Ming-Hui Chen, Lynn Kuo, and Paul O. Lewis

Bayesian Phylogenetics: Methods, Algorithms, and Applications

List of Figures

List of Tables

Contents

1

SMC (sequential Monte Carlo) for Bayesian phylogenetics

Alexandre Bouchard-Côté
Department of Statistics, University of British Columbia, Vancouver, British Columbia, Canada

CONTENTS

Until recently, Markov chain Monte Carlo (MCMC) has been the lone, faithful workhorse of Bayesian phylogenetics. MCMC methods have turned an abstract decision theoretic framework into a practical toolbox that has been applied to most phylogenetic inferential questions.

The theory behind MCMC is now a mature field offering a flexible suite of methods to approach approximate posterior computations. However, as we will discuss shortly, there are strong motivations for developing computational methods that can complement MCMC. The goal of this chapter is to give an accessible introduction to one promising complementary approach, *Sequential Monte Carlo* (SMC, also known as *particle filter methods*).

Before delving into the theory and practice of SMC methods, we start this

chapter with an overview of two motivations for applying SMC to Bayesian phylogenetic inference.

The first motivation is the growing gap between the amount of phylogenetic data and the computational resources available to analyze these data. It is not atypical to find in the literature examples of a sampler running for weeks (Hackett et al., 2008), and these examples can be viewed as symptoms of a larger problem. Simply put, advances in processor speed no longer seem able to catch up with the advances in sequencing technologies (Mardis, 2011; Wetterstrand, 2012). The phylogenetic data storm is coming from many fronts: more sequenced species (Valentini et al., 2009), more data per species (Wapinski et al., 2007), and measurements of variations at the population level (Li et al., 2008; Shah et al., 2009). The main response to this deluge has been to exploit advances in parallelization, either at the level of processing units (Altekar et al., 2004; Suchard and Rambaut, 2009) or clusters (Feng et al., 2003). However the intrinsically serial nature of MCMC makes it nontrivial to adapt it to parallel architectures. In contrast, we will argue that SMC is an architecture able to easily tap into these new computational resources.

The second motivation for developing MCMC complements is the growing gap between the resolution at which we understand molecular evolution, and the models we use on a day-to-day basis to perform phylogenetic inference. There are many pieces of information that are currently excluded from phylogenetic analyses, not from a lack of reasonable models, but from a lack of practical computational tools. Morrison (2009) discusses the example of *slipped strand mispairings* (SSMs). The repeat patterns left by SSMs form a cue commonly used when manually preprocessing alignments. However this information is often discarded from subsequent phylogenetic analyses. Another example comes from RNA or protein structural constraints, which are known to affect evolutionary inference (Nasrallah et al., 2011). Not all of these cues will necessarily improve reconstruction accuracy (Pachter, 2007), but it is unlikely that the cues that are easy to accommodate within MCMC precisely coincide with the phylogenetically informative ones.

In this chapter, we will focus on the first motivation, the computational gains brought by SMC methods. By using familiar models in our examples, we can keep the exposition easier to follow. We will return to the second motivation in the last section of this chapter, where we discuss future directions.

There is a large and healthy literature on SMC methods, and excellent books, reviews and tutorials have been written on the subject (Doucet et al., 2001; Kotecha et al., 2003; Doucet and Johansen, 2009). However most of this previous work has focussed on either the special case of state space models (Friedland, 2005) (models with the structure of a hidden Markov model), or on general setups which are non-trivial to apply to phylogenetics (Moral et al., 2006). Recent work has started filling this gap, and the main goal of this chapter is to summarize these results (Teh et al., 2008; Görür and Teh, 2008; Bouchard-Côté et al., 2012; Wang, 2012).

This chapter is organized into three parts. In the first part, we describe

SMC from the user's perspective, focussing on how to use the output of SMC for Bayesian phylogenetic analyses. In the second and third parts, we describe the core algorithm behind SMC and extensions of SMC, respectively.

1.1 Using phylogenetic SMC samplers

1.1.1 Overview

In its most general form, SMC methods cover a wide range of algorithms: at one end of the spectrum, we end up with a close cousin of MCMC, and at the other end, with an algorithm with very different computational properties. In most of this chapter, we will focus on the latter category to give a concrete example of a different approach to phylogenetic posterior inference. We will come back to less "radical" SMC flavors in Section 1.3.5.

The most striking difference between the SMC methods discussed here and MCMC lies in the way they represent hypotheses over phylogenies. At every intermediate iteration of an MCMC chain, a complete state (fully resolved tree over the set of taxa under study X) is kept in memory. By fully resolved, we mean that all the taxa under study are connected through an hypothesized phylogenetic tree. In contrast, the SMC methods discussed here will maintain a partially resolved tree in intermediate steps, or *partial state* for short.

What do we mean by partially resolved? There is again considerable flexibility here, but the most fundamental example of a partially resolved tree is a forest. Recall that a *forest* is an acyclic graph, but in contrast to a tree, it is not required to be connected. Equivalently, a forest can be described as a collection of disjoint trees. See Figure 1.1(a).

For concreteness, assume for now that inference is performed over clock trees (how to do SMC inference over non-clock trees is discussed in Section 1.2.5). Assume also that the evolutionary parameters are known—we will discuss how this can be relaxed in Section 1.3.5. In this setup, the partial states are sets of rooted trees such that the set of species $X_i \subset X$ at the leaves of each tree form a partition of X. By this, we mean that the X_i's satisfy the following two conditions: (1) the union of the X_i's is the set of species under study, X; (2) the X_i's are disjoint, i.e. $X_i \cap X_j = \emptyset$ for all $i \neq j$.

Two special cases are worth mentioning. First, the case of the completely disconnected forest, where each observed taxon forms a trivial tree with one node and zero edges. This extreme case, which we call a *trivial forest*, denoted $\perp$, will be used to initialize our SMC algorithms. The second special case consists of forests containing a single tree connecting all the observed taxa, i.e. a standard phylogeny. We call this state *complete*.

The partial state representation is close in spirit to neighbor joining, or more generally, to agglomerative clustering methods (Teh et al., 2008). As in neighbor

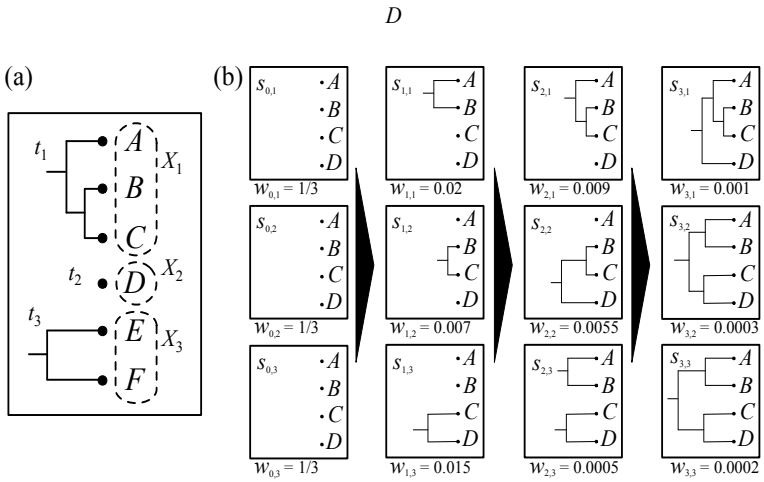

Figure 1.1: (a) An example of a partial state s, a clock forest in this case, i.e. a collection of clock trees. The set X_i is the set of leaves for tree $t_i \in s$. Note that the X_i's are disjoint, and their union is X, the set of all leaves. The tree t_2 is an example of a degenerate tree, with a topology having a single node and no edge. (b) Examples of particle populations at each generation of an SMC algorithm, with $K = 3$. Each particle is a forest $s_{n,k}$ associated with a positive weight $w_{n,k}$. Note that the weights do not sum to one within a population; they need to be normalized before creating the distribution $\hat{\pi}_{n,k}$ associated with each population. The details of how one population is produced from the previous one (denoted by black arrows in this figure), is explained in more detail in Section 1.2.

joining, SMC progresses by "merging" pairs of trees, i.e. by creating a new tree with one new root node connecting two trees from the previous iteration. However, in contrast to neighbor joining, SMC is based on a likelihood model rather than summary distance statistics. SMC is also less greedy than neighbor joining, entertaining several hypotheses simultaneously at every iteration. This reflects the fact that the goal of SMC is different than that of neighbor joining, aiming to sample from the posterior distribution rather than approximating the maximum of an objective function.[1]

Each of the parallel hypotheses is called a *particle* (see Figure 1.1(b)). A particle is composed of a forest as described above, along with a weight, which is simply a positive number reflecting the algorithm's current assessment of the quality of the corresponding hypothesis. This number does not directly reflect a probability, but we will see shortly how it can be transformed into

[1]But by being less greedy, SMC can also have an edge in maximization tasks where local optima are expected (Görür and Teh, 2008).

