprologus callipterus, Julidochromis marlieri*, *Chalinochromis popelini*.

2. The MH ratio is computed as

$$\alpha(x_{r-1}, x_r^*) = \min\left\{1, \frac{\pi_r(x_r^*)q(x_r^*, x_{r-1})}{\pi_r(x_{r-1})q(x_{r-1}, x_r^*)}\right\}.$$

3. With probability $\alpha(x_{r-1}, x_r^*)$, the proposal x_r^* is accepted, and with (1-$\alpha(x_{r-1}, x_r^*)$) probability, the chain remains at x_{r-1}.

In phylogenetics, there is a rich literature on using MCMC algorithms to sample from the posterior and then do phylogenetic inference. In order to take an advantage of these methods, we can combine different MCMC samplers into mixtures and cycles of several individual samplers. This is justified by a very powerful and useful property of MCMC [2, 108]: if each of the transition kernels $\{K^i\}, i = 1, \cdots, M$, have invariant distribution π, then the *cycle hybrid kernel* $\prod_{i=1}^{M} K^i$ and the *mixture hybrid kernel* $\sum_{i=1}^{M} p_i K^i, \sum_{i=1}^{M} p_i = 1$, are also transition kernels with invariant distribution π.

Algorithm 5.5 summarizes the SMC sampler for phylogenetics where the proposal K_r^i can be any MCMC kernel, including those proposed in Bayesian phylogenetics literature, e.g. [52, 66, 67, 70]. In the numerical examples of this section, we used the proposals K_r^i defined as follow:

1. K_r^1: the *multiplicative branch proposal* described in Section 3.4.4.

2. K_r^2: the *global multiplicative branch proposal* that proposes all the branch lengths by applying the above multiplicative branch proposal to each branch.

3. K_r^3: the *stochastic nearest neighbor interchange (NNI) proposal* described in Section 3.4.4.

4. K_r^4: the *stochastic NNI proposal with resampling the edge* that uses the above nearest neighbor interchange (NNI) proposal and the multiplicative branch proposal for the edge under consideration.

5. K_r^5: the *independent branch proposal* that proposes an branch length for an arbitrary branch with an exponential distribution.

R	log-likelihood	RF	PM	KF
10	-2147.78(289.48)	0.81(0.01)	1.00(0.08)	3.16(21.97)
100	-1505.19(241.31)	0.43(0.13)	0.42(0.136)	0.27(42.39)
500	-1375.70(221.25)	0.17(0.14)	0(0.18)	0.036(154.43)
1000	-1371.15(203.14)	0.11(0.11)	0(0.16)	0.014(92.45)

Table 5.5: Comparison of the SMC sampler with MH moves with different number of iterations for a set of non-clock trees in terms of estimating the phylogenetic tress. Log-likelihood refers to that of the consensus tree. Here the tree distance metric, RF and PM, is the normalized version.

Note that we have only considered estimating phylogenetic trees in our current research. But this method can be used for estimating evolutionary parameters θ simply by using $\{K_r^i\}$ that propose θ.

Algorithm 5.5 An SMC Sampler using MH Kernels for Phylogenetics

$x_{1,k} \leftarrow \perp$, $\forall k \in \{1, \cdots, K\}$
$w_{1,k} \leftarrow 1/K$
for $r \in 2, \ldots, R$ **do**
  Sample $x_{r,k} \sim \sum_{i=1}^{M} p_i K_r^i(x_{r-1,k}, \cdot), \sum_{i=1}^{M} p_i = 1$
  $w_{r,k} \leftarrow \{p(x_{r-1,k})p(\mathcal{Y}|x_{r-1,k})\}^{\phi_r - \phi_{r-1}}$
  Normalize weights $W_{r,k} \propto w_{r,k}$, and resample $\{x_{r,k}, W_{r,k}\}$
end for

We are interested in how different choices of K and R affect the performance of Algorithm 5.5. In our numerical examples, we used a sequence of $\phi_i = i/R$ for simplicity. We simulated 100 non-clock trees with rate 10 and one dataset for each tree. Each sequence has length 200. Figure 5.6 shows the median (50% credible interval) of the log-likelihoods of the consensus trees and their tree distances from the reference trees. From these results, the log-likelihoods are very similar and the tree distances slightly decrease with different values of K. From the results of varying R with a fixed $K = 5000$, shown in Table 5.5, a larger number of R leads to a better estimate.

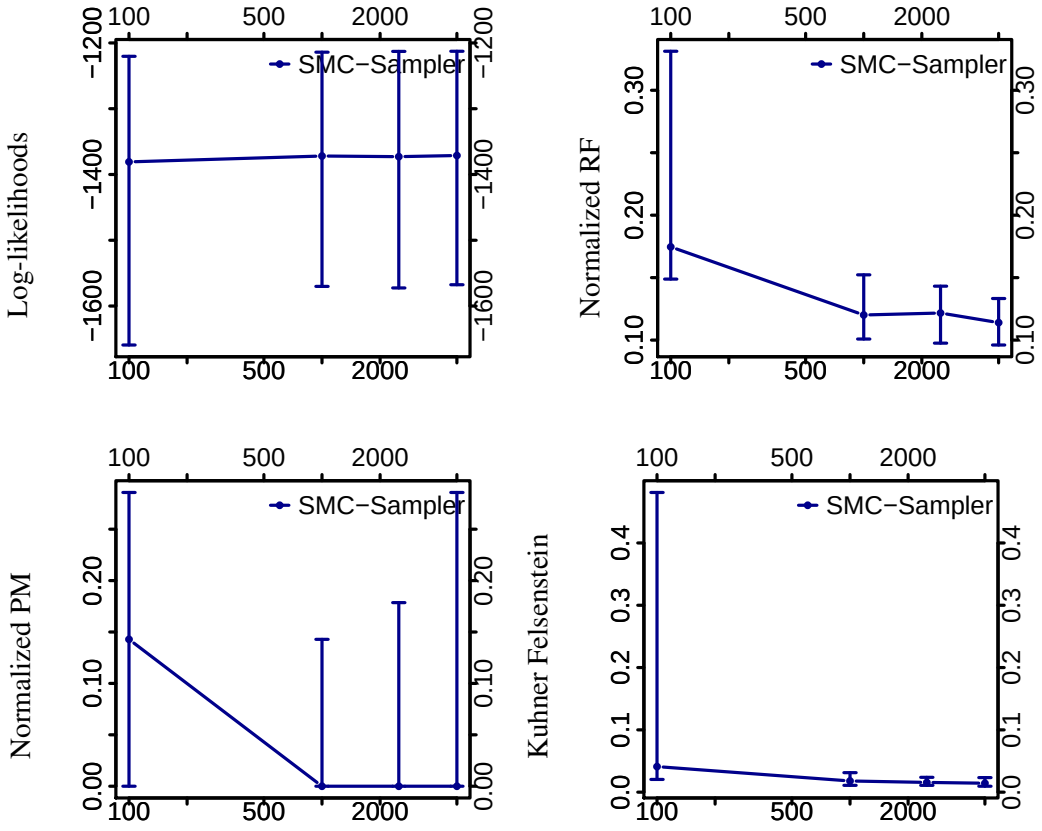

Figure 5.6: Results of running the SMC sampler with MH moves using different number of particles, K, for a set of simulated non-clock trees.

5.3 Parallel Computing

Inferring large scale trees is often computationally expensive or infeasible, which motivates us to develop parallel Monte Carlo algorithms to tackle the computational challenges in phylogenetics.

There are two types of parallelization, coarse-grain and fine-grain methods [100]. Coarse-grain parallelization refers to multi-core CPUs that are linked loosely through an Ethernet network where communication between nodes is supported by Message Passing Interface (MPI). This type of parallelization is limited by a com-

bination of latency, throughput, and cost. In contrast, fine-grain parallelization refers to shared memory multiprocessing, e.g. multi-threaded processing within a single computer, where communication between computing units is fast and cheap. In the following, we will focus on the fine-grain methods.

5.3.1 Fine-Grain Parallelization

There is a large literature on parallelization of both MCMC and SMC algorithms. SMC is intrinsically *parallelizable* algorithms. Most of the work has been on parallelization of the proposal steps [69]. Although MCMC algorithms are intrinsically *serial* algorithms, complicated statistical models typically require us to use advanced MCMC algorithms in which auxiliary variables are added to consider a larger space for the purpose of improving mixing of the Markov chain. Many advanced MCMC algorithms, e.g. slice sampling and parallel tempering [103], can be decomposed into fine-grain problems for parallelizing *within* MCMC iterations.

Algorithm 5.6 Outline of the Fine-grain Parallelization for SMC Algorithms

```
1: ... Serial code ...
2: for r = 1, ··· , R do
3:     Update particles to their r-th partial state using multiple threads.
4:     if Resampling is needed then
5:         Synchronize the threads.
6:         Resampling the particles.
7:         Synchronize the threads.
8:     end if
9: end for
10: ... Serial code ...
```

The art of designing efficient parallelization programming is essentially to reduce the serial computations and optimize the use of the multi-cores [69, 100]. In other words, the bottleneck is the fraction of irreducibly serial computations in the code. Fine-grain parallelization can be implemented by running multiple threads on a single CPU or on graphics processing units (GPUs) [69, 100].

Algorithm 5.6 outlines the fine-grain parallelization for SMC algorithms in which we distribute the computing for K particles to multiple threads (either on

CPU or GPUs). The bottleneck is the resampling step which involves synchronizing the threads. Fortunately, as mentioned in Section 2.2.2, resampling at every iteration is neither necessary nor efficient; instead, we resample only when the variation of the particle weights is higher than some prefixed threshold.

It is straightforward to do parallelization programming for PMCMC simply by using Algorithm 5.6 whenever an SMC algorithm is required. Furthermore, we can apply the idea of parallel tempering to PMCMC and implement subchains in parallel. Within each iteration of the PMCMC subchain, we implement the SMC algorithm in parallel. Therefore, parallel tempering PMCMC is highly parallelizable. To be concrete, we will briefly describe the parallel tempering PMCMC and its parallelization mainly based on [69] but in the context of phylogenetics and PMCMC.

The idea of using parallel tempering for PMCMC is to speed up convergence of the PMCMC chain for π defined on $\mathcal{X}$ by constructing a Markov chain on a joint space $\mathcal{X}^M$ using $M-1$ auxiliary variables each in $\mathcal{X}$. In other words, we have M parallel subchains each with stationary distribution $\pi_{(i)}, i \in \mathcal{M} = \{1, \cdots, M\}$ such that $\pi_{(M)} = \pi$. Each subchain i is associated with a PMCMC kernel that leaves $\pi_{(i)}$ invariant. The stationary distribution of the joint chain is $\tilde{\pi}(x_{1:M}) \equiv \prod_{i=1}^{M} \pi_{(i)}(x_i)$ with support in the joint space $\mathcal{X}^M$. By construction, $\tilde{\pi}$ admits π as a marginal distribution.

