Version dated: May 24, 2018

An Annealed Sequential Monte Carlo Method for Bayesian Phylogenetics

LIANGLIANG WANG[1], SHIJIA WANG[1], ALEXANDRE BOUCHARD-CÔTÉ[2]

[1]*Department of Statistics and Actuarial Science, Simon Fraser University, Burnaby, BC, V5A 1S6, Canada;*

[2]*Department of Statistics, University of British Columbia, Vancouver, Canada*

Corresponding author: Liangliang Wang, Department of Statistics and Actuarial Science, Simon Fraser University, Burnaby, BC, V5A 1S6, Canada; E-mail: lwa68@sfu.ca.

Abstract.—Bayesian phylogenetics, which approximates a posterior distribution of phylogenetic trees, has become more and more popular with the development of Monte Carlo methods. Standard Bayesian estimation of phylogenetic trees can handle rich evolutionary models, but requires expensive Markov chain Monte Carlo (MCMC) simulations, which suffers from two difficulties, the curse of dimensionality and the local-trap problem. Recent research has shown that sequential Monte Carlo (SMC) methods can serve as good alternatives to MCMC in posterior inference over phylogenetic trees. In this paper, an annealed SMC method with an adaptive temperature scheduling is proposed to estimate posterior distributions for general unrooted trees. The proposed SMC can incorporate the MCMC kernels from the literature of Bayesian phylogenetics and provide unbiased estimator of the normalizing constant without extra computational cost. We illustrate our method using simulation studies and real data analysis.

(Keywords: sequential Monte Carlo; phylogenetics; Markov chain Monte Carlo; continuous-time Markov chain)

(1) [[ABC: I think it will be useful to shorten and refocus intro/abstract. Focus should be on: adaptive normalization estimates. Probably less background on phylo needed if targeting sys bio, and perhaps more on Bayes/computational stats. Will get back to intro+abstract after going over other sections. We could also add two points: new check for correctness, and the new smc is easier to incorporate into existing phylo software. Finally, no need to list problems with MCMC I think. Also, correction of claim of unbiasness.]]

Introduction

- Model selection is important in phylogenetics - selection of evolutionary models - routinely needed - reversibility, homogeneity, etc - in Bayes phylo, done by comparing Bayes factors (review), introduce very minimal notation - difficulty (use Liangliang's text) - notation for estimator - contribution: exploration of novel phylogenetic annealed SMC method - adaptation story - unbiasness story

In Bayesian phylogenetics (Lemey et al. 2009; Drummond and Suchard 2010; Huelsenbeck and Ronquist 2001; Ronquist and Huelsenbeck 2003a; Ronquist et al. 2012; Suchard and Redelings 2006), the main challenge is to compute the posterior over a phylogenetic tree space. The exact calculation of this posterior involves summing over all possible trees, and for each tree, integrating over all possible combinations of branch lengths. This challenging posterior computation is typically carried out by running Markov chain Monte Carlo (MCMC)

algorithms for long periods (Rannala and Yang 1996; Yang and Rannala 1997; Mau et al. 1999; Larget and Simon 1999; Li et al. 2000; Holder and Lewis 2003; Rannala and Yang 2003; Lakner et al. 2008; Höhna et al. 2008; Höhna and Drummond 2012). Many user-friendly software packages have been developed for implementing MCMC for phylogenetics, such as MrBayes (Ronquist et al. 2012), BEAST (Drummond and Rambaut 2007), and BAli-Phy (Suchard and Redelings 2006). Due to combinatorial constraints, the distribution over tree space is complex and multimodal (Lakner et al. 2008), and the main difficulty lies in the efficiency with which topology proposals sample the tree space. It is mainly the proposal distribution for tree topology that determines the performance of an MCMC method in Bayesian phylogenetics. The tree topology proposals include the simple moves such as Nearest Neighbor Interchange (NNI) (Lakner et al. 2008) and more complicated moves such as Subtree Prune and Regraft (SPR) (Lakner et al. 2008; Höhna et al. 2008; Höhna and Drummond 2012).

There are several drawbacks for MCMC in phylogenetic inference. First, it is very challenging to design an MCMC algorithm with good mixing due to the complex posterior tree distribution. 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, at each MCMC iteration, only small MCMC moves are allowed as large moves result in a high rejection rate, requiring the MCMC chain to run for a long time to explore the full possibilities of 'tree space'. Second, it is hard to determine if the Markov chain gets converged and decide the burn-in stage. The distribution on tree space is often a complex multimodal distribution due to combinatorial constraints. It is nontrivial to judge if the chain has actually converged or it is trapped to a local mode. Third, Bayesian model selection requires to compute the marginal likelihood, i.e. the normalizing constant of the posterior distribution, which is generally computationally expensive using MCMC. One of the most popular methods for estimating the normalizing constant is the Stepping Stone method (Xie et al. 2010) used in MrBayes (Ronquist and Huelsenbeck 2003b).