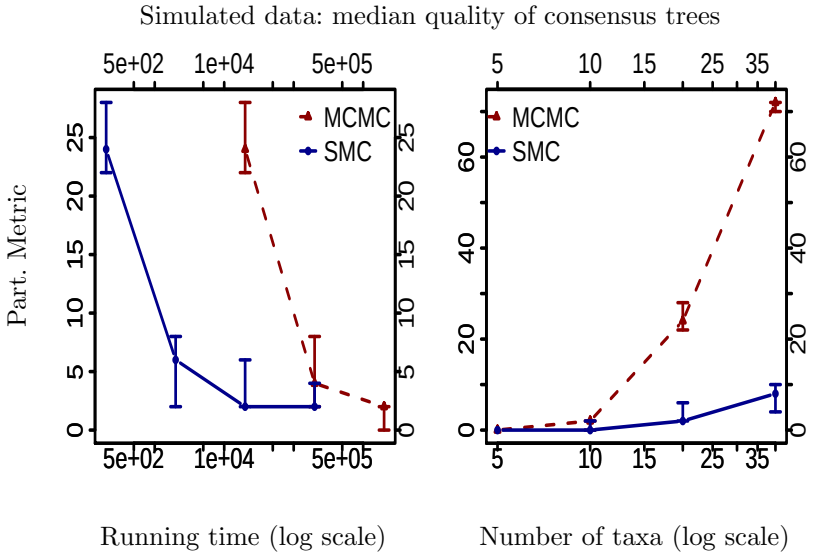

Figure 1.2: Assessment of the quality of the posterior clade estimates from Bouchard-Côté et al. (2012). Left: the rate at which the partition metric between the held-out tree used to simulate the data and the consensus tree decreases as a function of the running time. Running time is measured by the number of times the Felsenstein recursion is computed (see Section 1.3.1). For MCMC, the running time is controlled by the number of MCMC iterations, for SMC, by the number of particles K. Right: a computational budget is fixed, and the number of taxa is varied, creating increasingly difficult posterior inference problems. See Bouchard-Côté et al. (2012) for detailed experimental conditions.

a posterior probability estimate. We use the term *particle population* for a collection of competing hypotheses represented at one SMC generation. Each round consisting in selecting one merging operation for each particle is called a *generation*. Note that since we merge exactly two trees at each SMC generation, all the forests found in a given particle population have the same number of trees.

The output of a phylogenetic SMC algorithm is a particle population containing complete states. We will describe shortly how this population is computed, for now, let us look at how we can use this population to answer phylogenetic questions.

As a first example, consider the problem of assessing the support of a clade (the posterior probability that a subset of taxa $X' \subset X$ are the leaves of a

subtree of the phylogeny). To approximate this quantity from the population of complete states output by SMC, proceed as follows. First, compute the sum of the weights of the particles where the clade of interest is present; second, divide this number by the sum of all the weights. This gives a number between zero and one that approximates the posterior clade support. An important asymptotic result from SMC theory (Crisan and Doucet, 2002) is that as the number of particles K goes to infinity, this estimate converges to the posterior clade support. Since asymptotic theory does not guarantee the quality of approximations based on finite K, Bouchard-Côté et al. (2012) used simulations to assess the performance of SMC. Instead of evaluating a single clade posterior at a time, this previous work summarized all of the clades posteriors by constructing a consensus tree (Felsenstein, 2004). Refer to Figure 1.2, where two of their simulation results are shown. These results demonstrate that as the number of particles increases, the consensus tree constructed from the approximate clade posteriors becomes closer in average to the "true tree", i.e. the tree used to generate the data. The results also show that for a range of computational budgets, SMC achieves a tree reconstruction error lower than MCMC.

As a second example, let us suppose that we wish to estimate a divergence time between a pair of species $x, x' \in X$ using a simple clock model. Given a phylogenetic tree t, let us define the divergence time $d_{x,x'}(t)$ as half of the sum of the branch length separating x to x' in t. To estimate this quantity from a weighted population t_k, w_k, we start by looking at the divergence time $d_{x,x'}(t_k)$ for each particle t_k in turn, and we multiply each of these times by the corresponding weight w_k. We then divide the sum of the quantities computed in the first step by the sum of the weights, obtaining:

$$\frac{\sum_{k=1}^{K} d_{x,x'}(t_k) w_k}{\sum_{k'=1}^{K} w_{k'}}.$$

Again, this estimate will converge to the correct posterior mean as the number of particles K goes to infinity.

1.1.2 A general framework for understanding the output of Monte Carlo algorithms

Before going over the details of how the particle populations are computed, it is worth discussing in more generality how the output of SMC algorithms can be used to compute almost any expectation of interest. In particular, we will be able to put the two examples given at the end of the last section under the same umbrella.

To achieve this, we start by making the observation that the normalized weights, denoted by

$$\bar{w}_k = \frac{w_k}{\sum_{k'=1}^{K} w_{k'}}, \tag{1.1}$$

can be viewed as discrete probabilities (as they form a list of positive numbers summing to one). Each of these probabilities is assigned to a location, namely the sampled tree associated with it. This view gives us a discrete probability distribution over the space of trees, called the distribution induced by a particle population, $\hat{\pi}$. As the number of particles increases, this discrete distribution can get arbitrarily close to the posterior distribution over trees.

This suggests the following procedure for approaching generic inference problems. First, identify a function f such that the quantity of interest is expressed as an expectation under the posterior distribution, $\mathbb{E}_\pi f(t) = \mathbb{E}[f(t)|\mathcal{Y}]$, where π denotes the posterior distribution of the phylogenetic tree random variable t given the data $\mathcal{Y}$. Second, we simply replace the posterior distribution π in the above expectation by the discrete approximation $\hat{\pi}$ induced from the particle population output by SMC. This yields the estimator:

$$\mathbb{E}_\pi f(t) \approx \mathbb{E}_{\hat{\pi}} f(t) \tag{1.2}$$

$$= \sum_{k=1}^{K} \bar{w}_k f(t_k). \tag{1.3}$$

For example, in the first example, f can be taken as the function that is equal to one if the input tree has the clade of interest X', and zero otherwise. With such f, (1.3) reduces to the sum of the normalized weights corresponding to the trees consistent with the clade of interest X'.

In the second example, f can be taken as the function returning the divergence time of two species in an input phylogeny, i.e. $f = d_{x,x'}$. This also yields the weighted averaged described before.

Note that if one is interested in $d_{x,x'}$ for more than one pair of taxa x, x', only one posterior approximation $\hat{\pi}$ needs to be computed, and can be reused to compute all the required posterior expectations.

The technique described in this section is closely related to the way MCMC approximations are computed. With MCMC, the same basic equation is used, equation (1.2), but the discrete approximation $\hat{\pi}$ is formed differently. The support of $\hat{\pi}$ is taken to be the set of distinct phylogenies $\{t_k\}$ visited by MCMC, and the weight w_k, to be the number of times t_k was visited by the sampler (because of rejections, it is possible to find identical phylogenies in the MCMC output).