In phylogenetics, x is a description of the phylogenetic tree and evolutionary parameters. Using the parallel tempering methodology, we select $\pi_{(i)}(x) = \pi(x)^{\phi_i}$ with $0 < \phi_1 < \cdots < \phi_M = 1$. The rationale is that MCMC converges more rapidly when the target distribution is flatter. Hopefully, these auxiliary subchains can speed up convergence of the chain of interest by swapping their values [56]. The chains i and j swap their values with probability $\min\{1, \alpha_{ij}\}$ where

$$\alpha_{ij} = \frac{\pi_{(i)}(x_j)\pi_{(j)}(x_i)}{\pi_{(i)}(x_i)\pi_{(j)}(x_j)}.$$

It is obvious that the PMCMC subchains can be executed in parallel. However, the value-swapping moves among subchains need synchronization of the threads which becomes the bottleneck of using the parallel tempering. [69] proposed one way to exchange values between adjacent chains such that all value-

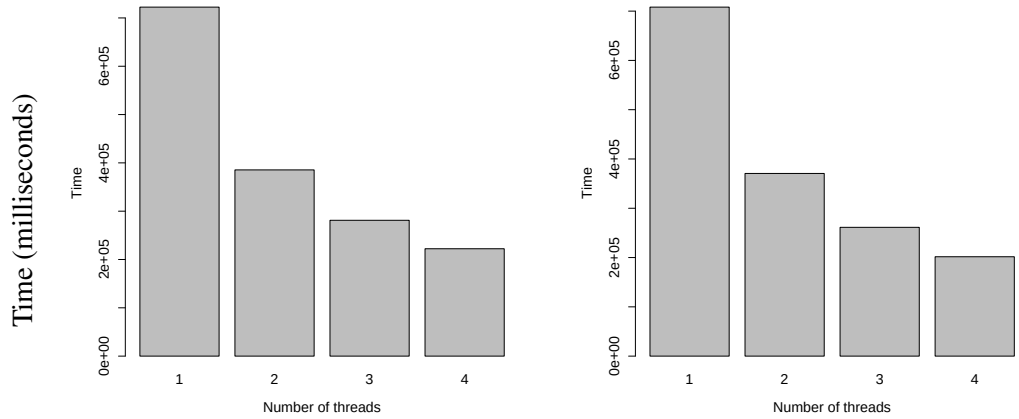

Figure 5.7: Time in milliseconds versus different number of threads for a dataset of an ultrametric tree (left) and a dataset of a non-clock tree (right) using CSMC.

swapping moves can be done in parallel, i.e. swapping values of each pair in either $\{\{1,2\},\{3,4\},\cdots,\{M-1,M\}\}$ or $\{\{2,3\},\{4,5\},\cdots,\{M-2,M-1\},\{M,1\}\}$, each with probability half.

5.3.2 Multiple Threads on a Single CPU

We provide some results of executing multiple threads on a single CPU to illustrate the advantage of parallelizing CSMC. We ran the CSMC algorithm on two simulated datasets from an ultrametric tree and a non-clock tree, respectively, of 30 taxa and 2000 sites using 100,000 particles with various number of threads on a 2.40GHz Intel Xeon 16-cores E7330 architecture. Figure 5.7 shows the computing time in milliseconds versus different number of threads for the two datasets. It is obvious to see the computing time decreases when the number of threads increases. If we parallelize more parts of our code, or use more machines, the CSMC algorithm will be even faster.

5.4 Discussion

PMCMC provides an alternative method to the standard MCMC approach to sample from the joint posterior of the phylogenetic trees and evolutionary parameters. We have demonstrated in our experiments that for a range of situations, PMCMC can achieve better estimates for the evolutionary parameters with the same computing cost. This result can be explained by comparing PMCMC and the standard MCMC, especially in the ways of proposing new states.

The main disadvantage of standard MCMC methods is that it is difficult to design proposal distributions that can efficiently explore the huge parameter space. Since the likelihood of a tree may change significantly with the change of the evolutionary parameters, proposing a new tree and a evolutionary parameter simultaneously or proposing a very different tree from the current one often leads to a high rejection rate. In a standard MCMC approach, this problem is alleviated by iterating between updating parameters given a fixed phylogenetic tree and updating the tree given the parameters with small moves. In other words, at each MCMC step, only a small perturbation is made on the current tree to propose a new tree.

To show the problem with this coordinate-wise strategy, we use a simplified situation as an illustration where the hyperparameter θ can only take two values, θ_1, and θ_2, and there are only two possible trees, t_1 and t_2. In other words, there are only 4 possible states: (θ_1, t_1), (θ_1, t_2), (θ_2, t_1), and (θ_2, t_2). At each iteration of MCMC, the chain is at one of these 4 states. Suppose the probability mass of the 4 states are 0.1, 0.3, 0.5, and 0.1, respectively. A good Markov chain should move quickly among the states with high probabilities. In this example, the states with higher probabilities are (θ_1, t_2) and (θ_2, t_1). However, if we start from the state (θ_1, t_2) in a standard MCMC in which we either fix θ to update t or fix t to update θ, the proposal is highly likely to be rejected. As a result, it is difficult to move from (θ_1, t_2) to (θ_2, t_1).

Contrary to the standard MCMC, PMCMC can make bold moves in the parameter space. Specifically, for a given θ, PMCMC will consider the whole population of trees and sample from them according to the posterior. As a result, the proposal is more sensible and more likely to be accepted, especially for high dimensional and multimodal distributions. Therefore, although the better proposals in PMCMC

are paid by the price of a much higher computing cost at each iteration, it is worthwhile because it leads to better estimates with a lower total cost.

The SMC sampler with MCMC moves provides another flexible framework to exploit the previous work in Bayesian phylogenetics using MCMC moves in an SMC algorithm. This method looks very similar to parallel tempering MCMC [103] in which subchains of tempered target distributions are implemented in parallel and value-swapping moves among subchains are used to help the chain for the target distribution to converge faster. The difference between the two methods is subtle. SMC samplers bypass the awkward value-swapping moves. In an SMC sampler, each tempered target distribution is approximated by a set of weighted particles at each SMC iteration. Contrary to running subchains with various temperatures in parallel, an SMC sampler starts from a very flat distribution and then approaches the target distribution gradually by increasing the temperature little by little. In this way, we can alleviate the main problem of using MCMC in phylogenetics, i.e. inefficient exploration in the tree space because only small moves are allowed. In the example of this chapter, we have used the MCMC moves that are designed for standard MCMC algorithms in phylogenetics. In future, it is desirable to design more bold MCMC moves that are more suitable for SMC samplers.

Chapter 6

A General Evolutionary Model Based on String-valued CTMC

Current tree inference methods ground their analysis on a relatively restricted range of evolutionary phenomena, limited in most cases to substitution events and, less frequently, to context independent long indels [80]. However, sequence change is caused by many other important molecular mechanisms, e.g. *slipped strand mispairing (SSM)*, a well known explanation for the evolution of repeated sequence [59, 80, 104]. Most of the molecular mechanisms have not yet been exploited as phylogenetically informative events [80], since it is generally non-trivial to extend tractable sequence evolution models beyond point mutations.

Researchers have used string-valued continuous time Markov Chain (SCTMC) to use the marginals of the process to obtain the probability of trees (and alignments) [54, 73, 106]. The computational bottleneck has prevented a broader applications based on SCTMCs. Many applications of CTMC need to compute the transition probability $P_{a,b}(T)$ of the process moving from state a to state b in a finite time interval of length $T \geq 0$. Calculating transition probabilities for a CTMC can be regarded as a building block of the likelihood computation of a full phylogenetic tree. [19] proposed an efficient method to evaluate the finite-time transition probabilities for general birth-death processes (BDPs) which is a special class of CTMC. However, currently there is no robust and efficient method to evaluate transition probabilities for a more general CTMC, e.g. SCTMCs.

We propose a novel way to estimate the transition probabilities for a general CTMC. Our approach is based on the idea of expressing explicitly the probability of a sequence of states of a continuous-time Markov jump process (MJP) [16] and using importance sampling to sample a set of weighted sequences of states. Our method is different from the relevant work of [84] in which they sample MJP paths, completely characterized by the jump times and the sequence of states, based on a method named *uniformization* [49]. We have verified our method using numerical examples of birth-death processs (BDPs).

Further, we describe a more flexible evolutionary model based on string-valued CTMC that can incorporate a wide range of evolutionary phenomena. The Bayesian phylogenetic inference is implemented by extending the importance sampling for transition probabilities of a general CTMC to a sequential Monte Carlo (SMC) algorithm.

The rest of this chapter is organized as follows. In Section 6.1, we introduce the transition probabilities for a general CTMC and propose to compute it using importance sampling. In Section 6.2, we provide an example of the proposed importance sampling for transition probabilities using BDPs. Section 6.3 describes an evolutionary model, a general SCTMC, and how we implement Bayesian phylogenetic inference using a novel sequential Monte Carlo (SMC) algorithm. Discussions are provided in Section 6.4.

6.1 Importance Sampling for Transition Probabilities

6.1.1 Transition Probabilities

CTMCs, $\{Y_t : 0 \leq t \leq T\}$, are often used to model data sampled at discrete time points. For example, we only observe partial data, typically on the leaves, in phylogenetics. We are interested in the transition probabilities $P_{a,b}(T) = \mathbb{P}(Y_T = b | Y_0 = a)$. Let m be the number of state transitions between two states, the beginning state a and the ending state b. The total time spent in these states is T. The m state-change events occur at time points $T_0 = 0 < T_1 < \cdots < T_m < T_{m+1} = T$.

Let $Q_{x,x'}$ be the instantaneous rate of transiting from state x to state x'. Let $\lambda(x) = \sum_{x \neq x'} Q_{x,x'}$ be the instantaneous rate of departing from state x.

A path of a CTMC $\{Y_t\}$ is completely described by the sequence of time $T_{0:m+1} = \{T_0, T_1, \cdots, T_{m+1}\}$ and the sequence of states $x_{0:m} = \{x_0, x_1, \cdots, x_m\}$. Denote $t_{0:m} = \{t_0, \cdots, t_m\}$, where $t_i = T_{i+1} - T_i$, denoting the waiting time until the i-th state change.

The probability density of this path conditional on the beginning state x_0 is

$$\begin{aligned} & g(x_{0:m}, t_{0:m}|x_0, T) \\ = & \left\{\prod_{i=0}^{m-1} f(t_i;\lambda_i) J(x_i \to x_{i+1})\right\}(1 - F(t_m;\lambda_m)) \\ = & \left\{\prod_{i=0}^{m-1} \lambda_i e^{-\lambda_i t_i} \cdot \frac{Q_{x_i,x_{i+1}}}{\lambda_i}\right\} e^{-\lambda_m t_m} \\ = & \prod_{i=0}^{m-1} \frac{Q_{x_i,x_{i+1}}}{\lambda_i} \prod_{i=0}^{m-1} \lambda_i e^{-\lambda_i t_i} e^{-\lambda_m t_m} \end{aligned} \tag{6.1}$$

where $f(t;\lambda)$ is the density of an exponential distribution with rate λ; $F(t;\lambda)$ is the cdf of an exponential distribution; $J(x \to x')$ is the probability of jumping from x to x'. Here λ_i is a function that depends on the i-th state x_i; i.e. $\lambda_i = \lambda(x_i)$.