Recent research (Teh et al. 2008; Görür and Teh 2009; Görür et al. 2012; Bouchard-Côté et al. 2012; Wang et al. 2015) has shown that sequential Monte Carlo (SMC) methods can serve as good alternatives to and combine with MCMC algorithms in posterior inference over phylogenetic trees. Earlier work on applying SMC to phylogenetics (Teh et al. 2008; Görür and Teh 2009; Görür et al. 2012; Bouchard-Côté et al. 2012) is limited in the type of available phylogenetic proposals, and cannot handle unrooted trees in a natural framework. This is an important limitation, as most current work in phylogenetics relies on unrooted tree models. To overcome the limitation, Wang et al. (2015) proposed an SMC algorithm for general trees based on a graded partially ordered set on an extended combinatorial space and jointly inferred the phylogenetic tree and the associated evolutionary parameters based on particle MCMC (Andrieu et al. 2010). Moreover, some researchers worked on the online phylogenetic tree inference via SMC algorithm, in which new observations can be sequentially incorporated to update the posterior tree distribution. Dinh et al. (2016) focused on the theoretical framework for the online phylogenetic inference using SMC approaches. Fourment et al. (2017) investigated the importance of 'guided' proposal distribution in online phylogenetic tree inference. Everitt et al. (2016) described an online phylogenetic inference targeting the spaces of varying dimension for coalescent trees. In addition, SMC algorithm has been applied to estimate the transmission network. Smith et al. (2017) jointly estimated the phylogenetic tree and disease transmission model via sequential Monte Carlo methods.

In this paper, we propose to use an annealed SMC algorithm for general unrooted trees for Bayesian phylogenetic inference. The proposed method is based on the framework of the SMC sampler (Del Moral et al. 2006, 2007), which is a general method for obtaining a set of samples from a sequence of distributions which can exist on the same spaces. The most challenging part is to design the sequence of intermediate distributions. We decide to choose the intermediate distribution to be proportional to the temperated likelihood times the prior. Then the difficulty becomes how to choose the sequence of temperatures appropriately. We investigate the choice of

the temperatures. Further, motivated by the bias of the stepping stone method, we examine the marginal likelihood estimates from the proposed SMC algorithm and from linked importance sampling.

We have three main contributions. First, we introduce the annealed SMC algorithm to Bayesian phylogenetic inference for unrooted trees. The work in Del Moral et al. (2006, 2007) has been used for continuous spaces but has not been explored for the combinatorial tree space. The annealed SMC algorithm provides an alternative and straightforward way to combine MCMC and SMC algorithms. The commonly used proposal distributions of MCMC in Bayesian phylogenetics can be adapted to the proposed annealed SMC. Meanwhile, the annealed SMC is a valid SMC algorithm which enjoys the advantages of SMC.

Secondly, we explore different approaches to design the sequence of temperatures. In our proposed adaptive annealed SMC, the temperatures for each intermediate distribution is random, which is controlled by the pre-determined conditional effective sample size and sampled particles. This adaptive temperature scheme will automatically select the 'optimum' intermediate temperature to avoid the SMC from being too expensive or collapsed due to improper selection of temperatures.

Thirdly, we compare four methods for estimating the normalizing constant of the posterior distribution, i.e. the marginal likelihood, in Bayesian phylogenetics: adaptive annealed SMC, deterministic annealed SMC, stepping stone (widely used in MrBayes), and linked importance sampling. SMC method can provide an unbiased estimator of marginal likelihood as a by-product of the algorithm, which does not involve any extra computational cost. The marginal likelihood estimates provided by the adaptive annealed SMC is a better alternative to the ones obtained from linked importance sampling and stepping stone for Bayesian phylogenetics. The adaptive temperature scheme makes the estimation of normalizing constants more stable than the deterministic annealed SMC.

The rest of the paper is organized as follows. In Section 2 (Bayesian phylogenetics), we

provide the backgrounds and introduce notations. The method of the annealed SMC for unrooted phylogenetic tree inference is described in Section 3 (methodology). Performance of the proposed method are examined through simulation studies in Section 4. In Section 5, we analyze two real datasets downloaded from TreeBase and compare with results of running MrBayes for a long time. The conclusion and discussion is given in Section 6. Our implementation is available at `https://github.com/...`.

Bayesian Phylogenetics

A phylogenetic tree t represents the evolutionary relationship among species (observed taxa). We assume a tree t contains the tree topology and a set of positive branch lengths. We consider the general unrooted trees that can handle non-constant evolutionary rates (Thorne et al. 1998; Drummond and Suchard 2010). Phylogenetic reconstruction is based on observed information located at the leaves of phylogeny (e.g. DNA sequences for different species). Our objective is to infer the phylogenetic tree t and unknown parameters in the evolutionary model θ using n observed biological sequences, denoted $\mathcal{Y}$. We consider the Bayesian framework for phylogenetic inference. Let $p(\theta)$ be the prior density for θ. Let $\mathcal{X}$ be the space of all possible trees. The prior density given θ is denoted by $p(t|\theta)$. For example, a common prior over unrooted trees consists of a uniform distribution over topologies and a product of independent exponential distributions over the branch lengths. The probability of the observed data $\mathcal{Y}$ given parameters θ and a tree t is $\mathbb{P}(\mathcal{Y}|\theta, t)$. Our interest relies on the joint posterior inference of t and θ,

$$p(\theta, t|\mathcal{Y}) = p(\theta|\mathcal{Y})p(t|\mathcal{Y}, \theta) = \frac{\mathbb{P}(\mathcal{Y}|\theta, t)p(t|\theta)p(\theta)}{\mathbb{P}(\mathcal{Y})}, \tag{1}$$

where the normalizing constant, $\mathbb{P}(\mathcal{Y}) = \int\int \mathbb{P}(\mathcal{Y}|\theta, t)p(t|\theta)p(\theta)\,\mathrm{d}\theta\,\mathrm{d}t$, is intractable.

In phylogenetic literature, the sites of a biological sequence are often assumed to be independent, and a continuous-time Markov chain (CTMC) is used to model the evolution of each

site. Let Q denote the rate matrix of the CTMC, depending on the unknown parameter(s) θ. In the simulation studies, We will use the Kimuras two parameter (K2P) model (Kimura 1980), and fix the only unknown parameter κ of the transition/transversion rate to be 2. In the real data analysis, we will use the Jukes and Cantor 1969 (JC69) model (Jukes et al. 1969), in which each base (A, C, G, T) in the sequence has an equal probability of transition/transversion.