1.2 How phylogenetic SMC works

In this section, we elaborate on how the particles are computed. We start with some background on two related, simpler algorithms: *importance sampling* (IS), and *sequential importance sampling*(SIS). We then present SMC itself, which adds resampling on top of SIS. These three ideas, importance sampling,

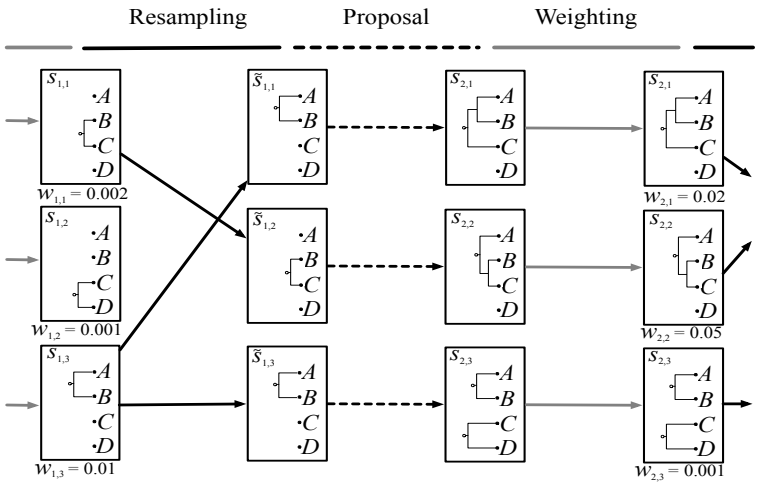

Figure 1.3: A schematic example from Bouchard-Côté et al. (2012) of the tree steps used in SMC to produce one particle population to the next population.

sequential proposals, and resampling, form the core of the SMC methodology. We then give a description of the techniques involved in implementing scalable samplers.

As in the previous section, π is the posterior distribution over trees given the data. From Bayes' theorem, this posterior can be written as a density of the form $\gamma(t)/Z$, where γ is the product of the prior density times the likelihood, and Z is the marginal probability of the observed data, $Z = \mathbb{P}(\mathcal{Y})$. Note that the function γ can be efficiently evaluated pointwise. This involves summing over the internal nucleotides while fixing the tree, an operation that can be carried efficiently using Felsenstein's peeling recursion (Felsenstein, 1981), a special case of the sum-product or junction tree algorithm (Bishop, 2006).

Computing Z, on the other hand, is intractable, as it would involve a sum over all possible tree topologies, as well as an iterated integral over all possible branch lengths.

1.2.1 The foundations: importance sampling

We start this section by describing importance sampling, an algorithm that is simpler than SMC but that also shares important similarities with its more advanced cousin. In particular, the output of an importance sampling has the same form as the output of an SMC algorithm, i.e. it is a particle population as described earlier. The main difference with SMC (and SIS) is that this population is computed in a single step. This generally creates a less reliable

approximation than the one produced by SMC, which, as we will see, proceeds in many incremental steps interspersed by resampling steps.

Before using importance sampling, the user is required to design a proposal distribution q_{imp}. This proposal distribution, defined over the space of phylogenies, should be chosen according to two criteria. The first criterion concerns the support of q: if a tree t has positive posterior density, then q_{imp} should assign positive density to t as well. Note that this condition says nothing about the relative magnitudes of the posterior and proposal. Having the regions of high probability in the posterior and proposal roughly align will decrease the number of particles needed to get to a certain level of accuracy, but is not required to prove asymptotic results. The second criterion is that it should be computationally easy to sample from q_{imp}, and to evaluate the density $q_{\text{imp}}(t)$ of this sample.

In a Bayesian context, a simple (but often inefficient) way to construct a proposal for an importance sampling algorithm is to use the prior as the proposal. This automatically satisfies the first criterion as the support of the posterior is always a subset of the support of the prior. The second criterion will also be satisfied for most prior distributions—how this is done is explained in more detail in the next section. Computing the density under the prior is also possible using the known closed-form formula for the number of topologies on $|X|$ leaves.[2]

Importance sampling creates each particle in the population independently and in two steps. In the first step, we sample from the proposal distribution with density q_{imp} to obtain a tree t_k. In the second step, we correct the discrepancy between the proposal and the posterior distribution by assigning a weight w_k to the proposed tree t_k. This is done using the following ratio:

$$w_k = \frac{\gamma(t_k)}{q_{\text{imp}}(t_k)}. \tag{1.4}$$

We now have the ingredients needed to form a particle population. As described in the first part of the chapter, this population can be used to answer a wide range of phylogenetic questions.

Note that the form of (1.4) is motivated by the law of large numbers. It ensures that the approximation $\hat{\pi}$, defined in Section 1.1.2, converges to the target posterior distribution π as the number of particles K goes to infinity (see Bishop (2006)).

1.2.2 Towards SMC: sequential importance sampling (SIS)

Sequential importance sampling (SIS), is an equivalent way of viewing importance sampling, but reparameterized in a way that naturally leads to

[2]Technically, for importance sampling the normalization of the proposal is not required. This contrast with SMC, where being able to compute the normalization of the proposal is important and often overlooked.

SMC. Instead of computing each particle in one step, as in IS, SIS builds them in stages or generations. In our phylogenetic setup, the number of stages is equal to the number of observed taxa minus one. To avoid confusion with MCMC iterations, we use the terminology of generation for these stages.

It is often natural to break the proposal into smaller steps. For example, a uniform topology is usually sampled by merging rooted trees in stages. Formally, assume that at generation $n \in 1, 2, \ldots, N$, $N = |X| - 1$, each particle is a forest of $|X| - n$ rooted trees. We denote the forest or partial state at generation n and particle index k by $s_{n,k}$. Proposing a successor of this particle $s_{n,k}$ is done by sampling one of the $\binom{n}{2}$ pairs uniformly at random, that is, by sampling a subset of size two without replacement from the set of trees in the forest.

Proposing branch lengths incrementally is also generally straightforward. Consider for example the task of sampling from phylogenies equipped with a Yule prior, and let us define the height of a forest as the height of the tallest tree in the forest. Sampling branch lengths in this case is equivalent to sampling increments δ between two forests. To mirror the definition of the Yule prior, we let this increment be distributed according to an exponential distribution with rate $|X| - n + 1$.

In order to formalize the proposal used in SIS and SMC, we first need to introduce some notation. We denote the set of partial states (introduced in Section 1.1.1) by $\mathcal{S}$, a superset of the set of trees of interest $\mathcal{T}$. For example, if $\mathcal{T}$ is the set of clock trees, one natural choice for $\mathcal{S}$ is the set of forest containing clock subtrees (we call these clock forests).

The incremental proposals q used for SIS (and SMC) are specified by a transition probability or density on $\mathcal{S}$. We denote the conditional density of proposing $s_{n,k}$ from $s_{n-1,k}$ by $q(s_{n,k}|s_{n-1,k})$. Note that proposals for SIS and SMC are different than those used in IS—the former being conditional densities, while the latter was just a density over trees.

Since we now propose states incrementally, we also need to discuss how to weight the intermediate states. In order to do this, we start by rewriting the monolithic proposal q_{imp} of importance sampling into a sequence of small conditional probabilities. This is done using the chain rule of probability:

$$\begin{aligned} q(t_k) &= q(s_{1,k}, s_{2,k}, \ldots, s_{N,k}) \\ &= q(s_{1,k}|\ \perp)\ q(s_{2,k}|s_{1,k}) \ldots q(s_{N,k}|s_{N-1,k}), \end{aligned} \tag{1.5}$$

where $\perp$ denotes a completely disconnected forest, i.e. a forest where each tree is degenerate and consists in a single observed taxon and no edge. Note that (1.5) holds because in the ultrametric case, a tree t_k uniquely correspond to a sequence of partial states $s_{1,k}, s_{2,k}, \ldots, s_{N,k}$—the correspondence is given by the sequence of speciation events in their chronological order. In the case of non-clock tree, the discussion is slightly more involved (see Section 1.2.5).