Using Monte Carlo methods, we can sample the sequence of states and the jump times to approximate transition probabilities. Alternatively, we can marginalize the time intervals when we are not interested in the jump times. That is,

$$g(x_{0:m}|x_0, T) = \prod_{i=0}^{m-1} \frac{Q_{x_i,x_{i+1}}}{\lambda_i} G(\lambda_1, \lambda_2, \cdots, \lambda_m; T) \tag{6.2}$$

where

$$\begin{aligned} & G(\lambda_0, \lambda_2, \cdots, \lambda_m; T) \\ = & \int \cdots \int_{0<t_0+t_2+\cdots+t_{m-1}<T} \prod_{i=0}^{m-1} \lambda_i e^{-\lambda_i t_i} e^{-\lambda_m (T - \sum_{i=0}^{m-1} t_i)} dt_1 dt_2 \cdots dt_{m-1}. \end{aligned}$$

The above integral is equal to the $(1, m+1)$-th element of a matrix exponential,

$e^{\tilde{Q}T}$, where $\tilde{Q}$ is

$$\tilde{Q} = \begin{bmatrix} -\lambda_0 & \lambda_0 & 0 & \cdots & 0 & 0 \\ 0 & -\lambda_1 & \lambda_1 & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & -\lambda_m & \lambda_m \\ 0 & 0 & 0 & \cdots & 0 & 0 \end{bmatrix}.$$

When the rates, $\lambda_0, \lambda_1, \cdots, \lambda_m$, are distinct,

$$G(\lambda_0, \lambda_1, \cdots, \lambda_m; T) = \left(\prod_{i=0}^{m-1} \lambda_i\right) \sum_{k=0}^{m} \frac{e^{-\lambda_k T}}{\prod_{j=0, j\neq k}^{m} (\lambda_j - \lambda_k)}.$$

When all the rates are equal, i.e. $\lambda = \lambda_0 = \lambda_1 = \cdots = \lambda_m$, the probabilities are given by the Poisson probabilities

$$G(\lambda_0, \lambda_1, \cdots, \lambda_m; T) = \frac{e^{-\lambda T}(\lambda T)^m}{m!}.$$

If some of the rates are equal, i.e. $\lambda_k = \lambda_j$ then the expression should be interpreted as a limit when $\lambda_k \to \lambda_j$. Assume that $\lambda_0, \cdots, \lambda_m$ takes m^* distinct values $\tilde{\lambda}_1, \cdots, \tilde{\lambda}_{m^*}$ with the frequencies $n_1, \cdots, n_{m^*}$, where $\sum n_i = m + 1$, then

$$G(\lambda_0, \lambda_1, \cdots, \lambda_m; T) = \left(\prod_{i=0}^{m} \lambda_i\right) \prod_{i=1}^{m^*} \frac{(-1)^{n_i - 1}}{(n_i - 1)!} \frac{\partial^{n_i - 1}}{\partial \tilde{\lambda}_i^{n_i - 1}} \sum_{k=1}^{m^*} \frac{e^{-\tilde{\lambda}_k T}}{\prod_{j\neq k} (\tilde{\lambda}_j - \tilde{\lambda}_k)}. \tag{6.3}$$

6.1.2 Importance Sampling

Since the probability density of the sequence of states and jump times, denoted $g(x_{0:m}, t_{0:m} | x_0, T)$, can be computed by (6.1) pointwise, we can use Monte Carlo methods to obtain a sample of sequences of states and jump times that start from x_0 and end with x_m. We propose to use importance sampling. Denote $q_{\text{path}}(x_{0:m}, t_{0:m})$ as the density of the proposal distribution for $x_{0:m}$ and $t_{0:m}$ satisfying $x_0 = a$, $x_m = b$, and $\sum_{i=0}^{m} t_i = T$. The importance weight for the proposed path, $\{x_{0:m}, t_{0:m}\}$, can be

computed by

$$w_{\text{path}}(x_{0:m}, t_{0:m}) = \frac{g(x_{0:m}, t_{0:m}|x_0, T)}{q_{\text{path}}(x_{0:m}, t_{0:m})}.$$

The transition probability $P_{a,b}(T)$ can be rewritten as

$$\begin{aligned}
& \mathbb{P}(Y_T = b|Y_0 = a) \\
= & \sum_{\{x_{0:m}, t_{0:m}\}: x_0 = a, x_m = b, \sum_{i=0}^{m} t_i = T} g(x_{0:m}, t_{0:m}|x_0 = a, T) \\
= & \sum_{\{x_{0:m}, t_{0:m}\}: x_0 = a, x_m = b, \sum_{i=0}^{m} t_i = T} \frac{g(x_{0:m}, t_{0:m}|x_0 = a, T)}{q_{\text{path}}(x_{0:m}, t_{0:m})} \cdot q_{\text{path}}(x_{0:m}, t_{0:m}) \\
= & \sum_{\{x_{0:m}, t_{0:m}\}: x_0 = a, x_m = b, \sum_{i=0}^{m} t_i = T} w_{\text{path}}(x_{1:m}, t_{0:m}) \cdot q_{\text{path}}(x_{0:m}, t_{0:m}).
\end{aligned}$$

Hence, a consistent estimate of $P_{a,b}(T)$ is

$$\hat{P}_{a,b}(T) = \frac{1}{K} \sum_{k=1}^{K} w_{\text{path}}(x_{0:m}^{(k)}, t_{0:m}^{(k)}),$$

where $w_{\text{path}}(x_{0:m}^{(k)})$ is the importance weight for the k-th proposed paths.

Alternatively, we can sample a sequence of states without sampling the jump times. Denote $q_{\text{state}}(x_{0:m})$ as the density of the proposal distribution for $x_{0:m}$ with the fixed ends, i.e. $x_0 = a$ and $x_m = b$. The importance weight for the proposed path, $x_{0:m}$, can be computed by

$$w_{\text{state}}(x_{0:m}) = \frac{g(x_{0:m}|x_0, T)}{q_{\text{state}}(x_{0:m})}.$$

The transition probability $P_{a,b}(T)$ can be rewritten as

$$\begin{aligned}
& \mathbb{P}(Y_T = b | Y_0 = a) \\
= & \sum_{x_{0:m}: x_0 = a, x_m = b} g(x_{0:m} | x_0 = a, T) \\
= & \sum_{x_{0:m}: x_0 = a, x_m = b} \frac{g(x_{0:m} | x_0 = a, T)}{q_{\text{state}}(x_{0:m})} \cdot q_{\text{state}}(x_{0:m}) \\
= & \sum_{x_{0:m}: x_0 = a, x_m = b} w_{\text{state}}(x_{1:m}) \cdot q_{\text{state}}(x_{0:m}).
\end{aligned}$$

Hence, a consistent estimate of $P_{a,b}(T)$ is

$$\hat{P}_{a,b}(T) = \frac{1}{K} \sum_{k=1}^{K} w_{\text{state}}(x_{0:m}^{(k)}),$$

where $w_{\text{state}}(x_{0:m}^{(k)})$ is the importance weight for the k-th proposed sequence of states.

In order to use the importance sampling method to estimate $P_{a,b}(T)$, we need to choose a proposal distribution for sampling $x_{0:m}$ that starts from $x_0 = a$ and ends with $x_m = b$. The choice of a proposal distribution depends on the particular application. We will provide more details about the proposal distribution in importance sampling for transition probabilities of CTMCs in Section 6.2 and that in the SMC algorithm for an evolutionary model in Section 6.3.

6.2 Birth-death Processes

In this section, we will use BDPs as an example of CTMC to illustrate the importance sampling method for transition probabilities. In a BDP with an instantaneous birth rate υ_n and a death rate μ_n, the rate of departing from state n is $\lambda(n) = \upsilon_n + \mu_n$. After we sample a sequence of states, $x_{0,m}$, we evaluate its probability using Equation (6.2), in which the λ_i's are computed by $\lambda_i = \lambda(x_i) = \upsilon_{x_i} + \mu_{x_i}$.

6.2.1 Sampling a Sequence of States with Fixed Two Ends

Now the task is to sample $x_{0,m}$ that starts from $x_0 = a$ and ends with $x_m = b$ for the BDP. Suppose the current state is x, the next state, denoted x', can only be $x - 1$

or $x+1$ in BDP. We build the proposal using a sequence of small proposals from the current state to its adjacent state. We use two parameters, α_{stop} and η, in our proposal distributions. If x is the final state b, the proposed sequence of states ends with probability α_{stop}, or jumps to either $x-1$ or $x+1$ with probability $(1-\alpha_{\text{stop}})/2$; if x is not the final state, it jumps to $x+1$ with probability

$$p_0 = \mathbf{1}(b > x)(1-\eta) + \mathbf{1}(b < x)\eta,$$

or to $x-1$ with probability $1-p_0$, where $0 < \eta < 0.5$, and $\mathbf{1}$ is an indicator function. In this way, the proposal avoids the possibility of being trapped in a loop of jumping back and forth between some states such that it will never end with state b.

6.2.2 Example 1

For the simple BDP with $\upsilon_n = n\upsilon_0$ and $\mu_n = n\mu_0$, we have the closed-form solution given explicitly by Bailey [4] as

$$\begin{aligned} P_{a,b}(t) &= \sum_{j=0}^{\min(a,b)} \binom{a}{j}\binom{a+b-j-1}{b-1}\alpha^{a-j}\beta^{b-j}(1-\alpha-\beta)^j \\ P_{a,0}(t) &= \alpha^a, \end{aligned}$$

where $\alpha = \frac{\mu_0(e^{(\upsilon_0-\mu_0)t}-1)}{\upsilon_0 e^{(\upsilon_0-\mu_0)t}-\mu_0}$, and $\beta = \frac{\upsilon_0(e^{(\upsilon_0-\mu_0)t}-1)}{\upsilon_0 e^{(\upsilon_0-\mu_0)t}-\mu_0}$.

Figure 6.1 shows comparison of transition probabilities $P_{10,b}(T = 1)$ computed using our method for the birth-death processes with $\upsilon_0 = 0.2$ and $\mu_0 = 0.4$. In this example, the number of sampled sequences of states is $K = 1000$; the tuning parameters in the proposal distribution are $\alpha_{\text{stop}} = 0.3$, and $\eta = 0.4$.

6.2.3 Example 2

The second example is a BDP with the linear rates $\upsilon_n = n\upsilon_0 + \nu$ and $\mu_n = n\mu_0 + \gamma$. This model can be used in a population model for the number of organisms in an area with new immigrants arrive at rate ν, and emigrants leave at rate γ [19].

Figure 6.2 shows $P_{10,b}(T)$ for several times T and state b, with the parameters

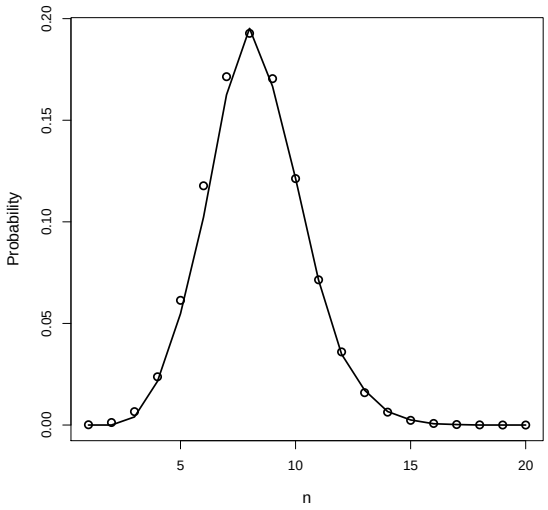

Figure 6.1: Comparison of transition probabilities $P_{10,b}(T = 1)$ computed using our importance sampling algorithm (solid line) and their true values (circles) for the birth-death processes with $\upsilon_0 = 0.2$ and $\mu_0 = 0.4$.

$\upsilon_0 = 0.5, \nu = 0.2, \mu_0 = 0.3$, and $\gamma = 0.1$. The number of sampled sequences of states is chosen to be $K = 1000$; the tuning parameters in the proposal distribution are $\alpha_{\text{stop}} = 0.3$, and $\eta = 0.4$. For the purpose of evaluating our method, we deliberately use the same setting for this example as in [19], and the results agree with each other.

6.3 A General Evolutionary Model for Phylogenetics

6.3.1 String-valued CTMC on a Phylogenetic Tree

We are interested in modelling the evolution between two sequences at the ends of one branch on a phylogenetic tree. Suppose the branch length is T. We use an SCTMC, $\{Y_t : 0 \le t \le T\}$, to model the evolution between two sequences. The individual variable Y_t is assumed to have a countable infinite domain of all pos-

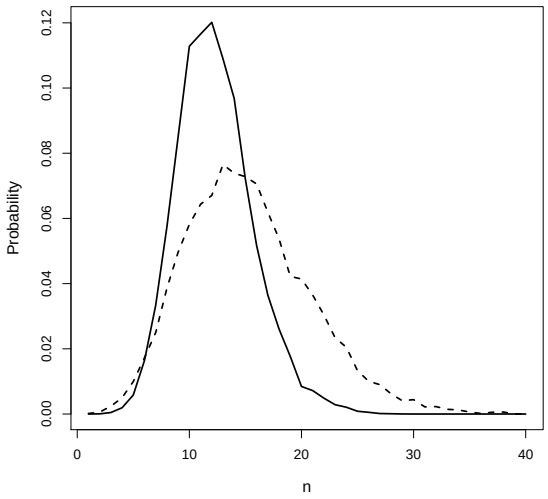

Figure 6.2: Transition probabilities $P_{10,b}(T)$ with $T = 1$ (solid) and $T = 2$ (dashed) computed using our importance sampling algorithm for the birth-death processes with $\upsilon_0 = 0.5, \nu = 0.2, \mu_0 = 0.3$, and $\gamma = 0.1$.

sible molecular sequences. Let m be the total number of mutation events between two sequences. The sequence of states is a sequence of molecular strings $x_{0:m} = \{x_0, x_1, \cdots, x_m\}$ along the branch under consideration. The state changes whenever a mutation event occurs. Hence the sequence of states is completely determined by the mutation events.