The space under consideration is a joint space over all the possible trees and all the evolutionary parameters, denoted $E = \mathcal{X} \times \Theta$. An MCMC algorithm generates a sequence of dependent samples of phylogenetic trees and evolutionary parameters from the space E that are distributed approximately according to the posterior distribution. In the next section, we will propose an annealed SMC method as an alternative Monte Carlo method for Bayesian phylogenetics.

Besides sampling the phylogenetic trees and parameters from the posterior distribution, another goal is to estimate the normalizing constant in Equation (1), which plays an important role in model selection. Currently, stepping stone (Xie et al. 2010) is one of the most popular methods for estimating normalizing constant in Bayesian phylogenetic, which is widely used in MrBayes (Ronquist and Huelsenbeck 2003b). In stepping stone, a list of powered posterior distributions are introduced to bridge between prior and posterior distributions. Importance sampling is used to estimate the ratio between two intermediate normalizing constant. Fan et al. (2010) proposed the generalized stepping stone by using a reference distribution sampled from posterior distribution. This method can alleviate the challenge exists in the original stepping stone, sampling from intermediate posterior distributions that are close to the prior distribution. These methods are based on multiple runs of expensive MCMC algorithms. More importantly, the estimate is biased, which will be shown in the simulation studies.

Annealed SMC for phylogenetics

The SMC sampler framework proposed by Del Moral et al. (2006, 2007) is a general method for obtaining a set of samples from a sequence of distributions $\{\pi_r\}$, where $r = \{1, 2, \cdots, R\}$. In this section we describe a way this general methodology can be applied to Bayesian phylogenetics in the context of annealing.

Sequences of distributions

(2) [[TODO: introduce gamma]]

In standard MCMC methods, we are interested in a single probability distribution, the posterior distribution. There are two reasons why we may use a *sequence* of distributions rather than only one.

A first possibility is that we may have an online problem, where the data is revealed sequentially and we want to perform inference sequentially in time based on the data available so far. The distribution at step r is then the posterior distribution conditioning on the first r batches of data. This approach is explored in the context of phylogenetics in Dinh et al. (2016), where a batch of data in this paper consists in genomic information for one additional operational taxonomic unit.

A second reason for having multiple distributions is to facilitate the exploration of the state space. ***(3) [[TODO: import notation from later in the paper.]]*** This is achieved for example by raising the likelihood term to a power between zero and one. If the exponent is zero, then the distribution becomes the prior which is often easy to explore and in fact independent samples can be extracted in many situations. At the other extreme, the distribution at power one is the distribution of interest. Such sequences of distributions have been used before in the context of phylogenetics, to perform parallel tempering for instance. Here we use the construction in the context of an SMC algorithm instead.

A powerful feature of the the SMC sampler framework is that the space on which the distributions π_r are defined is allowed to vary from one iteration to the next. All previous work on

SMC methods for phylogenetics have exploited this feature for different purposes: ***(4) [[TODO: move to literature review]]***

1. A range of bottom up methods ***(5) [[cite]]*** have been proposed which allows more efficient reuse of intermediate stages of the Felsenstein pruning recursions. For these methods, the intermediate distributions are defined over forests over the observed taxa, and hence their dimensionality increases with r. These methods are most effective in clock or nearly-clock trees. For general trees, it is typically necessary to perform additional MCMC steps, which makes it harder to use in the context of estimation of Bayes factors.

2. Methods ***(6) [[cite]]*** optimized for scenarios where taxonomic data comes in an online fashion.

3. XXX reversible jump method

One drawback of letting the dimensionality of π_r vary with r is that it makes it significantly harder to incorporate SMC into existing Bayesian phylogenetic inference packages such as RevBayes or BEAST. Motivated by this, we explore phylogenetic SMC algorithms where the target distributions π_r are all defined in the same space. The SMC samplers framework in this context utilizes Metropolis-Hastings kernels in the inner loop but combines them in a different fashion compared to standard MCMC algorithms, or even compared to parallel tempering MCMC algorithms.

Basic Annealed SMC Algorithm

We now turn to the description of the annealed SMC in the context of Bayesian phylogenetic inference. The algorithm fits into the generic framework of SMC algorithms Del Moral (2004): at each iteration, indexed by $r = 1, 2, \ldots, R$, we maintain a collection indexed by $k \in \{1, 2, \ldots, K\}$ of imputed latent states $x_{r,k}$, each paired with a non-negative number called a

weight $w_{r,k}$; such a pair is called a particle. A latent state in our context consists in a hypothesized tree $t_{r,k}$ and evolutionary set of parameter $\theta_{r,k}$, i.e. $x_{r,k} = (t_{r,k}, \theta_{r,k})$. In contrast to previous SMC methods, $x_{r,k}$ is always of the same data type: no partial states such as forest or trees over subsets of leaves are considered here.

A particle population consists in a list of particles $(x_{r,\cdot}, w_{r,\cdot}) = \{(x_{r,k}, w_{r,k}) : k \in \{1, \ldots, K\}\}$. A particle population can be used to estimate posterior probabilities as follows: first, normalize the weights, which we denote using capital letter by convention, $W_{r,k} = w_{r,k} / \sum_{k'} w_{r,k'}$. Second, use the approximation:

$$\int \pi_r(x) f(x) \, \mathrm{d}x \approx \sum_{k=1}^{K} W_{r,k} f(x_{r,k}) \tag{2}$$

For example if we seek a posterior clade support for a subset $X' \subset X$ of the leaves X, this becomes

$$\sum_{k=1}^{K} W_{r,k} 1[\text{sampled tree } t_{r,k} \text{ admits } X' \text{ as a clade}],$$

where $1[s]$ denotes the indicator function which is equal to one if the logical statement s is true and zero otherwise. The above formula is most useful at the last SMC iteration, $r = R$, since π_R coincides with the posterior distribution by construction.