To obtain intermediate weights $w_{n,k}$, we rewrite the importance sampling weights, (1.4), into a product of N factors. We combine (1.5) with a telescoping

product to obtain:

$$\begin{aligned}
&\frac{\gamma(t_k)}{q_{\text{imp}}(t_k)} \\
&= \frac{\gamma(t_k)}{q(s_{1,k}|\perp)q(s_{2,k}|s_{1,k})\dots q(s_{N,k}|s_{N-1,k})} \\
&= \underbrace{\underbrace{\underbrace{\left(\frac{\gamma(s_{1,k})}{\gamma(\perp)q(s_{1,k}|\perp)}\right)}_{w_{2,k}}\left(\frac{\gamma(s_{2,k})}{\gamma(s_{1,k})q(s_{2,k}|s_{1,k})}\right)}_{\substack{w_{2,k}\\ \vdots}}\cdots\left(\frac{\gamma(s_{N,k})}{\gamma(s_{N-1,k})q(s_{N,k}|s_{N-1,k})}\right)}_{w_{N,k}}.
\end{aligned} \tag{1.6}$$

Note that all factors of the form $\gamma(\cdot)$ cancel out, except for the last one, which is the target $\gamma(s_{N,k}) = \gamma(t_k)$, and the first one, $\gamma(\perp)$, which is constant for all particles, and therefore disappears after normalizing the weights in (1.3). Note also that these weights can be updated incrementally as follows:

$$w_{n+1,k} = w_{n,k}\frac{\gamma(s_{n+1,k})}{\gamma(s_{n,k})\ q(s_{n+1,k}|s_{n,k})}. \tag{1.7}$$

Before going further, let us pause and look at (1.6) more closely. The careful reader may have noticed that we have not yet formally defined some of the symbols in this equation: recall that γ is a density over phylogenetic trees, while in several of the factors in (1.6), we feed γ with forests. We therefore need to generalize our definition of γ, to allow not only trees but forests as arguments.

How should this be done? We show a concrete example in Section 1.2.4, but surprisingly, the only restriction for the asymptotic correctness of the algorithm is that γ should assign a positive density to all of the proposed forests. The intuition behind why the numerical value of γ for intermediate forests does not affect asymptotic correctness is that these values all cancel each other in the telescoping product of (1.6). However, when we will add the resampling step in the next section, the choice of values γ assigned to intermediate forests will have an effect on the quality of the approximation since a finite number of particles is used in practice. But fortunately, even with resampling, the asymptotic results still hold under weak conditions.

To summarize, SIS proceeds in N generations, maintaining a population of particles at each generation. To obtain a new population from the previous one, the incremental proposal is used to propose a new particle $s_{n+1,k}$ from each previous particle $s_{n,k}$. Equation (1.7) is then used to compute a new weight for each particle.

1.2.3 Resampling

It is apparent from the derivation in (1.6) that the distribution of the final particle population in SIS is the same as in IS. In particular, both SIS and IS have serious limitations when the target distribution π is defined over a large space, for example over the space of phylogenies. The main symptom of this problem is a large weight imbalance: one gets a population where most particles in the population have a negligible contribution to expectation estimates in (1.3). This problem is called *particle degeneracy*.

In this section, we show how this problem can be alleviated by pruning unpromising particles. Remarkably, this can be done while preserving the asymptotic properties of the approximation, meaning that as the number of particles increases, the approximation can still become arbitrarily close to the true posterior.

It is important to realize that not all pruning strategies would preserve these asymptotic properties. For example, a deterministic scheme that would naively pick the k particles of highest weights and create K/k duplicates of each would not preserve asymptotic correctness.

The solution is to use a randomized scheme instead. In this section, we describe a simple-to-implement randomized scheme called multinomial resampling (alternatives do exist however, see Douc et al. (2005)). This scheme is an asymptotically correct method for addressing degeneracy.

Multinomial resampling consists in sampling K times independently from the probability distribution induced by the particle population from the previous generation, namely the discrete distribution giving probability $\bar{w}_{n,k}$ to the particle $s_{n,k}$. To get more intuition on this procedure, assume that the weights have been normalized already, and that we place the particles in an arbitrary but fixed order. Having done that, we can think of the weights as line segments covering the unit interval (see Figure 1.4 (left)). Multinomial resampling consists in throwing K darts on this one-dimensional target, and looking at how many darts fall on each segment (particle).

From this picture, we can see that particles with a high weight will generally get resampled several times, while particles with low weights will generally get pruned (i.e. get a weight of zero). By construction, this scheme also has the property that the expected number of times each particle will be resampled is proportional to the particle weight. This property is key when establishing that the resampling step preserves the asymptotic results.

After throwing the darts, the weight of each particle is modified. It is set to the number of darts falling on the corresponding segments, divided by the total number of darts K.

Note that multinomial resampling can be viewed as a generalization of the classical bootstrap procedure: if all the weights before resampling were equal to $1/K$, multinomial resampling is equivalent to resampling uniformly with replacement.

After adding a resampling step between the proposal steps of SIS, we get a

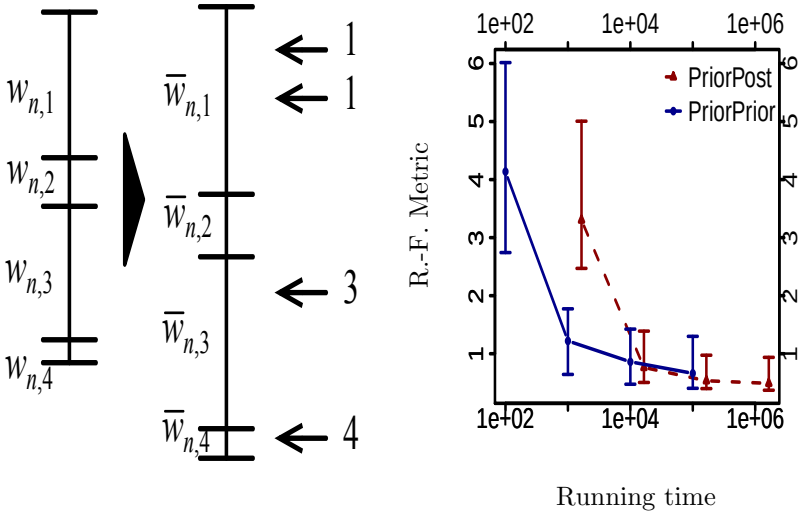

Figure 1.4: Left: schematic representation of a multinomial resampling step. After ordering arbitrarily the weights $w_{n,k}$, we can view them as segments on a stick. We then normalize these weights using (1.1) to obtain $\bar{w}_{n,k}$. Finally, we simulate K iid uniform random variables ($K = 4$ in this example) to get a new list of particles (1,1,3,4 here). Right: comparison of two SMC proposals, Prior-Prior and Prior-Post. Experimental conditions are as in Figure 1.2.

first example of an SMC algorithm. We show in Figure 1.5 that this addition has a big impact in practice. For the SIS algorithm, increasing the number of particles only improves the performance of the reconstructed tree at a very slow rate. In contrast, the improvement is substantial with SMC, thanks to the resampling step.

We summarize in Algorithm 1 all the steps involved in a simple phylogenetic SMC algorithm for ultrametric trees. In this pseudo-code, we denote the distribution assigning probability $\bar{w}_{n-1,k}$ to $s_{n-1,k}$ by $\hat{\pi}_n$. As a special case, we get that $\hat{\pi}$ as defined in Section 1.1.2 coincides with $\hat{\pi}_N$.

Note that there is an important difference between the weight equation for SIS, (1.7) and the counterpart for SMC, (1.8) in the pseudocode of Algorithm 1. In the latter equation, the weight from the previous generation is not multiplied. This is occasioned by the resampling step: after this step, each state in $\tilde{s}_{n-1,1}, \tilde{s}_{n-1,2}, \ldots, \tilde{s}_{n-1,K}$ is equally weighted.

Algorithm 1 SMC for ultrametric trees

for all $k \in \{1, \ldots, K\}$ **do**
  initialize $s_{0,k}$ to $\perp$
  initialize $w_{0,k}$ to $1/K$
end for
for generation $n = 1, 2, \ldots, N$ **do**
  for all $k \in \{1, \ldots, K\}$ **do**
    resample $\tilde{s}_{n-1,k}$ from $\hat{\pi}_{n-1}$
    propose $s_{n,k}$ from the proposal with density $q(\cdot|\tilde{s}_{n-1,k})$
    weigh:

$$w_{n,k} = \frac{\gamma(s_{n,k})}{\gamma(\tilde{s}_{n-1,k})\ q(s_{n,k}|\tilde{s}_{n-1,k})} \tag{1.8}$$

  end for
end for
return $\hat{\pi} = \hat{\pi}_N$

1.2.4 Example: Yule trees

As a concrete example, we show in this section the detailed form of the proposal, extension, and weight updates in the case of inference over Yule trees. To do this, we use a notation similar to Bouchard-Côté et al. (2012), viewing each particle as a set of trees $s = \{t_j\}$. Each tree has a set of leaves corresponding to a subset of the observations, denoted by $X(t_j) \subset X$. If s' was proposed from s, then s' is obtained from s by removing two trees, $t_l(s')$ and $t_r(s')$ and adding a new tree $t_m(s')$ formed by merging t_l and t_r. In particular, $X(t_m) = X(t_l) \cup X(t_r)$. We also use $L(t)$ to denote the likelihood of the observations corresponding to $X(t)$ given t.