We propose a general evolutionary model that can incorporate various evolutionary phenomena. This is accomplished by defining the rate of jumping away from the current state x as a flexible function of various mutation events, e.g. the single-nucleotide indels, substitutions etc. As an example, in the following, we define the rate, $\lambda(x)$, as a function of the point mutation rate per base θ_{sub}, the global point insertion rate λ_{pt}, the point deletion rate per base μ_{pt}, the global SSM insertion rate λ_{SSM}, and the SSM deletion rate per valid SSM deletion location μ_{SSM}:

$$\lambda(x) = n\theta_{sub} + \lambda_{pt} + n\mu_{pt} + \lambda_{SSM} + k(x)\mu_{SSM}, \tag{6.4}$$

where n is the length of x, and $k(x)$ is the number of valid SSM deletion locations. The substitution model in this example is a simple Jukes-Cantor model, but our method can be easily adapted to handle more complicated ones such as the General Time Reversible (GTR) family.

The jump probability from x_k to x_{k+1} is

$$J(x \to x') = \frac{1}{\lambda(x)} \begin{cases} \theta_{sub} & \text{Point substitution} \\ \frac{\lambda_{pt}}{n+1} & \text{Point insertion} \\ \mu_{pt} & \text{Point deletion} \\ \frac{\lambda_{SSM}}{k(x)} & \text{SSM insertion} \\ \mu_{SSM} & \text{SSM deletion.} \end{cases}$$

Mutation type from x to x'

To extend the proposed SCTMC on one branch to describe the evolution on a phylogenetic tree, we first introduce some notations. Let τ denote the phylogenetic X-tree under consideration, including the tree topology and the associated branch lengths. Let $V(\tau)$ is the set of internal nodes except for the root. Let M denote the mutation events and their times on the tree, which fully determine the hidden molecular sequences. For the internal node $v \in V(\tau)$, $\rho(v)$ denotes its parent; $\tilde{x}^{(v)}$ denotes the hidden molecular sequence on v; $\beta^{(v)}$ is the length of the branch, denoted e_v, that points to v in the direction from the root.

We assume there exists a bijection between M and the hidden strings. The probability of observed sequences $\mathcal{Y}$ and the mutation events M given the phylogenetic X-tree τ is

$$\begin{aligned} p(\mathcal{Y}, M|\tau) = p(\tilde{x}^{(0)}) \quad & \prod_{v \in V(\tau)} \quad g(x^{(v)}_{0:m_v}, t^{(v)}_{0:m_v} | x^{(v)}_0 = \tilde{x}^{(\rho(v))}, \beta^{(v)}) \\ & \prod_{u \in X} \quad g(x^{(u)}_{0:m_u}, t^{(u)}_{0:m_u} | x^{(u)}_0 = \tilde{x}^{(\rho(u))}, \beta^{(u)}) \mathbf{1}(x^{(u)}_{m_u} = \mathcal{Y}(u)), \end{aligned} \tag{6.5}$$

where $\{x^{(v)}_{0:m_v}, t^{(v)}_{0:m_v}\}$ are the sequences of hidden strings and the jump times along the branch e_v. We use an improper uniform distribution over the strings on {A,C,G,T} as the distribution for the root sequence $\tilde{x}^{(0)}$.

Using Bayesian approaches, the target distribution is the posterior of τ and M

given $\mathcal{Y}$ that is proportional to

$$\gamma(\tau, M) = \gamma_{\mathcal{Y}}(\tau, M) = p(\mathcal{Y}, M|\tau)p(\tau), \tag{6.6}$$

where the likelihood model is described in Equation (6.5) and $p(\tau)$ is a prior on the tree τ.

6.3.2 Sequential Monte Carlo

We currently propose to use a simple SMC algorithm that requires fixing the evolutionary parameters. In the proposed method for our evolutionary model, the r-th partial state s_r is a forest that includes the forest topology and the associated branch lengths, denoted τ_r, as well as the mutation events and the jump times, denoted M_r; i.e. $s_r = (\tau_r, M_r)$. Consider a sequence of intermediate distributions over forests:

$$\gamma(s_r) = \prod_{(\tau_i, X_i) \in s_r} \gamma_{\mathcal{Y}(X_i)}(\tau_i, M_r(\tau_i)),$$

where $M_r(\tau_i)$ denotes the mutation events and their jump times on tree τ_i.

At the r-th iteration of SMC, the weight update for the k-th particle is

$$w_{r,k} = w(\tilde{s}_{r-1,k}, s_{r,k}) = \frac{\gamma(s_{r,k})}{\gamma(\tilde{s}_{r-1,k})} \cdot \frac{\nu^-(s_{r,k} \to \tilde{s}_{r-1,k})}{\nu^+(\tilde{s}_{r-1,k} \to s_{r,k})},$$

where ν^+ is the forward proposal distribution, and ν^- is the backward proposal distribution. For simplicity, in this chapter we only consider clock trees for which $\nu^- \equiv 1$.

The proposal $\nu^+(s \to \cdot)$ randomly selects a pair of trees in s to merge and proposes the hidden strings and jump times along the newly proposed branches. Suppose the root strings of the selected subtrees are a and b. We sample the hidden strings $x_{0:m}$ that starts from $x_0 = a$ and ends with $x_m = b$ using an epsilon greedy strategy based on Levenshtein distances to the fixed ending string. We use two parameters, α_{stop} and α_{greedy}, in our proposal distributions. For the current string x, we consider all possible next strings. The potential next string, denoted x', can be obtained by applying one mutation event to x. If x' is the final state b, the proposed sequence of states ends with probability α_{stop}; otherwise, x' is selected

probabilistically according to

$$q_M(x, x') \propto \exp(\alpha_{\text{greedy}} \cdot d(x', b)),$$

where $d(x', b)$ is the Levenshtein distance between x' and the ending string b.

6.3.3 Simulation Studies

Asymptotic String Lengths

We investigate the distribution of the asymptotic string length by varying one parameter and fixing all the other parameters. Figure 6.3 shows the mean string lengths and standard deviations using 10000 iterations of forward simulation. One interesting result is the asymptotic string lengths versus the point substitution rates. If there were no SSM mutations, we expect the string length does not change much with the point substitution rates. However, we can see that the mean string length increases with the number of point substitution rates due to the interplay between the SSM mutations and the point mutations. More specifically, SSM deletion is one large factor for evolving shorter strings, and a larger point substitution rate will reduce the positions for possible SSM deletions.

Tree Inference

We sampled 10 random trees from the coalescent on ten leaves, and along each of randomly generated tree, we simulated 5 sets of molecular sequences with different random seeds according to our evolutionary model. The parameters used are: SSM length=3, the point mutation rate per base $\theta_{sub} = 0.03$, the global point insertion rate $\lambda_{pt} = 0.05$, the point deletion rate per base $\mu_{pt} = 0.2$, the global SSM insertion rate $\lambda_{SSM} = 2.0$, and the SSM deletion rate per valid SSM deletion location $\mu_{SSM} = 2.0$. Figure 6.4 shows one subset of simulated data. The unaligned sequences on leaves are used for tree reconstruction using the SMC algorithm. In the proposal distribution, we used $\alpha_{\text{stop}} = 0.9$ and $\alpha_{\text{greedy}} = 50$. We summarized the posterior over trees using a consensus tree optimizing the posterior expected

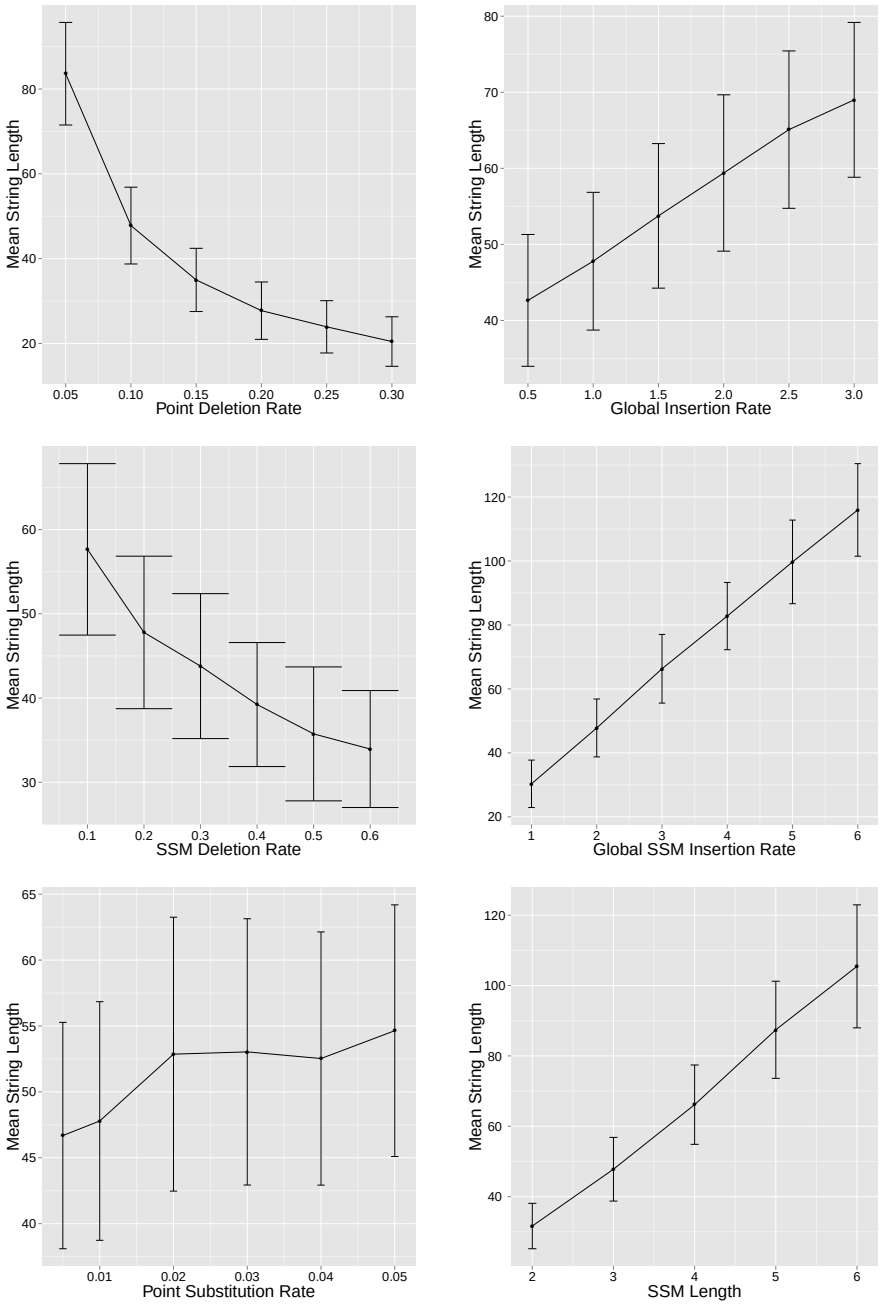

Figure 6.3: Mean string lengths and standard deviations using 10000 iterations of forward simulation. The basic setting for parameters is: SSM length=3, $\theta_{sub} = 0.01$, $\lambda_{pt} = 1.0$, $\mu_{pt} = 0.1$, $\lambda_{SSM} = 3.0$, and $\mu_{SSM} = 0.2$.

```
internal_0|CA--G---C---A--G------------TG--A---
internal_1|-GAG-C---G-G------AA----GA----TGC-TGC
internal_2|--AG-CAG--CC------CG--C-GAC---TG-----
internal_3|-GAG-C---G-G------AA----GA----TGC----
internal_4|-GAG-C---G-G------AA----GA----TGC----
internal_5|-GAG-C---G-G------AA----GA----TGC----
internal_6|-GAG-C---G-G------AA----GA----TGC----
internal_7|-TAG-C---G-C------CA--C-GAC---TGC----
internal_8|ATAG-C---G------C----A----G-C-GGCA---
leaf_0    |CA--G---C---A--G----C--A---G-TG--A---
leaf_1    |-GAG-C---G-G------AA----GT----TGC-TGC
leaf_2    |-GAG-C---G-G------AA----GA----TGC-TGC
leaf_3    |-GAG-C---G-G------AA----GA----TGC----
leaf_4    |CA--G---C---A--G------------TG--A---
leaf_5    |-GAT-C---G-G------AA----GA----TGC----
leaf_6    |--AG-CAG--CC------CG--C-GAC---TG-----
leaf_7    |-GAG-C---G-G------AA----GA----TGC----
leaf_8    |--AG-CAG--CC-GC--CCG--C-GAC---CG-----
leaf_9    |-GAT-----G-G------GA----GT-----GC----
```

Figure 6.4: A subset of simulated data.

pairwise distances. Figure 6.5 shows tree distances between generated trees and consensus trees reconstructed using our evolutionary model. The partition metric decreases with the number of particles increases, suggesting that it is possible to reconstruct fairly accurate topologies from noisy data that are generated by complex evolutionary mechanisms. Although the KF metric results also improve with more particles, a non-trivial error remains even using a large number of particles. It is probably because we are currently sampling the times of the changes using a simple proposal, the uniform order statistics distribution.