At the first iteration, each of the particles' tree and evolutionary parameters are sampled independently and identically from their prior distributions. We assume for simplicity that this prior sampling step is tractable, a reasonable assumption in many phylogenetic models. ***(7) [[example and reference for sampling over tree priors]]*** After initialization, we therefore have a particle-based approximation of the prior distribution. Intuitively, the goal behind the annealed SMC algorithm is to progressively transform this prior distribution approximation into a posterior distribution approximation.

To formalize this intuition, we use the sequence of distributions introduced in the previous section. The last ingredient required to construct an SMC algorithm is an SMC proposal distribution $K_r(x_{r-1,k}, x_{r,k})$, used to sample a particle for the next iteration given a particle from the

previous iteration. Since $x_{r-1,k}$ and $x_{r,k}$ have the same dimensionality in our setup, it is tempting to use *MCMC* proposals $q_r(x_{r-1,k}, x_{r,k})$ (for example, subtree prune and regraft moves, and Gaussian proposals for the continuous parameters and branch lengths) in order to build *SMC* proposals.

There are several advantages of using MCMC proposal for SMC. First, this mean we can leverage a rich literature on the topic ***(8) [[some cites]]***. Second, it makes it easier to add SMC support to existing MCMC-based software libraries. Third, it makes certain benchmark comparison between SMC and MCMC more direct, as we can then set the set of moves to be the same for both. On the flip side, constructing MCMC proposals is somewhat more constrained, so some of the flexibility provided by the general SMC framework is lost.

Naively, we could pick the SMC proposal directly from an MCMC proposal, $K_r(x_{r-1,k}, x_{r,k}) = q_r(x_{r-1,k}, x_{r,k})$. However, doing so would have the undesirable property that the magnitude of the fluctuation of the weights of the particles from one iteration to the next, $\|w_{r-1,\cdot} - w_{r,\cdot}\|$, does not converge to zero when the annealing parameter change Δ goes to zero. This can cause particle degeneracy problems ***(9) [[cite]]***, forcing the use of a number of particles larger than what can be realistically accommodated in memory.

To avoid this issue, we follow Del Moral (2004) and use as SMC proposal the accept-reject Metropolis-Hastings transition probability based on q_r, which can be simulated from as follows:

1. Propose a new tree and/or new evolutionary parameters, denoted x_r^*, from $q_r(x_{r-1}, \cdot)$.

2. Compute the Metropolis-Hastings ratio is computed as

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

3. With probability $\alpha_r(x_{r-1}, x_r^*)$, output the proposal x_r^*, and with (1-$\alpha_r(x_{r-1}, x_r^*)$) probability, output the previous state x_{r-1}.

We review in Appendix ***(10) [[move argument to appendix]]*** the argument of Del Moral (2004) showing that if we use the weight formula:

$$w_{r,k} = \frac{\gamma_r}{\gamma_{r-1}}(x_{r-1}), \tag{3}$$

then the SMC algorithm has the standard key theoretical properties under regularity conditions (unbiasness, consistency, asymptotic normality). ***(11) [[TODO: can we check if the conditions are satisfied here? can we prototype confidence interval estimation?]]***

In the important special case where $\gamma(x_r)$ is equal to the prior times an annealed likelihood, we obtain

$$w_{r,k} = [p(y|x_{r-1,k})]^{\phi_r - \phi_{r-1}}, \tag{4}$$

This update equation has the property that weight fluctuations vanish as the temperature difference $\Delta_r = \phi_r - \phi_{r-1}$ goes to zero. This will form the basis of the annealing sequence adaptation strategies described in the next section.

Notice also that in contrast to typical SMC algorithms, the annealed SMC algorithm does not require pointwise evaluation of the proposal $K_r(x_{r-1,k}, x_{r,k})$ (i.e. given $x_{r-1,k}$ and a sampled $x_{r,k}$, we do not need to compute the numerical value of $K_r(x_{r-1,k}, x_{r,k})$ as it does not appear in the weight update formula, Equation 3. This point is important, since for Metropolis-Hastings kernels, pointwise evaluation would require computation of a typically intractable integral under the proposal in order to compute the total probability of rejection. The theoretical justification as to why we do not need pointwise evaluation of K is detailed in Appendix ***(12) [[TODO]]***.

We give in Appendix the list of MCMC proposal we consider. ***(13) [[we should also write more explicitly what is used for the continuous parameters]]***

For pedagogical purpose we show in Algorithm ***(14) [[first algo]]*** the simplest version of annealed SMC. The simple version of the algorithm essentially alternates between sampling from the proposals, reweighting, and resampling. However to make the algorithm more efficient and

reduce the number of tuning parameters, two adaptive mechanisms are described in the next section.