As mentioned in Section 1.2.2, the form of the weight update used in Algorithm 1 assumes that each tree t over X corresponds to a unique sequence of particles $s_0, s_1, \ldots, s_N$ starting from the degenerate forest $s_0 = \perp$ and ending in tree over all the leaves $s_N = t$. The method we have discussed so far to achieve this is to ensure that the height of the forest increases at each generation. Formally, we denote by $h(s) = (h_0, h_1, \ldots, h_{n(s)})$ the list of the heights of the forests leading to s. From this, we can derive a list of increments, $\delta_i(s) = h_i - h_{i-1}$. With this notation, the height increase condition is simply that $\delta_i(s) > 0$. The set of successor forests that can be reached from s by merging a pair and increasing the height will be denoted by $S(s)$.

We can now express the extended density over forests as follows:

$$\gamma(s) \propto \left(\prod_{i=1}^{n(s)} f(\delta_i(s); |X| - i + 1) \right) \left(\prod_{t \in s} L(t) \right),$$

where $\propto$ denotes proportionality up to a constant that can depend on s only through $|s|$, and $f(x; \lambda)$ denotes the exponential density with rate λ.

As for the proposal, it can be written as

$$q(s'|s) \propto \mathbf{1}[s' \in S(s)]\ f(\delta_{n(s')}(s'); |X| - i + 1).$$

The weight is therefore equal to:

$$\begin{aligned} w_{n,k} &= \frac{\gamma(s')}{\gamma(s)q(s'|s)} = \frac{f(\delta_i(s); |X| - i + 1)\ L(t_m(s'))}{L(t_l(s'))\ L(t_r(s'))\ q(s'|s)} \\ &\propto \frac{L(t_m(s'))}{L(t_l(s'))\ L(t_r(s'))}. \end{aligned}$$

1.2.5 Inferring non-clock trees

We have assumed so far that the trees being sampled are ultrametric. We review in this section results from Wang (2012) showing how this assumption can be relaxed. For concreteness, we assume in this section a simple non-clock model where the prior distribution over topologies is uniform, and the prior distribution over branch lengths is exponential with rate λ.

To infer non-clock trees, there are two changes to make to Algorithm 1: a different proposal, and a different weighting equation.

For non-clock proposals, the simplest construction, denoted q_{NC}, is obtained by adding one degree of freedom to a clock proposal. This extra degree of freedom allows the incremental construction of rooted non-clock trees[3] More precisely, q_{NC} first picks a pair of rooted non-clock trees to merge, and then samples from two independent exponential distributions to determine the lengths of the two edges joining the pair of trees to a new internal node.[4]

As for the non-clock weight update, it consists in a generalization of the ultrametric weight formula. The new equation is given by:

$$w_{n,k} = \frac{\gamma(s_{n,k})}{\gamma(\tilde{s}_{n-1,k})}\ \frac{c(\tilde{s}_{n-1,k}|s_{n,k})}{q_{\mathrm{NC}}(s_{n,k}|\tilde{s}_{n-1,k})}, \tag{1.9}$$

where c is an *over-counting function*. The theoretical justifications and constraints on c are described in more detail in Wang (2012). A simple choice that works well in practice is to set $c(s|s')$ to the number of non-degenerate trees in the forest s', i.e. the number of trees with at least two leaves.

Note that (1.9) superficially looks like a Metropolis-Hastings ratio, but has the important difference that in general $c \neq q_{\mathrm{NC}}$.

[3] Even in models where the root is not identifiable, it is advantageous to construct rooted trees for computational reasons described in Section 1.3.1. The artificial sampled rootings can always be discarded.

[4] Except for the merging operation at the last generation N, where only one edge is added, and consequently only one branch length is sampled.

1.3 Extensions and implementation issues

After having described the core ideas behind phylogenetic SMC, we now look at various ways in which the basic algorithm can be made more efficient, and can be applied to more general setups.

1.3.1 Efficiently computing and storing the particles

The most expensive operation in phylogenetic SMC algorithms is generally the weight update, shown in (1.8) in Algorithm 1. As shown in Section 1.2.4, this amounts to evaluating the likelihood $L(t)$, for different subtrees t in a forest, which in turn involves summing over the internal nucleotides. This is done using Felsenstein's tree-peeling algorithm. Note that this computation is also the bottleneck of MCMC tree sampling algorithms, where it has to be computed for each proposed tree in order to determine the Metropolis-Hastings acceptance ratio.

Although tree-peeling is needed in both SMC and MCMC algorithms, an advantage of SMC is that we can reuse subtree recursion computations across particle generations. In contrast, even the simplest MCMC proposal, perturbing a single random branch length, requires recomputing the tree-peeling recursion for a large portion of the tree (more precisely, recomputation will be needed for all the edges along the path from the first perturbed edge to the next).

In SMC, the order in which the trees are incrementally constructed mirrors the order in which the tree-peeling recursion is computed. This means that by storing the recursion in each tree in the forest, we can avoid redundant tree-peeling computations.

The storage cost for each recursion is in the order of the number of sites times the number of characters in the alphabet. When the number of particles is large, the storage requirements can become problematic. In fact, while MCMC is time-bound, the simple phylogenetic SMC algorithm described so far is generally memory bound. Several implementation strategies can be used to alleviate this memory limitation.[5]

Additionally, it is possible to exploit the resampling step to trade memory for space. The key insight is that because of resampling, a large fraction of the particles will be pruned. It is therefore useless to keep the peeling recursions in memory for these particles.

In order to do this, we can use a two pass method: in a first pass, we compute the weight of each particle in the population, but only store the random seed used by the proposal. In other words, the forest (and peeling recursion arrays) are stored implicitly in the first pass. This is very memory

[5]First, identical subtrees will be shared by several particles (because of resampling), so particles should have pointers to recursion values rather than the values themselves. Second, when a node is not a root in any particles in the current generation, it can be safely discarded.

efficient since only two variables are needed per particle at this point. After this step, since we have the weights for all particles, the resampling step can then be performed. Finally, the actual particles are reconstructed from the random seeds, but only for the particles that survived the resampling step, and once per group of identical particles (since resampling is done with replacement, there is a positive probability of having duplicate resampled particles).

This scheme takes at most twice the time needed for the standard SMC implementation, but can significantly reduce the memory requirements. Moreover, since memory writes are slower than floating point operations in modern architectures, we have observed that in practice the two-pass scheme can actually be both faster and more memory efficient.

If further memory reductions are needed, one can compose two stages of multinomial sampling to control the number of distinct particles present in the final population. This is done by doing a first round of sampling, sampling K' particles with replacement with $K' < K$, and then doing a second round on the output of the first round, this time sampling K times (also with replacement) from these K' particles. Given a memory limit of L distinct particles, the number of particles K' in the first generation can be chosen using the formula for the expected number of distinct particles sampled from a population with weights w_k, i.e. by finding via line search the largest K' such that

$$K' - \sum_{k=1}^{K} (1 - w_{n,k})^{K'} \leq L.$$

Another option available to reduce the memory footprint is to use Particle MCMC (PMCMC) methods (Andrieu et al., 2010).

1.3.2 More SMC proposals

So far we have used the simplest possible proposal: picking a pair of trees in the forest uniformly at random, and a height increment of the forest from an exponential distribution. In this section, we review some alternative proposals that have been described in the literature.

The seminal paper by Teh et al. (2008) introduces three proposals, named "Prior-Prior," "Prior-Post," and "Post-Post." The proposal discussed so far is the equivalent, adapted to Yule trees, of the Prior-Prior proposal used in this previous work on coalescent trees. The two other proposals, Prior-Post and Post-Post, are computationally more expensive, but the hope is that these particles will be of better quality. In other words, the goal is to construct a proposal closer to the target distribution by using the information provided in the observations.