6.4 Discussion

We have presented a flexible evolutionary model using SCTMC that can incorporate a wide range of evolutionary phenomena. The flexibility comes from the perspective that the molecular evolution can be modelled as a string-valued jump process where the jumps depend only on a localized context. Specifically, the rate of jumping from the current state s to other states is a function of it, denoted $\lambda(s)$, which allows us to design complicated and realistic evolutionary models.

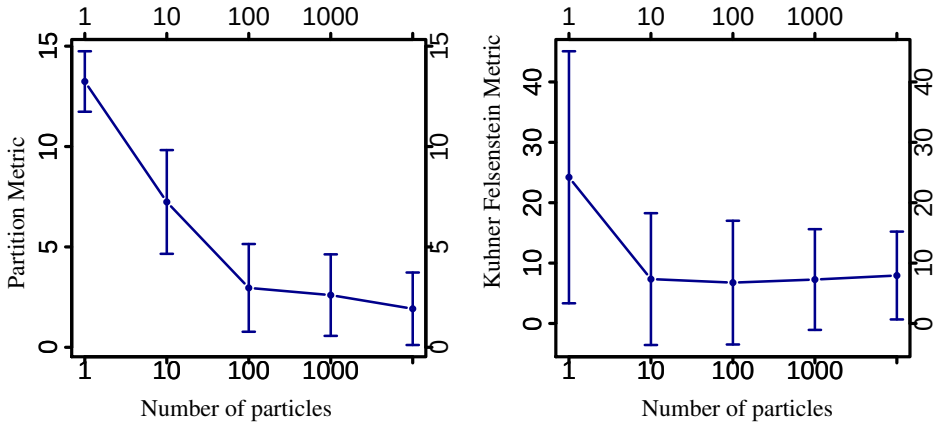

Figure 6.5: Performance on estimating trees.

The greatest challenge of the SMC method for our evolutionary model lies in designing appropriate proposal distributions with fixed starting and ending states. In the current SMC algorithm for our evolutionary model, we sample both the hidden strings and the jump times. As shown in our example, a non-trivial error in terms of the KF metric remains even using a large number of particles, implying poor proposals for the jump times. In future, we need to explore more sophisticated and efficient proposals for proposing both the states and jump times. One alternative is to integrate the jump times out using Equation (6.2). If there are identical values in $\{\lambda_i : i = 0, \cdots, m\}$, this method involves the complicated partial derivatives in Equation (6.3), which can be challenging to compute when the orders of derivatives are high.

Our ultimate goals include estimating the evolutionary parameters in our SCTMC model. Currently we simplify the problem by focusing on estimating phylogenetic trees with fixed evolutionary parameters. In future work, we can use our SMC method within each iteration of the particle Markov chain Monte Carlo (PMCMC) algorithms to jointly estimate the evolutionary parameters and the phylogenetic trees.

Chapter 7

Summary and Future Work

In this concluding chapter, we summarize our contributions in Bayesian phylogenetics and propose some research directions for future work.

7.1 Summary

We have presented various Monte Carlo methods for Bayesian Phylogenetic inference to sample from the joint posterior distribution of phylogenetic trees and evolutionary parameters.

We propose a novel and general class of SMC algorithms that is motivated by the over-counting problem when applying SMC to a non-clock phylogenetic tree. Contrary to the standard application of MCMC in Bayesian phylogenetic inference, the proposed CSMC can efficiently explore the huge phylogenetic tree space. Both theoretical and experimental results show that CSMC can obtain a consistent estimate of the expectation of a function with respect to a target measure.

We have proposed to jointly estimate phylogenetic trees and evolutionary parameters in two ways of combining MCMC and SMC. The first method is a set of PMCMC algorithms with CSMC at each iteration. At each iteration of a PMCMC algorithm, a phylogenetic tree is proposed by running CSMC. Since the main advantage of CSMC is that one can use low-dimensional proposal distributions to efficiently reconstruct high-dimensional phylogenetic trees, the proposal is more sensible such that it can lead to a bold move in the huge tree space. The second

method, an SMC sampler, provides a way to embed the standard MCMC algorithms in phylogenetics literature into an SMC algorithm. We have also described the framework for fine-grain parallelization of SMC algorithms, implemented using multiple threads on a single CPU.

The above work provides a set of flexible and efficient tools for all kinds of situations in Bayesian phylogenetic inference. First, our methods is very general that they works for both non-clock trees and restricted clock trees. Second, from our results, PMMH can achieve a better estimate for the evolutionary parameter than the standard MCMC. Third, users can fully utilize their computing resources and time by carefully specifying the number of particles at each PMCMC iteration and the number of iterations. For example, if computer memory does not allows us to run simultaneously a sufficient number of particles in CSMC, we can run PIMH for enough iterations as an alternative.

In addition to the efficient algorithms for general phylogenetic tree reconstruction, we have presented a general evolutionary model based on string-valued continuous time Markov Chain (SCTMC), incorporating a wide range of molecular mechanisms, and we have explored an SMC algorithm to do Bayesian phylogenetic inference.

7.2 Inferring Large Scale Trees

Genome-wide datasets offer opportunities for resolving difficult phylogenetic problems but also pose computational and statistical challenges for data analysis [83]. Our ultimate goal in phylogenetics is to develop new statistical evolutionary models and computational algorithms for efficient analysis of large-scale datasets. A large-scale dataset may have a large number of taxa, or a large number of sites for each sequence, or both.

When the dataset is large-scale, our proposed methods in the previous chapters might fail for two possible reasons: main memory shortages, or unrealistic running times. We propose to infer large scale trees by: 1) algorithms using divide-and-conquer strategies; 2) parallel computing.

7.2.1 Divide-and-Conquer Strategies

We propose three divide-and-conquer strategies to decompose a large scale phylogenetic inference problem. The first one, which consists in using the PIMH algorithm to replace SMC, is a general strategy to deal with a main memory shortage. The second one, the particle block Gibbs algorithm, targets datasets with a large number of taxa. The third one, an SMC algorithm using partial data, targets datasets with a relatively small number of taxa but each taxa has a long sequence.

Using the PIMH Algorithm

The PIMH algorithm can be used as one divide-and-conquer strategy to divide the task of running a large number K of particles required in an SMC algorithm into several smaller steps. More precisely, when the K particles cannot be accommodated in main memory, we can use a smaller number K' of particles in each of the R iterations using the PIMH algorithm, where K' is the maximum number of particles that fit in main memory, and $RK' \geq K$. This method is justified by the simulation studies of Section 5.1.4 in which PIMH and SMC perform similarly in terms of phylogenetic tree inference.

The Particle Block Gibbs Algorithm

We can sample sub-blocks of a phylogenetic tree t when the number of taxa, n, is so large that a prohibitive number of particles is required to lead to an efficient global update. We divide t into D large sub-blocks, $\{t_1, \cdots, t_D\}$, and use a conditional SMC algorithm for each sub-block. Denote $t_{-d} = \{t_1 \cdots, t_{d-1}, t_{d+1}, \cdots, t_D\}$ for $d \in 1, \cdots, D$. Algorithm 7.1 describes the proposed particle block Gibbs sampler for phylogenetics, in which $p(t_d|t_{-d}, \mathcal{Y})$ can be $p_\theta(t_d|t_{-d}, \mathcal{Y})$ that depends on a hyperparameter, θ. We omit θ from the notation for simplicity.

The compact pseudo-code shown in Algorithm 7.1 hides two difficulties. The first difficulty lies in how we divide t into sub-blocks. Another difficulty is that we need to adapt the conditional CSMC algorithm to make it target $p(t_d|t_{-d}, \mathcal{Y})$, which is dependent on how we define sub-blocks of a tree. A natural way is to define a sub-block as a subtree which can be resampled using our CSMC algorithm. In

Algorithm 7.1 The Particle Block Gibbs Sampler for Phylogenetics

Initialization: $i = 0$; sample $t(0)$ arbitrarily.
for $i \geq 1$ **do**
  Randomly divide $t(i-1)$ into D_i large sub-blocks. $\{t_1(i-1), \cdots, t_{D_i}(i-1)\}$
  for $d \in 1, \cdots, D_i$ **do**
    Run a conditional CSMC algorithm targeting $p(t_d|t_{-d}, \mathcal{Y})$ conditional on $t_d(i-1)$ and its ancestral lineage.
    Sample $t_d(i) \sim \hat{p}_K(\cdot|t_{-d}, \mathcal{Y})$.
  end for
end for

the following, we will describe how we partition a non-clock tree and a clock tree, respectively, into a set of subtrees.

For non-clock trees, we randomly choose a pair of leaves and identify the path between the two leaves. Note that the path between two nodes is unique on a tree. Then we randomly choose a node on the path and select one clade connected to this node as a sub-block to update.

It is more complicated to partition a clock tree than a non-clock tree because of the ultrametric constraints. We propose to cut a clock tree using a horizontal line with a random height. The block above the horizontal line is a clock subtree with pseudo-observations; and the part below this line is a forest of clock subtrees and part of their top branches. We randomly choose one of these subtrees to update. In order to maintain a clock tree after one subtree is updated, we restrict the height of an updated subtree below the horizontal line to be its height before the update. This height restriction is embedded in the proposal of the SMC algorithm in which the proposed forest height cannot exceed the original subtree height.

An SMC Algorithm Using Partial Data

Chopin [18] introduced an SMC algorithm that targets a posterior distribution of partial data and applied it to two examples, mixture models and discrete generalized linear models. Relevant previous work include [5, 87] which focuses on a similar SMC algorithm for massive datasets targeting on the posterior distribution of a smaller, more manageable portion of the whole dataset at each iteration. Their examples include a mixture of transition models that can be used to model web

traffic and robotics, and a Bayesian logistic regression model.

We propose to extend the traditional applications of this type of SMC algorithm to Bayesian phylogenetics. Recall that our target distribution in Bayesian phylogenetics is the joint posterior of a phylogenetic tree t and evolutionary parameters θ, i.e. $\pi(x) \equiv p(x|\mathcal{Y})$, where $x = (t, \theta)$. For the situation where each taxa is described by a long sequence, we can use the observed data, $\mathcal{Y}$, sequentially by considering sub-blocks of the data, $\mathcal{Y}_1, \mathcal{Y}_2, \cdots, \mathcal{Y}_R$. Denote the first r blocks of data by $\mathcal{Y}_{1:r}$. We define the r-th intermediate distribution as

$$\pi_r(x) = p(x|\mathcal{Y}_{1:r}), \tag{7.1}$$

where $\mathcal{Y}_{1:R} = \mathcal{Y}$ and $\pi_R(x) = \pi(x) = p(x|\mathcal{Y})$.