Algorithm 1 The simplest version of annealed SMC algorithm

Inputs: (a) Prior over evolutionary parameters and trees, $p(x)$, where $x = (\theta, t)$; (b) Likelihood function $p(y|x)$; (c) Sequence of annealing parameters $0 = \phi_0 < \phi_1 < \cdots < \phi_R = 1$.
Outputs: Approximation of the posterior distribution, $\sum_k \tilde{W}_{R,k}\delta_{\tilde{x}_{R,k}}(\cdot) \approx \pi(\cdot)$.
SMC iteration index $r \leftarrow 0$
Annealing parameter $\phi_r \leftarrow 0$
for each particle index $k \in \{1, 2, \ldots, K\}$ **do**
    Initialize particles with independent samples from the prior over evolutionary parameters and trees, $x_{0,k} \leftarrow (\theta_{0,k}, t_{0,k}) \sim p(\cdot)$
    Initialize weights to unity: $w_{0,k} \leftarrow 1$.
end for
for SMC iterations $r = 1, 2, \ldots R$ **do**
    for $k \in \{1, 2, \ldots, K\}$ **do**
        Sample particles $\tilde{x}_{r,k} \sim K_r(x_{r-1,k}, \cdot)$ for each k, using a transition probability K_r admitting π_r as stationary (typically, K_r is built via a mixture or an alternation of Metropolis-Hastings (MH) moves; note that K_r encapsulates both the MH proposal as well as the accept-reject step).
        Compute unnormalized weights: $w_{r,k} = [p(y|x_{r-1,k})]^{\phi_r - \phi_{r-1}}$.
    end for
    if $r < R$ **then**
        for $k \in \{1, 2, \ldots, K\}$ **do**
            Resample the particles: $x_{r,k} \sim \sum_{k'} W_{r,k'}\delta_{\tilde{x}_{r,k'}}(\cdot)$.
        end for
    else
        No resampling needed at the last iteration.
    end if
    Return the particle population $\tilde{x}_{r,\cdot}, W_{r,\cdot}$
end for

Before moving on to more advanced versions of the algorithm, we provide first some intuition to motivate the need for resampling. Theoretically, the algorithm produces samples from an artificial distribution with state space $\mathcal{X} \times \mathcal{X} \times \cdots \times \mathcal{X} = \mathcal{X}^R$ (this is described in more details in Appendix ***(15) [[TODO]]***). However since we only make use of one copy of $\mathcal{X}$ (corresponding to the particles at the final SMC iteration), we would like to decrease the variance of the state at iteration R (more precisely, of Monte Carlo estimators of functions of the state at iteration R).

This is what resampling for iteration $r < R$ achieves, at the cost of increasing the variance for the auxiliary part of the state space $r < R$. From this argument, it follows that resampling at the last iteration should be avoided.

When resampling is performed at every iteration but the last, an estimate of the marginal probability of the data, $p(y)$, is given by the product of the average unnormalized weights, namely:

$$\hat{Z}_K := \prod_{r=1}^{R} \frac{1}{K} \sum_{k=1}^{K} w_{r,k}. \tag{5}$$

Review of other normalization constant estimation methods

For completeness, we review here some alternatives to Equation 5 for estimating normalization constants, which we will compare to SMC from both a theoretical and empirical stand-point.

Stepping Stone

The Stepping Stone method (Xie et al. 2010) is an alternative method to provide a normalizing constant estimator. It is widely used via its MrBayes implementation Huelsenbeck and Ronquist (2001). As with SMC, the Stepping Stone method introduces a list of annealed posterior distributions connecting the posterior distribution and the prior distribution. We use a notation analogous to SMC, with π_d ($d = 0, 1, 2, \cdots, D$) denoting the D intermediate distributions, $\pi_d(x) \propto \gamma_d(x) = \mathbb{P}(\mathcal{Y}|x)^{\phi_d}\pi(x)$, $0 = \phi_0 < \phi_1 < \phi_2 < \cdots < \phi_D = 1$. The normalizing constant Z can be written as

$$Z \equiv Z_R = Z_0 \prod_{d=1}^{D} \frac{Z_d}{Z_{d-1}}.$$

We can rewrite the ratio of Z_d and Z_{d-1} as ***(16) [[point to same equation that will be used earlier to illustrate norm constant of SMC]]***

$$\frac{Z_d}{Z_{d-1}} = \int \frac{\gamma_r(\boldsymbol{x})}{\gamma_{d-1}(x)} \pi_{d-1}(x) dx.$$

The Stepping Stone method prescribes running several MCMC chains targeting $\pi_{d-1}(x)$ to obtain N posterior samples $x_{d-1,1}, x_{d-1,2}, \cdots, x_{d-1,N}$, then

$$\widehat{\frac{Z_d}{Z_{d-1}}} = \frac{1}{N} \sum_{i=1}^{N} \{\mathbb{P}(\mathcal{Y}|x_{d-1,i})\}^{\phi_d - \phi_{d-1}}. \tag{6}$$

The estimator of the normalizing constants admits the form

$$\widehat{Z_D} = \prod_{d=1}^{D} \frac{1}{N} \sum_{i=1}^{N} \{\mathbb{P}(\mathcal{Y}|x_{d-1,i})\}^{\phi_d - \phi_{d-1}}.$$

The number of intermediate distributions is a trade-off between computing cost and accuracy. A larger number of MCMC chains can provide a better approximation for the normalizing constant, but the computational cost will be higher. To make fair comparison between the marginal likelihood estimators provided by the annealed SMC and SS, we set $K_{SMC} R_{SMC} = N_{SS} D_{SS}$. Another factor that will impact the SS estimator is the strategy we choose the temperature sequence $\{\phi_d\}_{d=1,2,\cdots,D}$. In this paper, we use the temperature scheme $\phi_d = (d/D)^{1/a}$ recommended by Xie et al. (2010), where a is between 0.2 and 0.4.

Linked Importance Sampling

Stepping stone uses importance sampling to approximate the ratio of normalizing constants for two intermediate distributions. However, the IS approximation would be poor if the two successive distributions do not have enough overlaps. Linked Importance Sampling (LIS) (Neal 2005) improves the performance of IS by introducing bridge distributions, e.g. 'geometric' bridge: $\gamma_{d-1*d}(x) = \sqrt{\gamma_{d-1}(x)\gamma_d(x)}$. LIS also provides an unbiased normalizing constant estimator.