With Prior-Post, $q_{\text{pr-po}}(s'|s)$, the topology proposal is improved, keeping the branch length proposal simple and inexpensive. This is done as follows. First, an increment on the height of the forest, δ, is sampled from the same distribution as in the Prior-Prior proposal. Second, for all unordered pairs of

trees in the forest, $\{t, t'\} \subset s$, the likelihood of merging t and t' to from a new tree $t_{t,t',\delta}$ is computed. Third, a multinomial distribution over the possible pairs is sampled, with each outcome $\{t, t'\}$ having probability proportional to

$$p_{t,t'} = \frac{L(t_{t,t',\delta})}{L(t)\ L(t')}.$$

The weight update (assuming resampling at every step) is equal to

$$w_{n,k} = \sum_{\{t,t'\} \in s} p_{t,t'}. \tag{1.10}$$

See Bouchard-Côté et al. (2012) for a derivation of (1.10), as well as an empirical evaluation of this proposal, reproduced in Figure 1.4(right) for convenience. As in previous similar plots, the running time is measured by the number of times the peeling recursion was computed, with the four points in each series obtained using different numbers of particles in $\{10, 100, 1000, 10000\}$. It can be seen that in this dataset, for a given number of particles, Prior-Post slightly outperforms Prior-Prior, but since the particles of Prior-Post are more expensive to compute, Prior-Prior achieves better performances on a fixed computational budget. One possible explanation is that Prior-Post spends too much computational resources for the shallow merging operations relative to the deeper ones. It might be possible to alleviate this issue by allocating different numbers of particles at different generations.

The third proposal introduced in Teh et al. (2008), Post-Post, uses the information in the observation to inform the sampling of both the height of the new forest, and the pair to merge. At a high level, Post-Post works as follows. First, a pair to merge is sampled using a process similar to the one described above for Prior-Post, except that δ is marginalized in the weight calculation. Once a pair is picked, the height of the forest is then sampled proportionally to π. The difficulty in this scheme is the first step. The integral involved is intractable when the likelihood is a general continuous time Markov chain. However if the likelihood model is a Brownian motion (Felsenstein, 1973), the integral can be identified as the tail of a generalized inverse Gaussian integral Jorgensen (1982). Even in this case, the calculation is non-trivial to implement in practice.[6]

One benefit of Post-Post is that its proposal over height increments can have fatter tails than the exponential proposal of Prior-Prior and Prior-Post. This can be important when the branch lengths of a dataset significantly deviate from the expected branch lengths of the prior. Note that there might be simpler ways to achieve these fatter tails, as the height increment proposal in Prior-Prior is not required to mirror the prior over branch lengths, as the weights will correct the discrepancy.

[6] In the first generations of the algorithm, this integral can be shown to be equivalent to a modified Bessel function of the second kind, but in general, numerical integration computed in log-space to avoid underflows is necessary.

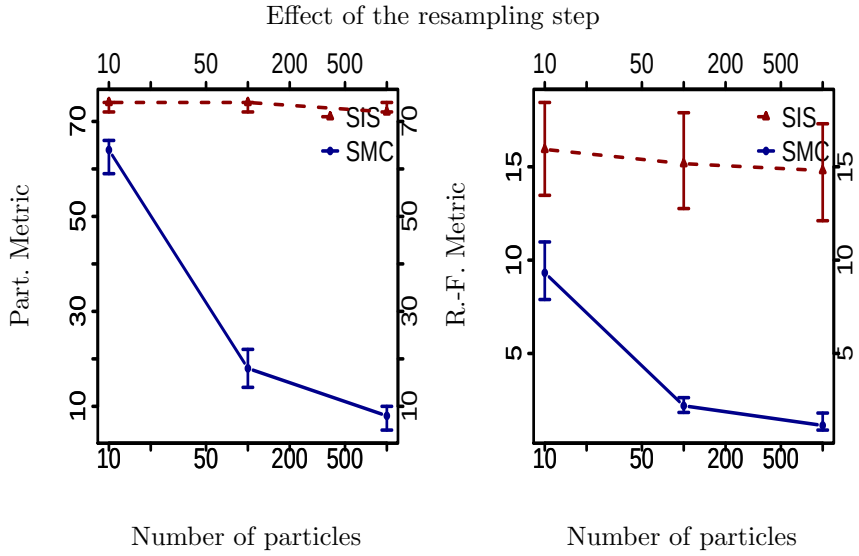

Figure 1.5: Error rates of SMC and SIS as a function of the number of particles, averaged over ten random forty taxon trees and ten executions per tree. Left: error is measured using the partition metric. Right: error is measured using the Robinson-Foulds metric. It is clear from these results that the resampling step plays a crucial role in the performance of phylogenetic SMC samplers. The experimental conditions are otherwise identical to those of Figure 1.2.

In general, there is more flexibility in designing SMC proposals than in designing MCMC proposals, in part because the reversed move, $q(s|s')$, does not need to be considered. The literature has only scratched the surface of the benefits of this flexibility. One exception is the work of Görür et al. (2012), which uses an efficient heuristic to propose pairs to merge in a more informed fashion.

1.3.3 More on resampling

In this section, we discuss the resampling step in more detail. We start by describing how to efficiently implement multinomial resampling.

When the number of particles is small to moderate, the naive scheme described in Section 1.2.3 (throwing dart on the unit length), works reasonably well. In this regime, most of the computational budget is spent in the proposal step. However, for large number of particles, the naive resampling algorithm can become the bottleneck.

This is because in the naive implementation, each of the K darts requires as much as K operations to look up which particle the dart falls into. This means that this implementation can take time in the order of K^2, while the cost of proposals grows in the order of K (but with a much larger constant).

Fortunately, multinomial resampling can also be implemented in time linear in K, by using classical methods for simulating order statistics (Ripley, 1987; Doucet, 1997; Pitt and Shephard, 1999; Carpenter et al., 1999). The basic idea is to sample $K+1$ independent exponential random variable with an equal but arbitrary rate. Since these points can be viewed as a realization of a Poisson process, their locations after normalization to $[0, 1]$ are uniformly distributed. We can then traverse the sorted list once to determine the number of particles of each type to keep in the next generation.

Beyond multinomial resampling, other schemes that preserve the asymptotic correctness of SMC exist. Examples include residual resampling, stratified resampling, systematic resampling, and dynamic resampling (see Douc et al. (2005) for a review). Some of these alternatives have been shown to theoretically improve over multinomial sampling (Douc et al., 2005).

It may also be advantageous in certain contexts to only do resampling in a subset of the generations. For example, one may choose to do no resampling between generation n and $n+1$, but to resample between generation $n+1$ and $n+2$. From an algorithmic point of view, doing this is easy to implement: skipping resampling can be reduced to the case where resampling is done at every step but with a transformed proposal. Suppose for example that we only want to skip resampling between generations n and $n+1$. We can construct a transformed proposal q'_n equal to the composition of the proposals q_n and q_{n+1} at generations n and $n+1$ in the original list of proposals. By using this transformed proposal q'_n, we can construct a new SMC algorithm that requires one fewer generation but that is equivalent to the original SMC algorithm. The advantage is that this transformed SMC scheme can be handled with the Algorithm 1.

The most widely used method for determining when to resample is based on a quantity called the *effective sample size* (ESS). ESS is typically approximated using the formula

$$\frac{1}{\sum_k (\bar{w}_{n,k})^2},$$

where $\bar{w}_{n,k}$ denotes the normalized weights. ESS is maximized when all the weights have the same value, in which case the effective sample size is equal to the number of particles K, and minimized when one particle has all the weight, an extreme case of particle degeneracy. A popular heuristic to select which subset of the generations to do resampling is to compute the ESS at each particle generation, and to resample when it falls under a threshold.

1.3.4 Estimating the marginal likelihood

In addition to computing expectation, another quantity of interest in Bayesian phylogenetic inference is the marginal likelihood, $\mathbb{P}(\mathcal{Y})$ also known as the evidence. For example, a popular way to compare two probability models $\mathbb{P}$ and $\mathbb{P}'$ in the Bayesian framework is to look at their Bayes factor, defined as the ratio of the marginal likelihood of the data under each model, $\mathbb{P}(\mathcal{Y})/\mathbb{P}'(\mathcal{Y})$.