We will use the SMC sampler (Algorithm 2.5) with the backward kernel,

$$L_{r-1}(x_r, x_{r-1}) = \pi_r(x_{r-1})K_r(x_{r-1}, x_r)/\pi_r(x_r),$$

which leads to the incremental importance weight, $w(x_{r-1}, x_r) = \gamma_r(x_{r-1})/\gamma_{r-1}(x_{r-1})$. More precisely, using (7.1), we have

$$w(x_{r-1}, x_r) = \frac{p(x_{r-1})p(\mathcal{Y}_{1:r}|x_{r-1})}{p(x_{r-1})p(\mathcal{Y}_{1:r-1}|x_{r-1})} = p(\mathcal{Y}_r|x_{r-1}).$$

The proposed method is summarized in Algorithm 7.2, where the MH kernels, $\{K_r^i\}$, are described in Section 5.2.

Algorithm 7.2 An SMC Sampler using Partial Data

$x_{1,k} \leftarrow \perp$, $\forall k \in \{1, \cdots, K\}$
$w_{1,k} \leftarrow 1/K$
for $r \in 2, \ldots, R$ **do**
  Sample $x_{r,k} \sim \sum_{i=1}^{M} p_i K_r^i(x_{r-1,k}, \cdot)$, $\sum_{i=1}^{M} p_i = 1$
  $w_{r,k} \leftarrow p(\mathcal{Y}_r|x_{r-1,k})$
  Normalize weights $W_{r,k} \propto w_{r,k}$, and resample $\{x_{r,k}, W_{r,k}\}$
end for

Regarding how we define sub-blocks of the sequence data, one choice is: $\mathcal{Y}_1$ includes data from site 1 to site $[m/R]$, $\mathcal{Y}_2$ includes data from site $[m/R + 1]$ to

site $[2m/R]$, and so on, where m is the total number of sites, and the number of sub-blocks of data is equal to the number of SMC iterations, R.

7.2.2 Parallel Computing on GPUs

As discussed in Section 5.3, SMC and PMCMC algorithms are more efficient if implemented in parallel. We have demonstrated that the performance of CSMC can be improved when it is executed with multiple threads on a single CPU. Since graphics processing units (GPUs) have an extremely large number of computer cores [69, 100], it is worthwhile to explore ways to parallelize PMCMC algorithms on GPUs.

GPUs are self-contained parallel computational devices. Contrary to CPU, GPUs have an extremely large number of relatively simple computer cores, each being capable of executing fast arithmetic and logical instructions. Computer cores on GPUs are organized in blocks. Each block has a *shared* memory that is common to all the cores of this block. In addition, there is *device* memory that is accessible to all cores. Communication between CPU and GPUs is only possible via device memory.

The threads on GPUs are different from those of CPUs. Creating and destroying threads on a CPU is very expensive. In contrast, the creation of GPU threads is significantly faster than that of CPU threads. Actually, one can create several thousand threads in less than ten clock cycles [69, 100].

GPU computing is characterized by SIMD, single instruction, multiple data [69, 100]. This characteristics makes GPUs suitable for SMC algorithms in which the same set of instructions, i.e. updating and reweighting a particle, is used to update each of a large number of particles of the same structure.

It is challenging to use GPUs for reconstructing phylogenetic trees because each particle might be too large to be loaded into device memory. Therefore, inferring large scale trees on GPUs will involve a combination of the proposed divide-and-conquer strategies and super-fine grain parallelization. Another challenge of computing with GPUs is to develop skills of low-level programming for the hardware, GPUs. Two software packages, Open Multi-Processing (OpenMP) and Compute Unified Device Architecture (CUDA), can be used to parallelize computation

on GPUs.

7.3 Harnessing Non-Local Evolutionary Events for Tree Inference

In Chapter 6, we have presented our preliminary work on this topic. By non-local, we mean that sequence changes can depend on contexts of unbounded random size. More precisely, the model is based on an extension of the Doob-Gillespie construction for continuous time Markov chain (CTMC), where the exponential rates are allowed to depend on an unbounded context. In the near future, we are going to work towards the following directions:

1. Understand some important molecular mechanisms, e.g. context-sensitive substitutions, structural constraints in RNA evolution etc. We are particularly interested in *slipped strand mispairing (SSM)*, a well known explanation for the evolution of repeated sequence [59, 80, 104]. Although $\lambda(s)$ in Equation (6.4) is very flexible, biology backgrounds are required to understand how we can incorporate appropriately various molecular mechanisms in $\lambda(s)$.

2. Develop a tree sampling algorithm allowing non-local sequence evolution models. We propose to use PMCMC algorithms to approximate the posterior distribution of the evolutionary parameters, phylogenetic trees, and the hidden molecular sequences given the observed data on leaves. At each iteration, the evolutionary parameters are updated; within each iteration, we use the SMC algorithm proposed in Chapter 6 to update the hidden molecular sequences and the phylogenetic trees.

3. We will apply our method to study how the complex interplay between SSMs and point mutations affects the inferred trees. Our experiments will involve simulated sequence data with realistic patterns of SSMs. We will compare the tree reconstructions obtained with our model incorporating SSMs to other models that ignore SSMs. We will also apply our method to plant intron datasets, where SSMs play an important role in sequence change.

7.4 Joint Estimation of MSA and Phylogeny

It is desirable to jointly estimate phylogeny and alignment from their joint posterior distribution [86, 101]. Using a fixed single multiple sequence alignment (MSA) will often make the phylogenetic tree estimate biased toward the guide tree that is used to do the multiple sequence alignments [112]. In addition, a single guide tree may decrease MSA accuracy [38] . On the other hand, estimating MSA depends on the tree reconstruction. [72] proposed an MSA inference algorithm based on the result from a high performance algorithm for phylogenetic tree. Estimating MSA and phylogenetic tree can be iterated for several times. However, these systems have a lack of theoretical understanding, the difficulty of getting calibrated confidence intervals, and over-alignment problems [74, 94].

An ambitious goal is to extend our model do joint estimation of MSA, phylogeny, and evolutionary parameters. In contrast to methods based on an MSA point estimate, joint estimation avoids guide tree bias and over-confidence problems. Moreover, non-local evolutionary events, such as SSMs, are known to affect MSAs and are already used to do manual alignment. One direction in future is to develop efficient algorithms to do joint estimation of MSAs and phylogeny based on the work proposed in Section 7.3.

Bibliography

[1] M. E. Alfaro and M. T. Holder. The posterior and the prior in Bayesian phylogenetics. *Annual Review of Ecology, Evolution, and Systematics*, 37 (1):19–42, 2006. → pages 2, 31

[2] C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan. An introduction to MCMC for machine learning. *Machine Learning*, 50:5–43, 2003. ISSN 0885-6125. → pages 76

[3] C. Andrieu, A. Doucet, and R. Holenstein. Particle Markov chain Monte Carlo methods. *J. R. Statist. Soc. B*, 72(3):269–342, 2010. → pages 4, 17, 18, 61, 62, 65

[4] N. T. J. Bailey. *The elements of stochastic processes with applications to the natural sciences*. John Wiley & Sons Inc., New York, 1964. → pages 90

[5] S. Balakrishnan and D. Madigan. A one-pass sequential Monte Carlo method for Bayesian analysis of massive datasets. *Bayesian Analysis*, 1: 345–362, 2006. → pages 102

[6] O. E. Barndorff-Nielsen and N. Shephard. Non-gaussian ornsteinuhlenbeck-based models and some of their uses in financial economics. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):167–241, 2001. → pages 11

[7] M. A. Beaumont, W. Zhang, and D. J. Balding. Approximate Bayesian computation in population genetics. *Genetics*, 162:2025–2035, 2002. → pages 4

[8] J. Bergsten. A review of long-branch attraction. *Cladistics*, 21(2):163–193, 2005. → pages 27

[9] A. Bouchard-Côté, S. Sankararaman, and M. I. Jordan. Phylogenetic inference via sequential Monte Carlo. *Systematic Biology*, 2011. → pages 4, 37, 46, 48

[10] R. J. Boys, D. J. Wilkinson, and T. B. Kirkwood. Bayesian inference for a discretely observed stochastic kinetic model. *Statistics and Computing*, 18 (2):125–135, 2008. → pages 11

[11] S. T. Buckland, K. B. Newman, C. Fernández, L. Thomas, and J. Harwood. Embedding population dynamics models in inference. *Statistical Science*, 22:44–58, 2007. → pages 11

[12] N. C., H. D.G., and H. J. T-Coffee: A novel method for fast and accurate multiple sequence alignment. *Journal of Molecular Biology*, 302(1): 205–217, 2000. → pages 26

[13] J. Cannone, S. Subramanian, M. Schnare, J. Collett, L. D'Souza, Y. Du, B. Feng, N. Lin, L. Madabusi, K. Muller, N. Pande, Z. Shang, N. Yu, and R. Gutell. The comparative RNA web (CRW) site: An online database of comparative sequence and structure information for ribosomal, intron, and other RNAs. *BioMed Central Bioinformatics*, 2002. → pages 26, 55

[14] O. Cappé, E. Moulines, and T. Rydén. *Inference in Hidden Markov Models*. Springer, 2005. → pages 59

[15] O. Cappé, S. J. Godsill, and E. Moulines. An overview of existing methods and recent advances in sequential Monte Carlo. *Proceedings of the IEEE*, 95(5):899–924, May 2007. → pages 3, 37

[16] E. Çinlar. *Introduction to Stochastic Processes.* Prentice Hall., 1975. → pages 85

[17] S. Cheon and F. Liang. Phylogenetic tree construction using sequential stochastic approximation Monte Carlo. *Biosystems*, 91(1):94 – 107, 2008. → pages 26, 70, 72

[18] N. Chopin. A sequential particle filter method for static models. *Biometrika*, 89(3):539–552, August 01 2002. → pages 102

[19] F. W. Crawford and M. A. Suchard. Transition probabilities for general birth death processes with applications in ecology, genetics, and evolution. *J. Math. Biol.*, 2011. doi:doi:10.1007/s00285-011-0471-z. → pages 84, 90, 91

[20] M. P. Cummings, S. A. Handley, D. S. Myers, D. L. Reed, A. Rokas, and K. Winka. Comparing bootstrap and posterior probability values in the four-taxon case. *Systematic Biology*, 52(4):477–487, 2003. → pages 2, 31

[21] P. Del Moral. *Feynman-Kac Formulae: Genealogical and Interacting Particle Systems with Applications*. Springer, New York, 2004. → pages 13

[22] P. Del Moral, A. Doucet, and A. Jasra. Sequential Monte Carlo samplers. *Journal of The Royal Statistical Society Series B-statistical Methodology*, 68(3):411–436, 2006. → pages 4, 16, 37, 43, 46, 49, 62, 75

[23] P. Del Moral, A. Doucet, and A. Jasra. Sequential Monte Carlo for Bayesian computation. *Bayesian Statistics*, 8:1–34, 2007. → pages 4, 16, 37, 43, 49, 62, 75

[24] P. Del Moral, A. Doucet, and A. Jasra. On adaptive resampling strategies for sequential Monte Carlo methods. *Bernoulli*, 2011. To appear. → pages 59

[25] C. J. Douady, F. Delsuc, Y. Boucher, W. F. Doolittle, and E. J. P. Douzery. Comparison of Bayesian and maximum likelihood bootstrap measures of phylogenetic reliability. *Molecular biology and evolution*, 20(2):248–254, February 01 2003. → pages 2, 32, 62

[26] A. Doucet and A. M. Johansen. A tutorial on particle filtering and smoothing: fifteen years later. In *Handbook of Nonlinear Filtering (eds)*. University Press, 2009. → pages 3, 13, 37, 59

[27] A. Doucet, N. de Freitas, and N. Gordon. *Sequential Monte Carlo methods in practice*. Springer, 2001. → pages 3, 16, 37

[28] A. Drummond and A. Rambaut. Beast: Bayesian evolutionary analysis by sampling trees. *BMC Evolutionary Biology*, 7(1):214, 2007. → pages 2, 33