The ratio of two normalizing constants can be written as

$$\frac{Z_d}{Z_{d-1}} = \frac{Z_{d-1*d}}{Z_{d-1}} \Big/ \frac{Z_{d-1*d}}{Z_d} = \Big\{ \int \frac{\gamma_{d-1*d}(x)}{\gamma_{d-1}(x)} \pi_{d-1}(x) d\boldsymbol{x} \Big\} \Big/ \Big\{ \int \frac{\gamma_{d-1*d}(x)}{\gamma_d(x)} \pi_d(\boldsymbol{x}) dx \Big\}.$$

For $d = 1, \cdots, D$, to estimate the ratio Z_d/Z_{d-1}, we first run MCMC targeting $\pi_{d-1}(x)$ to obtain N posterior samples $x_{d-1,1}, x_{d-1,2}, \cdots, x_{d-1,N}$ (when $d = 1$, we sample from the prior distribution). Then we sample the initial state of π_d. Two successive MCMC chains $\pi_{d-1}(x)$ and $\pi_d(x)$ are linked by a state μ_{d-1}, which is sampled from $\{1, 2, \cdots, N\}$ according to the following probabilities:

$$p(\mu_{d-1} | x_{d-1,1:N}) = \frac{\gamma_{d-2*d-1}(x_{d-1,\mu_{d-1}})}{\gamma_{d-1}(x_{d-1,\mu_{d-1}})} \Big/ \sum_{i=1}^{N} \frac{\gamma_{d-2*d-1}(x_{d-1,i})}{\gamma_{d-1}(x_{d-1,i})}.$$

In case $d = 1$, the lined state μ_0 is uniformly sampled from the N samples of $\pi_0(x)$. Finally, we run MCMC chain $\pi_d(x)$ starting from initial state $x_{d-1,\mu_{d-1}}$ to obtain N posterior samples $x_{d,1}, x_{d,2}, \cdots, x_{d,N}$. The ratio of two normalizing constants can be approximated by

$$\frac{\widehat{Z_d}}{Z_{d-1}} = \frac{\widehat{Z_{d-1*d}}}{Z_{d-1}} \Big/ \frac{\widehat{Z_{d-1*d}}}{Z_d} = \Big\{ \frac{1}{N} \sum_{i=1}^{N} \frac{\gamma_{d-1*d}(x_{d-1,i})}{\gamma_{d-1}(x_{d-1,i})} \Big\} \Big/ \Big\{ \frac{1}{N} \sum_{i=1}^{N} \frac{\gamma_{d-1*d}(x_{d,i})}{\gamma_d(x_{d,i})} \Big\}.$$

In this paper, we use the 'geometric' bridge. Hence, the estimator of ratio can be simplified to

$$\frac{\widehat{Z_d}}{Z_{d-1}} = \Big\{ \sum_{i=1}^{N} \{\mathbb{P}(\mathcal{Y}|x_{d-1,i})\}^{\frac{\phi_d - \phi_{d-1}}{2}} \Big\} \Big/ \Big\{ \sum_{i=1}^{N} \{\mathbb{P}(\mathcal{Y}|x_{d,i})\}^{\frac{\phi_{d-1} - \phi_d}{2}} \Big\}.$$

Theoretical properties

In this section, we review three theoretical properties of interest, *consistency*, *evidence unbiasness*, and *central limit*, with an emphasis with their respective practical importance.

Properties of Sequential Monte Carlo

In the context of SMC algorithms, the first property, *consistency* means that as the number of particles is increased, the approximation of posterior expectations can become arbitrarily close to the the true posterior expectation. This makes the approximation in Equation 2 more precise:

$$\sum_{k=1}^{K} W_{r,k} f(x_{r,k}) \rightarrow \int \pi_r(x) f(x) \, \mathrm{d}x \text{ as } K \rightarrow \infty, \tag{7}$$

provided f satisfies regularity conditions, for example f bounded, and where convergence of the random variables holds for a set of random seeds having probability one. See for example Wang et al. (2015).

Consistency can be viewed as the "bare minimum" expected from modern SMC algorithms. A more informative class of result consists in central limit theorem equivalents of Equation 7. These results can be used to assess the variance of the total variance of Monte Carlo estimators (whereas measures such as effective sample size described shortly are local in nature) Chan and Lai (2013). However, since numerically stable versions of these methods are still at their infancy Olsson and Douc (2017), we will focus the remaining on unbiasness.

Recall that we say an estimator $\hat{Z}$ for a constant Z is unbiased if $\mathbb{E}\hat{Z} = Z$. Here the expectation is with respect to the randomness of the approximation algorithm. This contrasts with the classical statistical setup where the randomness comes from the data generation process, but the definition is otherwise identical.

For SMC algorithms, unbiasness holds in a more restrictive sense compared to consistency. In general:

$$\mathbb{E}\left[\sum_{k=1}^{K} W_{r,k} f(x_{r,k})\right] \neq \int \pi_r(x) f(x) \, \mathrm{d}x, \tag{8}$$

in other words, repeatedly running SMC with a fixed number of particle but different random seeds and averaging the result does not provide arbitrarily precise approximations. However, if we restrict our attention to the normalization constant estimates, remarkably the unbiasness

property does hold, i.e. for any finite K, $\hat{Z}_K$ as defined in Equation 5 is such that:

$$\mathbb{E}\left[\hat{Z}_K\right] = Z = \int \gamma_r(x)\,\mathrm{d}x. \tag{9}$$

While the notion of bias has been central to frequentist statistics since its inception, only in the past decade has it started to emerge as a property of central importance in the context of (computational) Bayesian statistics. Traditionally, the main theoretical properties analyzed for a given Monte Carlo method $\hat{Z}$ estimating Z was consistency.