SMC provides an easy to implement estimator for the marginal likelihood. When multinomial resampling is done at each step, this estimator is simply the product over generations of the weight averages:

$$\hat{\mathbb{P}}(\mathcal{Y}) = \prod_{n=1}^{N} \frac{1}{K} \sum_{k=1}^{K} w_{n,k}. \tag{1.11}$$

This estimator is consistent, i.e. as the number of particles goes to infinity, $\hat{\mathbb{P}}(\mathcal{Y})$ converges to $\mathbb{P}(\mathcal{Y})$. Compared to the naive marginal likelihood harmonic estimator from MCMC samples (Newton and Raftery, 1994), or to other importance sampling methods, the variance of the SMC estimator is generally better behaved, owing to the resampling step (Doucet and Johansen, 2009). However we do not know at the moment of empirical or theoretical studies that compare the behavior of the estimator $\hat{\mathbb{P}}(\mathcal{Y})$ from SMC against more sophisticated MCMC estimators (Chib, 1995; Gelman and Meng, 1998; Lartillot and Philippe, 2006; Xie et al., 2011).

1.3.5 Combining SMC and MCMC

In this section, we review a method for using SMC in combination with MCMC algorithms. Using such combinations can be motivated by tight memory constraints (as discussed in Section 1.3.1), or to jointly sample evolutionary parameters. Both cases are discussed in Wang (2012), and have a firm theoretical foundation based on *particle MCMC* (PMCMC) methods (Andrieu et al., 2010).

We limit the discussion in this section to the second motivation, jointly inferring evolutionary parameters. In this chapter, we have assumed so far that the parameters are fixed and known. This assumption is clearly unrealistic, and it defies one of the main motivations behind using Bayesian methods for phylogenetics: modeling our uncertainty over evolutionary model parameters. Fortunately, PMCMC provides a solution to this issue by alternating between evolutionary parameter resampling and SMC tree reconstruction. Formally, a PMCMC algorithm is an MCMC chain with a powerful proposal distribution constructed using an SMC algorithm. In this section, we describe one of the simplest versions of PMCMC, *particle marginal Metropolis Hastings* (PMMH). We refer the reader to Andrieu et al. (2010) and Wang (2012) for many other possibilities of potential interest in Bayesian phylogenetics.

In PMMH, each step in an MCMC chain is computed as follows. First, we propose a new value θ^* for the evolutionary parameter, using a proposal

that can depend on the current value θ of the parameter. We then use Algorithm 1 to create a new set of weights $w^*_{n,k}$, partial states $s^*_{n,k}$, and associated approximation $\hat{\pi}^*$ targeting $\pi(t) = p(t|\theta^*, \mathcal{Y})$. We select a proposed tree t^* by sampling from $\hat{\pi}^*$. Because the SMC algorithm is run with a finite number of particles, t^* is only approximately distributed according to the posterior given θ^*, however in the following we show how to compute an acceptance ratio to correct this discrepancy.

Computing the acceptance ratio is accomplished by making use of the approximation of the marginal likelihood introduced in (1.11). More precisely, we will construct a ratio reminiscent of a Bayes factor comparing the current population of particles to the previous population—i.e. to the population produced by the SMC algorithm during the last accepted MCMC step.

We denote the unnormalized weights of this previous population by $w^{(i-1)}_{n,k}$: the index i indexes MCMC iterations (which can be thought as the outer loop of the algorithm), the index n and k denote, as before, the particle generation and index (which become indices for inner loops in PMCMC). Similarly, $w^*_{n,k}$ denotes the weights of the proposed population. The population is accepted if a uniform random number u satisfies:

$$u \leq \frac{\prod_n \frac{1}{K} \sum_k w^*_{k,n}}{\prod_n \frac{1}{K} \sum_k w^{(i-1)}_{k,n}} = \prod_n \frac{\sum_k w^*_{k,n}}{\sum_k w^{(i-1)}}.$$

If the proposal is accepted, we set the next population, evolutionary parameter and tree to the proposed values, $w^{(i)}_{k,n} = w^*_{k,n}$, $\theta^{(i)} = \theta^*$, $t^{(i)} = t^*$. Otherwise, we keep the old values, $w^{(i)}_{k,n} = w^{(i-1)}_{k,n}$, $\theta^{(i)} = \theta^{(i-1)}$, $t^{(i)} = t^{(i-1)}$.

For all K, this scheme was shown in Andrieu et al. (2010) to be a valid MCMC algorithm. In other words, for a function of interest ϕ on the trees and evolutionary parameters, as the number of MCMC iterations I goes to infinity,

$$\lim_{I \to \infty} \frac{1}{I} \sum_{i=1}^{I} \phi(t^{(i)}, \theta^{(i)}) = \int \phi(t, \theta) \pi(\, \mathrm{d}t, \, \mathrm{d}\theta),$$

if weak regularity conditions are satisfied, see Andrieu et al. (2010) for a precise statement. Note that we do not have to assume an increase of K as the MCMC iteration index increases. This is important since higher values of K require more memory, while the cost of higher I is only an increase in time.

While they are not required by the basic consistency result, higher values of K do help obtaining a faster mixing. In fact, as K goes to infinity, the acceptance probability converges to one (Andrieu et al., 2010). At the other end of the spectrum, the case where $K = 1$ reduces to a standard MCMC algorithm.

1.4 Discussion

We have reviewed in this chapter various SMC techniques based on incremental construction of forest. These techniques differ from standard MCMC methods in interesting ways. In particular, new types of proposals can be considered, and likelihood calculations at a given generation reuse calculations from previous generations. Even more importantly, multi-core parallelization in SMC algorithms is easy to implement. The computational bottleneck in most cases is to sample from the proposal K independent times, so for a large K, SMC can be qualified as an "embarrassingly parallel algorithm" (Foster, 1995). This type of parallelization has been tested for phylogenetic SMC samplers in Wang (2012), yielding promising results on small numbers of cores. Scaling this technique to hundreds of cores is theoretically possible using the same techniques.

Some of the benefits of SMC, easy parallelization in particular, can also be brought to samplers using more traditional state representations and proposals. In *population Monte Carlo* samplers, each particle is a complete state (Cappé et al., 2004), just as in a standard MCMC sampler. Note that this type of sampler requires a weight update similar to (1.9), based on a backward kernel. See Cappé et al. (2004) and Moral (2004) for details.

We have focussed so far on purely *computational* considerations for motivating phylogenetic SMC: computing posterior on bigger datasets but using existing models. However, there is also a very promising *statistical* potential. We conclude this chapter by giving an overview of this future direction.

Let us turn our attention to the continuous time processes used to model the evolution of a biological sequence along one edge of a phylogenetic tree. Since the pioneer work by Jukes and Cantor (1969), there has been tremendous progress in making evolutionary models more realistic. These advances include using generalized rate models (Tavaré, 1986), local correlations (Goldman and Yang, 1994), rate variation (Yang, 1996; Thorne et al., 1998), and indel operations (Thorne et al., 1991).

However, current models are still lacking many known components of evolution. For example the work of Nasrallah et al. (2011) shows that ignoring structural constraints, as most current models do, can lead to biased tree estimates. This issue was shown to be especially severe in the case of RNA sequences. This motivates the development of more sophisticated likelihood models that take structural constraints into account.

Serious computational challenges arise in the development of these new likelihood models, especially within the Bayesian framework. The key difficulty in going beyond independent site models lies in the computation of the marginal probability of the observed sequences given a tree. This step requires complicated high-dimensional data augmentations or auxiliary variables (Tanner and Wong, 1987) that can be difficult to resample within the standard

MCMC methodology. Examples of these auxiliary variables include partially or fully resolved internal sequences, as well as non-local evolutionary events localized on a phylogeny.

By sequentially constructing trees, SMC can be used to jointly propose a population of possible values for the high-dimensional auxiliary variables. More precisely, in the SMC framework, the problem of proposing auxiliary values can be reduced to proposing values along a single branch at a time. To approach these smaller, single branch problems, techniques from end-point conditioned sampling setups can be used (Hobolth and Stone, 2009).

Bibliography

Altekar, G., Dwarkadas, S., Huelsenbeck, J. P., and Ronquist, F. (2004). Parallel Metropolis coupled Markov chain Monte Carlo for Bayesian phylogenetic inference. *Bioinformatics* **20**, 407–415.