[29] P. Erixon, B. Svennblad, T. Britton, and B. Oxelman. Reliability of Bayesian posterior probabilities and bootstrap frequencies in phylogenetics. *Systematic Biology*, 52(5):665–673, 2003. → pages 2, 31

[30] Fearnhead, Paul, Papaspiliopoulos, Omiros, Roberts, and O. Gareth. Particle filters for partially observed diffusions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(4):755–777, Sept. 2008. → pages 11

[31] P. Fearnhead and L. Meligkotsidou. Filtering methods for mixture models. *Journal of Computational and Graphical Statistics*, 16(3):586–607, 2007. → pages 11

[32] J. Felsenstein. Cases in which parsimony or compatibility methods will be positively misleading. *Systematic zoology*, 27(4):pp. 401–410, Dec. 1978. → pages 27

[33] J. Felsenstein. Evolutionary trees from DNA sequences: a maximum likelihood approach. *J. Mol. Evol.*, 17:368–376, 1981. → pages 28, 31, 36

[34] J. Felsenstein. Confidence limits on phylogenies: An approach using the bootstrap. *Evolution*, 39(4):pp. 783–791, Jul. 1985. → pages 31

[35] J. Felsenstein. *Inferring phylogenies*. Sinauer Associates, 2003. → pages 36

[36] W. Fitch. Toward defining the course of evolution: minimal change for a specific tree topology. *Syst. Zool.*, 20:406–416, 1971. → pages 26, 27

[37] W. M. Fitch and E. Margoliash. Construction of phylogenetic trees. *Science*, 155:279–284, 1967. → pages 26

[38] R. Fleissner, D. Metzler, and A. von Haeseler. Simultaneous statistical multiple alignment and phylogeny reconstruction. *Systematic Biology*, 54 (4):548–561, August 01 2005. → pages 106

[39] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-6(6):721–741, Nov. 1984. → pages 10

[40] N. Goldman, J. P. Anderson, and A. G. Rodrigo. Likelihood-based tests of topologies in phylogenetics. *Systematic Biology*, 49(4):652–670, December 01 2000. → pages 31

[41] A. Golightly and D. J. Wilkinson. Bayesian sequential inference for nonlinear multivariate diffusions. *Statistics and Computing*, 16:323–338, 2006. → pages 11

[42] N. Gordon, D. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *Radar and Signal Processing, IEE Proceedings F*, 140(2):107–113, Apr. 1993. → pages 3, 11

[43] D. Görür and Y. W. Teh. An efficient sequential Monte Carlo algorithm for coalescent clustering. In *Advances in Neural Information Processing Systems (NIPS)*, 2009. → pages 3, 37, 59

[44] M. Goujon, H. McWilliam, W. Li, F. Valentin, S. Squizzato, J. Paern, and R. Lopez. A new bioinformatics analysis tools framework at emblebi. *Nucleic Acids Research*, 38(suppl 2):W695–W699, 2010. → pages 26

[45] R. Griffiths and S. Tavaré. Monte Carlo inference methods in population genetics. *Math. Comput. Modelling*, 23:141–158, 1996. → pages 4

[46] S. Guindon and O. Gascuel. A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood. *Systematic Biology*, 52 (5):696–704, October 01 2003. → pages 28, 31

[47] M. Hasegawa, H. Kishino, and T. Yano. Dating of the human-ape splitting by a molecular clock of mitochondrial DNA. *Journal of Molecular Evolution*, 22:160–174, 1985. → pages 29

[48] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, April 01 1970. → pages 8

[49] A. Hobolth and E. A. Stone. Simulation from endpoint-conditioned, continuous-time Markov chains on a finite state space, with applications to molecular evolution. *Annals of Applied Statistics*, 3(3):1204, 2009. → pages 85

[50] S. Höhna and A. J. Drummond. Guided tree topology proposals for Bayesian phylogenetic inference. *Systematic Biology*, 61(1):1–11, January 01 2012. → pages 3, 33, 34

[51] S. Höhna, M. Defoin-Platel, and A. Drummond. Clock-constrained tree proposal operators in Bayesian phylogenetic inference. In *BioInformatics and BioEngineering, 2008. BIBE 2008. 8th IEEE International Conference on*, pages 1 –7, oct. 2008. → pages 2, 34

[52] M. Holder and P. Lewis. Phylogeny estimation: Traditional and Bayesian approaches. *Nat. Rev.: Genet.*, 4:275–284, 2003. → pages 2, 26, 27, 33, 62, 76

[53] R. Holenstein. *Particle Markov Chain Monte Carlo*. PhD thesis, University of British Columbia, 2009. → pages 4, 18, 61, 62, 65

[54] I. Holmes and W. J. Bruno. Evolutionary hmms: a Bayesian approach to multiple alignment. *Bioinformatics*, 17(9):803–820, September 01 2001. → pages 31, 84

[55] J. P. Huelsenbeck and F. Ronquist. MRBAYES: Bayesian inference of phylogenetic trees. *Bioinformatics*, 17(8):754–755, 2001. → pages 2, 33, 55

[56] K. Hukushima and K. Nemoto. Exchange Monte Carlo method and application to spin glass simulations. *Journal of the Physical Society of Japan*, 65(6):1604–1608, 1996. → pages 80

[57] M. D. Iorio and R. C. Griffiths. Importance sampling on coalescent histories. *Adv. Appl. Prob.*, 36:417–433, 2004. → pages 4

[58] H. Jow, C. Hudelot, M. Rattray, and P. G. Higgs. Bayesian phylogenetics using an RNA substitution model applied to early mammalian evolution. *Molecular Biology and Evolution*, 19(9):1591–1601, 2002. → pages ix, 34, 35

[59] S. A. Kelchner. The evolution of non-coding chloroplast DNA and its application in plant systematics. *Annals of the Missouri Botanical Garden*, 87(4):pp. 482–498, 2000. → pages 84, 105

[60] M. Kimura. A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *Journal of Molecular Evolution*, 16:111–120, 1980. → pages 29

[61] J. Kingman. On the genealogy of large populations. *Journal of Applied probability*, 19:27–43, 1982. → pages 51

[62] H. Kishino, T. Miyata, and M. Hasegawa. Maximum likelihood inference of protein phylogeny and the origin of chloroplasts. *J. Mol. Evol.*, 31: 151–160, 1990. → pages 28

[63] G. Kitagawa. Monte Carlo filter and smoother for non-gaussian nonlinear state space models. *J. Comput. Graph. Stat.*, 5:1–25, 1996. → pages 59

[64] T. D. Kocher, J. A. Conroy, K. R. McKaye, J. R. Stauffer, and S. F. Lockwood. Evolution of nadh dehydrogenase subunit 2 in east african cichlid fish. *Molecular phylogenetics and evolution*, 4(4):420–432, 12 1995. → pages 26, 70, 72

[65] M. K. Kuhner and J. Felsenstein. A simulation comparison of phylogeny algorithms under equal and unequal evolutionary rates. *Molecular Biology and Evolution*, 11(3):459–468, 1994. → pages 36

[66] C. Lakner, P. van der Mark, J. P. Huelsenbeck, B. Larget, and F. Ronquist. Efficiency of Markov chain Monte Carlo tree proposals in Bayesian phylogenetics. *Systematic Biology*, 57(1):86–103, 2008. → pages 2, 33, 34, 35, 62, 64, 76

[67] B. Larget and D. Simon. Markov chain Monte Carlo algorithms for the Bayesian analysis of phylogenetic trees. *Mol. Biol. Evol.*, 16:750–759, 1999. → pages 2, 28, 33, 34, 35, 54, 62, 76

[68] M. Larkin, G. Blackshields, N. Brown, R. Chenna, P. McGettigan, H. McWilliam, F. Valentin, I. Wallace, A. Wilm, R. Lopez, J. Thompson, T. Gibson, and D. Higgins. Clustal w and clustal x version 2.0. *Bioinformatics*, 23(21):2947–2948, 2007. → pages 26

[69] A. Lee, C. Yau, M. B. Giles, A. Doucet, and C. C. Holmes. On the utility of graphics cards to perform massively parallel simulation of advanced Monte Carlo methods. *Journal of Computational and Graphical Statistics*, 19(4):769–789, 2010. → pages 79, 80, 104

[70] S. Li, D. Pearl, and H. Doss. Phylogenetic tree construction using Markov chain Monte Carlo. *J. Am. Stat. Assoc.*, 95:493–508, 2000. → pages 2, 28, 29, 33, 62, 76

[71] J. S. Liu. *Monte Carlo Strategies in Scientific Computing*. Springer, 2001. → pages 3, 37

[72] K. Liu, S. Raghavan, S. Nelesen, C. R. Linder, and T. Warnow. Rapid and accurate large-scale coestimation of sequence alignments and phylogenetic trees. *Science*, 324(5934):1561–1564, June 19 2009. → pages 106

[73] Lunter, I. Miklós, A. Drummond, J. L. Jensen, and J. Hein. Bayesian coestimation of phylogeny and sequence alignment. *BMC Bioinformatics*, 6:83–83, 2005. → pages 31, 84

[74] G. Lunter, A. Drummond, I. Miklós, and J. Hein. *Statistical aligment: recent progress, new applications and challenges*. Springer, New York, 2004// 2004. → pages 106

[75] D. Maddison. The discovery and importance of multiple islands of most parsimonious trees. *Syst. Zool.*, 40:315–328, 1991. → pages 26

[76] P. Marjoram, J. Molitor, V. Plagnol, and S. Tavaré. Markov chain Monte Carlo without likelihoods. *Proc. Nat. Acad. Sci.*, 100:15324–15328, 2002. → pages 4

[77] B. Mau, M. Newton, and B. Larget. Bayesian phylogenetic inference via Markov chain Monte Carlo. *Biometrics*, 55:1–12, 1999. → pages 2, 28, 33

[78] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953. → pages 8

[79] I. Miklós, G. A. Lunter, and I. Holmes. A long indel model for evolutionary sequence alignment. *Molecular biology and evolution*, 21(3): 529–540, March 01 2004. → pages 31

[80] D. A. Morrison. Why would phylogeneticists ignore computerized sequence alignment? *Systematic Biology*, 58(1):150–158, February 01 2009. → pages 84, 105

[81] J. S. Paul, M. Steinrücken, and Y. S. Song. An accurate sequentially Markov conditional sampling distribution for the coalescent with recombination. *Genetics*, 187:11151128, 2011. → pages 4

[82] B. Rannala and Z. Yang. Probability distribution of molecular evolutionary trees: a new method of phylogenetic inference. *J. Mol. E*, 43:304–311, 1996. → pages 2, 28, 33

[83] B. Rannala and Z. Yang. Phylogenetic inference using whole genomes. *Annual review of genomics and human genetics*, 9(1):217–231, 2008. → pages 100

[84] V. Rao and Y. W. Teh. Fast MCMC sampling for Markov jump processes and continuous time Bayesian networks. In *Proceedings of the Twenty-Seventh Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-11)*, pages 619–626, Corvallis, Oregon, 2011. AUAI Press. → pages 85

[85] A. Rechnitzer and E. J. J. van Rensburg. Generalized atmospheric rosenbluth methods (garm). *J. Phys. A: Math. Theor.*, 41(44):442002, 2008. → pages 60

[86] B. D. Redelings and M. A. Suchard. Joint Bayesian estimation of alignment and phylogeny. *Systematic Biology*, 54(3):401–418, June 01 2005. → pages 106

[87] G. Ridgeway and D. Madigan. A sequential Monte Carlo method for Bayesian analysis of massive datasets, 2003. → pages 102

[88] C. P. Robert and G. Casella. *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005. ISBN 0387212396. → pages 6, 7, 8

[89] D. Robinson and L. Foulds. Comparison of phylogenetic trees. *Mathematical Biosciences*, 53:131 – 147, 1981. → pages 36

[90] F. Ronquist and J. P. Huelsenbeck. Mrbayes 3: Bayesian phylogenetic inference under mixed models. *Bioinformatics*, 19(12):1572–1574, August 12 2003. → pages 2, 33