With the emergence of pseudo-marginal methods, the bias of Monte Carlo methods is now under closer scrutiny. We refer the reader to Andrieu and Roberts (2009) for examples where unbiasness is used to implement MCMC algorithms for complex models. Another area where unbiasness can play a role is for testing correctness of Monte Carlo procedures. In contrast to correctness tests based on consistency, which are asymptotic in nature and hence necessarily have false positive rates (where the test indicates the presence of a bug when in fact the code is correct), tests based on unbiasness can achieve a false positive rate of zero, using the strategy described in the next section.

Using unbiasness to test implementation correctness

Typically, the algorithm shown in 1 is implemented in a model-agnostic fashion. Hence it is reasonable to assume that we can construct test cases on a discrete state spaces. For example, one can use phylogenetic tree with fixed branch lengths, or even simpler models such as HMM. Furthermore, we observed empirically that many software defects can be detected in relatively small examples, where exhaustive enumeration is possible, and hence Z can be computed exactly. We determined how small the examples to use via code coverage tools (Miller and Maloney 1963).

We would like to test if Equation 9 holds, now that we know what the right-hand side's

numerical value. To compute analytically the expectation on the left-hand side, we use a method that borrows ideas from probabilistic programming (Wingate et al. 2011), and use an algorithm, called ExhaustiveRandom that automatically visit all possible execution traces t_i of a given randomized algorithm. The execution trace of a randomized algorithm refers as all possible realizations of all random choices in the algorith (in the context of SMC, both the resampling steps and the proposal steps). ExhaustiveRandom also computes the respective probability p_i of each execution trace, and can be thought as performing depth first search of the decision tree corresponding to the randomized algorithm being tested. The number of execution traces grows exponentially fast but this is still a useful tool as very small examples are generally sufficient to reach code coverage.

For each execution trace t_i, we can also obtain the normalization estimate $\hat{z}_i$ corresponding to that trace, and hence get the value of the left-hand side of Equation 9 as $\sum_i p_i \hat{z}_i$. We implemented this check using an open source implementation of ExhaustiveRandom[1] to ensure our final implementation of SMC satisfies the unbiasness property. We also use this method to create counter-examples showing that the stepping stone method ran for a fixed number of iterations is biased.

Properties of the stepping stone method

For stepping stone, the expected value of Equation (6) depends on the nature of the samples $x_{d-1,1}, x_{d-1,2}, \ldots, x_{d-1,N}$. If they are independent, the procedure is unbiased. However, if the samples are obtained from a Markov chain, there are no guarantees that the procedure is unbiased unless the MCMC chain is initialized at the exact stationary distribution. In practise, this is not possible: Xie et al. (2010) uses a burned-in MCMC chain, which implies that the chain is *asymptotically unbiased*, for any finite number of iteration, it is biased. Unfortunately, the main

[1] Available at `https://github.com/alexandrebouchard/bayonet/blob/1b9772e91cf2fb14a91f2e5e282fcf4ded61ee22/src/main/java/bayonet/distributions/ExhaustiveDebugRandom.java`.

motivations for unbiasness (pseudo-marginal methods and no false positive correctness tests) require unbiasness for any finite number of Monte Carlo samples; asymptotic unbiasness is not sufficient.

We show in XXXX an explicit counter example where we compute the non-zero bias of the stepping stone method. This clearly motivates the need for implementable unbiased methods.

Log scale of normalizing constant

In Bayesian phylogenetics, the normalizing constant estimates is generally a very small non-negative value. Instead of computing $\hat{Z}$ directly, we compute the logarithm of normalizing constant estimates, $\log(\hat{Z})$. Although $\hat{Z}$ is an unbiased estimator, the log scale of $\hat{Z}$ introduces bias. Jensen's inequality shows that $\log(\hat{Z})$ is a biased estimator of $\log(Z)$, and is generally underestimated,

$$E[\log(\hat{Z})] \leq \log(E(\hat{Z})).$$

Adaptive mechanisms

We discuss how two adaptive schemes from the SMC literature can be applied in our Bayesian phylogenetic inference setup to improve the scalability and usability of the algorithm described in the previous section. The first scheme relaxes the assumption that resampling is performed at every step, and the second is a method for automatic construction of the sequence of annealing parameters. The two mechanisms go hand in hand and we recommend using both simultaneously. The combination yields Algorithm ***(17) [[TODO]]*** which we explain in detail in the next two subsections.

The two adaptive mechanisms make theoretical analysis considerably more difficult. This is a common situation in the SMC literature. A simple work-around in principle is to run the algorithm twice, a first time to adaptively determine the resampling and annealing schedules, and then a second independent time using the schedule fixed in the first pass.