Andrieu, C., Doucet, A., and Holenstein, R. (2010). Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society, Series B* **72**, 269–342.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Berlin: Springer.

Bouchard-Côté, A., Sankararaman, S., and Jordan, M. I. (2012). Phylogenetic inference via sequential Monte Carlo. *Systematic Biology* **61**, 579–593.

Cappé, O., Guillin, A., Marin, J., and Robert, C. (2004). Population Monte Carlo. *Journal of Computational and Graphical Statistics* **13**, 907–929.

Carpenter, J., Clifford, P., and Fearnhead, P. (1999). An improved particle filter for non-linear problems. *IEE Proceedings: Radar, Sonar and Navigation* **146**, 2–7.

Chib, S. (1995). Marginal likelihood from the Gibbs output. *Journal of the American Statistical Association* **90**, 1313–1321.

Crisan, D. and Doucet, A. (2002). A survey of convergence results on particle filtering for practitioners. *IEEE Transactions on Signal Processing* **50**, 736–746.

Douc, R., Cappé, O., and Moulines, E. (2005). Comparison of resampling schemes for particle filtering. In S. Lončarić, H. Babić, and M. Bellanger, editors, *Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis*, pages 64–69. Zagreb: Faculty of Electrical Engineering and Computing.

Doucet, A. (1997). *Monte Carlo methods for Bayesian estimation of hidden Markov models. Application to radiation signals*. Ph.D. thesis, University Paris-Sud Orsay.

Doucet, A., Freitas, N. D., and Gordon, N., editors (2001). *Sequential Monte Carlo methods in practice*. Berlin: Springer.

Doucet, A. and Johansen, A. M. (2009). A tutorial on particle filtering and smoothing: fifteen years later. In D. Crisan and B. Rozovsky, editors, *Oxford Handbook of Nonlinear Filtering*. Cambridge: Cambridge University Press.

Felsenstein, J. (1973). Maximum-likelihood estimation of evolutionary trees from continuous characters. *American Journal of Human Genetics* **25**, 471–492.

Felsenstein, J. (1981). Evolutionary trees from DNA sequences: a maximum likelihood approach. *Journal of Molecular Evolution* **17**, 368–376.

Felsenstein, J. (2004). *Inferring phylogenies*. Sunderland: Sinauer Associates.

Feng, X., Buell, D. A., Rose, J. R., and Waddell, P. J. (2003). Parallel algorithms for Bayesian phylogenetic inference. *Journal of Parallel and Distributed Computing* **63**, 707–718.

Foster, I. (1995). *Designing and building parallel programs*. London: Addison-Wesley.

Friedland, B. (2005). *Control system design: an introduction to state space methods*. Mineola: Dover Publications.

Gelman, A. and Meng, X.-L. (1998). Simulating normalizing constants: from importance sampling to bridge sampling to path sampling. *Statistical Science* **13**, 163–185.

Goldman, N. and Yang, Z. (1994). A codon-based model of nucleotide substitution for protein-coding DNA sequences. *Molecular Biology and Evolution* **11**, 725–736.

Görür, D., Boyles, L., and Welling, M. (2012). Scalable inference on Kingman's coalescent using pair similarity. *Journal of Machine Learning Research* **22**, 440–448.

Görür, D. and Teh, Y. W. (2008). An efficient sequential Monte Carlo algorithm for coalescent clustering. In *Advances in Neural Information Processing*, pages 521 – 528. Red Hook: Curran Associates.

Hackett, S. J. et al. (2008). A phylogenomic study of birds reveals their evolutionary history. *Science* **320**, 1763–1768.

Hobolth, A. and Stone, E. A. (2009). Simulation from endpoint-conditioned, continuous-time Markov chains on a finite state space, with applications to molecular evolution. *Annals of Applied Statistics* **3**, 1204–1231.

Jorgensen, B. (1982). *Statistical properties of the generalized inverse Gaussian distribution*, volume 9 of *Lecture Notes in Statistics*. Berlin: Springer.

Jukes, T. and Cantor, C. (1969). Evolution of protein molecules. In H. Munro, editor, *Mammalian Protein Metabolism*. Academic Press.

Kotecha, J., Jianqui, Z., Yufei, H., Ghirmai, T., Bugallo, M., and Miguez, J. (2003). Particle filtering. *Signal Processing Magazine, IEEE* **20**, 19–38.

Lartillot, N. and Philippe, H. (2006). Computing Bayes factors using thermodynamic integration. *Systematic Biology* **55**, 195–207.

Li, J. Z., Absher, D. M., Tang, H., Southwick, A. M., Casto, A. M., Ramachandran, S., Cann, H. M., Barsh, G. S., Feldman, M., Cavalli-Sforza, L. L., and Myers, R. M. (2008). Worldwide human relationships inferred from genome-wide patterns of variation. *Science* **319**, 1100–1104.

Mardis, E. (2011). A decade's perspective on DNA sequencing technology. *Nature* **470**, 198–203.

Moral, P. D. (2004). *Feynman-Kac formulae*. Berlin: Springer.

Moral, P. D., Doucet, A., and Jasra, A. (2006). Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society, Series B* **68**, 411–436.

Morrison, D. (2009). Why would phylogeneticists ignore computerized sequence alignment? *Systematic Biology* **58 (1)**, 150–158.

Nasrallah, C. A., Mathews, D., and Huelsenbeck, J. (2011). Quantifying the impact of dependent evolution among sites in phylogenetic inference. *Systematic Biology* **60(1)**, 60–73.

Newton, M. A. and Raftery, A. E. (1994). Approximate Bayesian inference by the weighted likelihood bootstrap. *Journal of the Royal Statistical Society, Series B* **56**, 3–48.

Pachter, L. (2007). Proceedings of symposia in applied mathematics **64**, 1–20.

Pitt, M. and Shephard, N. (1999). Filtering via simulation: auxiliary particle filters. *Journal of the American Statistical Association* **94**, 590–599.

Ripley, B. D. (1987). *Stochastic simulation*. New York: John Wiley and Sons.

Shah, S. et al. (2009). Mutational evolution in a lobular breast tumour profiled at single nucleotide resolution. *Nature* **461**, 809–813.

Suchard, M. A. and Rambaut, A. (2009). Many-core algorithms for statistical phylogenetics. *Bioinformatics* **25**, 1370–1376.

Tanner, M. A. and Wong, W. H. (1987). The calculation of posterior distributions by data augmentation. *Journal of the American Statistical Association* **82**, 528–550.

Tavaré, S. (1986). Some probabilistic and statistical problems in the analysis of DNA sequences. *Lectures on Mathematics in the Life Sciences (American Mathematical Society)* **17**, 57–86.

Teh, Y. W., Daume III, H., and Roy, D. M. (2008). Bayesian agglomerative clustering with coalescents. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing 20*, pages 1473–1480. Cambridge: MIT Press.

Thorne, J. L., Kishino, H., and Felsenstein, J. (1991). An evolutionary model for maximum likelihood alignment of DNA sequences. *Journal of Molecular Evolution* **33**, 114–124.

Thorne, J. L., Kishino, H., and Painter, I. S. (1998). Estimating the rate of evolution of the rate of molecular evolution. *Molecular Biology and Evolution* **15**, 1647–1657.

Valentini, A., Pompanon, F., and Taberlet, P. (2009). DNA barcoding for ecologists. *Trends in Ecology and Evolution* **24**, 110–117.

Wang, L. (2012). *Bayesian phylogenetic inference via Monte Carlo methods*. Ph.D. thesis, University of British Columbia.

Wapinski, I., Pfeffer, A., Friedman, N., and Regev, A. (2007). Automatic genome-wide reconstruction of phylogenetic gene trees. *Bioinformatics* **23(13)**, i549–i558.

Wetterstrand, K. A. (2012). DNA sequencing costs: data from the NHGRI large-scale genome sequencing program. `http://www.genome.gov/sequencingcosts`. Accessed: 20/06/2012.

Xie, W., Lewis, P. O., Fan, Y., Kuo, L., and Chen, M.-H. (2011). Improving marginal likelihood estimation for Bayesian phylogenetic model selection. *Systematic Biology* **60**, 150–160.

Yang, Z. (1996). Among-site rate variation and its impact on phylogenetic analyses. *Trends in Ecology and Evolution* **11**, 367–372.

Index