[91] F. Ronquist, M. Teslenko, P. van der Mark, D. L. Ayres, A. Darling, S. Hhna, B. Larget, L. Liu, M. A. Suchard, and J. P. Huelsenbeck. Mrbayes 3.2: Efficient Bayesian phylogenetic inference and model choice across a large model space. *Systematic Biology*, February 22 2012. → pages 2, 33

[92] N. Saitou and M. Nei. The neighbor-joining method: A new method for reconstructing phylogenetic trees. *Molecular Biology Evolution*, 4: 406–425, 1987. → pages 26, 27

[93] D. Sankoff. Minimal mutation trees of sequences. *SIAM Journal on Applied Mathematics*, 28(1):pp. 35–42, Jan. 1975. → pages 26, 27

[94] A. S. Schwartz and L. Pachter. Multiple alignment by sequence annealing. *Bioinformatics*, 23(2):e24–e29, January 15 2007. → pages 106

[95] C. Semple and M. Steel. *Phylogenetics*. Oxford, 2003. → pages 32

[96] M. P. Simmons, K. M. Pickett, and M. Miya. How meaningful are Bayesian support values? *Molecular Biology and Evolution*, 21(1): 188–199, 2004. → pages 2, 31

[97] D. Simon and B. Larget. Bayesian analysis in molecular biology and evolution (BAMBE), 2000. version 2.03 beta. → pages 2, 33

[98] R. R. Sokal and C. D. Michener. A statistical method for evaluating systematic relationships. *University of Kansas Science Bulletin*, 38: 1409–1438, 1958. → pages 26

[99] R. P. Stanley. *Enumerative Combinatorics. Volume I.* Cambridge University Press, Cambridge, 1986. → pages 45

[100] M. Suchard, Q. Wang, C. Chan, J. Frelinger, A. Cron, and M. West. Understanding GPU programming for statistical computation: Studies in massively parallel massive mixtures. *Journal of Computational and Graphical Statistics*, 19(2):419438, 2010. → pages 78, 79, 104

[101] M. A. Suchard and B. D. Redelings. Bali-phy: simultaneous Bayesian inference of alignment and phylogeny. *Bioinformatics*, 22(16):2047–2048, 2006. → pages 106

[102] Y. Suzuki, G. V. Glazko, and M. Nei. Overcredibility of molecular phylogenies obtained by Bayesian phylogenetics. *Proceedings of the National Academy of Sciences*, 99(25):16138–16143, December 10 2002. → pages 2, 31

[103] R. H. Swendsen and J.-S. Wang. Replica Monte Carlo simulation of spin-glasses. *Phys. Rev. Lett.*, 57:2607–2609, Nov 1986. → pages 79, 83

[104] J. S. Taylor and F. Breden. Slipped-strand mispairing at noncontiguous repeats in poecilia reticulata: A model for minisatellite birth. *Genetics*, 155 (3):1313–1320, July 01 2000. → pages 84, 105

[105] Y. W. Teh, H. Daumé III, and D. M. Roy. Bayesian agglomerative clustering with coalescents. In *Advances in Neural Information Processing Systems (NIPS)*, 2008. → pages 3, 37, 59

[106] J. Thorne, H. Kishino, and J. Felsenstein. An evolutionary model for maximum likelihood alignment of DNA sequences, 1991. → pages 30, 31, 84

[107] J. L. Thorne, H. Kishino, and J. Felsenstein. Inching toward reality: an improved likelihood model of sequence evolution. *Journal of molecular evolution*, 34(1):3–16, 1992. → pages 31

[108] L. Tierney. Markov chains for exploring posterior distributions. *Annals of Statistics*, 22:1701–1762, 1994. → pages 76

[109] J. A. Tom, J. S. Sinsheimer, and M. A. Suchard. Reuse, recycle, reweigh: Combating influenza through efficient sequential Bayesian computation for massive data. *Ann. Appl. Stat.*, 4:17221748, 2010. → pages 4

[110] E. J. J. van Rensburg. Monte Carlo methods for the self-avoiding walk. *J. Phys. A: Math. Theor.*, 42:323001, 2009. → pages 60

[111] E. J. J. van Rensburg and A. Rechnitzer. Generalized atmospheric sampling of self-avoiding walks. *J. Phys. A: Math. Theor.*, 42(33):335001, 2009. → pages 60

[112] K. M. Wong, M. A. Suchard, and J. P. Huelsenbeck. Alignment uncertainty and genomic analysis. *Science*, 319(5862):473–476, January 25 2008. → pages 106

[113] Z. Yang and B. Rannala. Bayesian phylogenetic inference using DNA sequences: a Markov chain Monte Carlo method. *Mol. Biol. E*, 14: 717–724, 1997. → pages 2, 28, 33

Appendix A

Proof of Consistency

To prove Proposition 7, we start by introducing a series of lemmas. In the following, we will use the following objects: $\phi : \mathcal{S} \to \mathbb{R}$, $\lambda : \mathcal{F}_{\mathcal{S}} \to [0, \infty)$.

Lemma 8. *For all positive measure λ with $\|\lambda\| < \infty$, we have:*

$$\mathbb{E}[(\mathrm{prop}_K\, \lambda)\phi] = (\mathrm{prop}\, \lambda)\phi,$$

where:

$$(\mathrm{prop}\, \lambda)\phi = \int \lambda(\mathrm{d}x) \int \nu^+(x, \mathrm{d}y) w(x, y) \phi(y).$$

Proof. By definition and linearity:

$$\begin{aligned}
\mathbb{E}[(\mathrm{prop}_K\, \lambda)(A)] &= \|\lambda\| \mathbb{E}[w(S_k, S'_k); S'_k \in A] \\
&= \|\lambda\| \int \bar{\lambda}(\mathrm{d}x) \int_A \nu^+(x, \mathrm{d}y) w(x, y) \\
&= (\mathrm{prop}\, \lambda)(A),
\end{aligned}$$

for all $A \in \mathcal{F}_{\mathcal{S}}$. □

Lemma 9. *For all positive measure λ with $\|\lambda\| < \infty$, we have:*

$$(\mathrm{prop}_K\, \lambda)\phi \xrightarrow{\mathbf{L}^2} (\mathrm{prop}\, \lambda)\phi,$$

with the following rate:

$$\mathbb{E}\left[(\mathrm{prop}_K\,\lambda)\phi - (\mathrm{prop}\,\lambda)\phi\right]^2 \leq \frac{(C_1C_2)^2\|\lambda\|^2}{K}.$$

Proof. Using independence of the S_k in the definition of prop_K, we have:

$$\begin{aligned}\mathbb{E}\left[(\mathrm{prop}_K\,\lambda)\phi - (\mathrm{prop}\,\lambda)\phi\right]^2 &= \frac{\|\lambda\|^2}{K}\mathbb{E}[w(S_k, S_k')\phi(S_k')]^2 \\ &\leq \frac{(C_1C_2)^2\|\lambda\|^2}{K}.\end{aligned}$$

□

Corollary 10. *We have:*

$$\mathbb{E}\left[\gamma_{r,K}\phi - (\mathrm{prop}\,\gamma_{r-1,K})\phi\right]^2 \to 0.$$

Proof. Since $\|\gamma_{r,K}\| \leq C^R < \infty$,

$$\begin{aligned}\mathbb{E}\left[\gamma_{r,K}\phi - (\mathrm{prop}\,\gamma_{r-1,K})\phi\right]^2 &= \mathbb{E}\left\{\mathbb{E}\left[(\gamma_{r,K}\phi - (\mathrm{prop}\,\gamma_{r-1,K})\phi)^2\,\middle|\,\gamma_{r-1,K}\right]\right\} \\ &\leq \mathbb{E}\left[\frac{\|\gamma_{r-1,K}\|^2C^4}{K}\right] \\ &\to 0.\end{aligned}$$

□

Lemma 11. *For all r,* $\mathrm{prop}\,\gamma_r = \gamma_{r+1}$.

Proof. First note that for all simple measurable $f : \mathcal{S}^2 \to \mathbb{R}$,

$$\int\int \gamma(\mathrm{d}x)\nu^+(x, \mathrm{d}y)f(x,y) = \int\int \tau^+(\mathrm{d}x, \mathrm{d}y)f(x,y),$$

so the same identity holds for all bounded measurable f by the dominated conver-

gence theorem. Similarly,

$$\int\int \gamma(\mathrm{d}y)\nu^-(y,\mathrm{d}x)f(x,y) = \int\int \tau^-(\mathrm{d}x,\mathrm{d}y)f(x,y).$$

Using these identities and basic properties of the Radon-Nikodym derivative $w = \mathrm{d}\tau^-/\mathrm{d}\tau^+$:

$$\begin{aligned}
(\mathrm{prop}\,\gamma_r)\phi &= \int \mathbf{1}[\rho(x)=r]\gamma(\mathrm{d}x)\int \nu^+(x,\mathrm{d}y)w(x,y)\phi(y)\\
&= \int\int \mathbf{1}[\rho(y)=r+1]\gamma(\mathrm{d}x)\nu^+(x,\mathrm{d}y)w(x,y)\phi(y)\\
&= \int\int \mathbf{1}[\rho(y)=r+1]\tau^+(\mathrm{d}x,\mathrm{d}y)w(x,y)\phi(y)\\
&= \int\int \mathbf{1}[\rho(y)=r+1]\tau^-(\mathrm{d}x,\mathrm{d}y)\phi(y)\\
&= \int\int \mathbf{1}[\rho(y)=r+1]\gamma(\mathrm{d}y)\nu^-(y,\mathrm{d}x)\phi(y)\\
&= \int \mathbf{1}[\rho(y)=r+1]\gamma(\mathrm{d}y)\phi(y)\int \nu^-(y,\mathrm{d}x)\\
&= \gamma_{r+1}\phi.
\end{aligned}$$

Here to change the indicator $\mathbf{1}[\rho(x)=r]$ into $\mathbf{1}[\rho(y)=r+1]$, we have used the definition of the poset and the fact that its Hasse diagram is connected. □

Lemma 12. *If for all measurable $\phi < C$,*

$$\gamma_{r,K}\phi \xrightarrow{\mathrm{L}^2} \gamma_r\phi, \tag{A.1}$$

then for all $\phi' < C$, we also have:

$$(\mathrm{prop}\,\gamma_{r,K})\phi' \xrightarrow{\mathrm{L}^2} (\mathrm{prop}\,\gamma_r)\phi'. \tag{A.2}$$

Moreover, by Lemma 11 the RHS of Equation (A.2) is equal to $\gamma_{r+1}\phi'$.

Proof. It is enough to show the result with $\phi'(x) = \mathbf{1}[x \in A]$. Let

$$\phi''(x) = \int_A \nu^+(x, \mathrm{d}y) w(x, y).$$

Since $\phi'' < C$, we can use $\phi = \phi''$ in Equation (A.1) to get Equation (A.2). □

We can now prove the main proposition:

Proof. We proceed by induction, showing for $r \geq 0$ that the following equation holds:

$$\gamma_{r,K}\phi \xrightarrow{\mathrm{L}^2} \gamma_r \phi.$$

The base case is trivial, since $\gamma_{0,K}$ and γ_r are equal to a Dirac delta on the same atom.

To prove the induction hypothesis, we first decompose the L2 distances using Minkowski inequality, and control each term separately:

$$\begin{aligned}\mathbb{E}^{1/2}\left[\gamma_{r+1,K}\phi - \gamma_{r+1}\phi\right]^2 \leq & \mathbb{E}^{1/2}\left[\gamma_{r+1,K}\phi - (\mathrm{prop}\,\gamma_{r,K})\phi\right]^2 + \\ & \mathbb{E}^{1/2}\left[(\mathrm{prop}\,\gamma_{r,K})\phi - \gamma_{r+1}\phi\right]^2.\end{aligned}$$

But by Corollary 10, the first term goes to zero; and by Lemma 12 and the induction hypothesis, the second term also goes to zero. □