Algorithm 2 An adaptive annealed SMC algorithm

Inputs: (a) Prior over evolutionary parameters and trees, $p(x)$, where $x = (\theta, t)$; (b) Likelihood function $p(y|x)$.
Outputs: (a) Approximation Z of the marginal data likelihood, $Z \approx p(y) = \int p(\mathrm{d}x)p(y|x)$; (b) Approximation of the posterior distribution, $\sum_k \tilde{W}_{R,k}\delta_{\tilde{x}_{R,k}}(\cdot) \approx \pi(\cdot)$.
SMC iteration index $r \leftarrow 0$
Annealing parameter $\phi_r \leftarrow 0$
Evidence estimate $Z \leftarrow 1$
for each particle index $k \in \{1, 2, \ldots, K\}$ **do**
    Initialize particles with independent samples from the prior over evolutionary parameters and trees, $x_{0,k} \leftarrow (\theta_{0,k}, t_{0,k}) \sim p(\cdot)$
    Initialize weights to unity: $w_{0,k} \leftarrow 1$.
end for
for SMC iterations $r = 1, 2, \ldots$ **do**
    Determine the next annealing parameter, $\phi_r = \mathrm{NextAnnealingParameter}(x_{r-1,\cdot}, w_{r-1,\cdot}, \phi_{r-1})$.
    for $k \in \{1, \ldots, K\}$ **do**
        Compute pre-resampling unnormalized weights: $\tilde{w}_{r,k} = W_{r-1,k}[p(y|x_{r-1,k})]^{\phi_r - \phi_{r-1}}$.
        Sample particles $\tilde{x}_{r,k} \sim K_r(x_{r-1,k}, \cdot)$, using a transition probability K_r admitting π_r as stationary (typically, K_r is built via a mixture or an alternation of Metropolis-Hastings (MH) moves; note that K_r encapsulates both the MH proposal as well as the accept-reject step).
    end for
    if $\phi_r = 1$ **then**
        update $Z \leftarrow (Z/K)\sum_k \tilde{w}_{r,k}$, then return updated Z and the current particle population $\tilde{x}_{r,\cdot}, \tilde{W}_{r,\cdot}$.
    end if
    if particle degeneracy is too severe, i.e. $\mathrm{relativeESS}(\tilde{W}_{r,\cdot}) = \left(K\sum_{k=1}^K \tilde{W}_{r,k}^2\right)^{-1} < \epsilon$, where ϵ is the relative Effective Sample Size (relativeESS) threshold (we recommend $\epsilon = 1/2$ as a reasonable default value), **then**
        Update normalization constant estimate, $Z \leftarrow (Z/K)\sum_k \tilde{w}_{r,k}$.
        Resample the particles, for example using multinomial resampling, $x_{r,k} \sim \sum_{k'} \tilde{W}_{r,k'}\delta_{\tilde{x}_{r,k'}}(\cdot)$, but we recommend more advanced scheme such as stratified resampling ***(18) [[add reference]]***.
        for $k \in \{1, \ldots, K\}$ **do**
            Reset particle weights: $w_{r,k} = 1$.
        end for
    else
        No resampling is needed: $w_{r,k} = \tilde{w}_{r,k}$; $x_{r,k} = \tilde{x}_{r,k}$ for all $k \in \{1, \ldots, K\}$.
    end if
end for

Measuring particle degeneracy using (conditional) Effective Sample Size (ESS)

(19) [[say we will be doing conditional ESS]]

Both adaptive methods rely on being able to assess the quality of a particle approximation. For completeness, we provide some background in this section on the classical notation of Effective Sample Size (ESS) and of conditional ESS, a recent generalization which we use here ***(20) [[cite]]***. The notion of ESS in the context of importance sampling (IS) or SMC is distinct from the notion of ESS in the context of MCMC. The two are related in the sense of expressing a inflation of variance compared to an idealized Monte Carlo scheme but differ in the details. We will assume from now on that ESS refers to the SMC context.

The fundamental motivation of (relative) ESS stems from the analysis of the error of Monte Carlo estimators. Recall that for a given function of interest f (think for example as f being an indicator function on a clade),

$$\int \pi_r(\mathrm{d}x) f(x) \approx \frac{1}{K} \sum_{k=1}^{K} W_{r,k} f(x_{r,k}) =: \hat{I}.$$

The quantity on the right hand side is a random variable (with respect to the randomness of the SMC algorithm), $\hat{I}$ and we can think about it as an estimator of the deterministic quantity on the left hand side, I. Moreover the right hand side is a real-value random variable so we can define its mean square error, which can be further decomposed as a variance term and a squared bias term. For SMC algorithms, the variance term dominates as the number of particles goes to infinity ***(21) [[cite]]***. For this reason, we are interested in estimates of the variance of $\hat{I}$ across SMC random seeds, $\mathrm{Var}_{\mathrm{SMC}}[\hat{I}]$. However, the variance of $\hat{I}$ depends on the choice of function f, which is problem dependent, and we would like to remove this dependency. The first step is to consider a notion of relative variance, comparing to the variance we would obtain from a basic Monte Carlo scheme $\hat{I}^*$ relying on iid exact samples $x_1^\star, \ldots, x_K^\star \sim \pi$, $\mathrm{Var}_{\mathrm{MC}}[\hat{I}^*] = \mathrm{Var}[\frac{1}{K}\sum_k f(x_k^\star)]$. To make further progress, we will make approximations of the ratio $\mathrm{Var}_{\mathrm{MC}}[\hat{I}^*]/\mathrm{Var}_{\mathrm{SMC}}[\hat{I}]$.

To understand these approximations, let us start with a simplified version of Algorithm 2, where the function NextAnnealingParameter returns one. In this setting, no resampling occurs, and the algorithm reduces to an importance sampling algorithm (more specifically, it reduces to a single iteration of the Annealed Importance Sampling (AIS) algorithm ***(22) [[Cite]]***). Importance sampling is easier to analyse since the individual particles are independent and identically distributed, allowing us to do summarize the behaviour based on one particle, say $k = 1$. If we assume further that $\gamma_i = \pi_i$, i.e. that the normalization constant is one, then a classical argument by ***(23) [[Kong]]*** based on the Delta method yields

$$\begin{aligned}\frac{\mathrm{Var}_{\mathrm{MC}}[\hat{I}^*]}{\mathrm{Var}_{\mathrm{SMC}}[\hat{I}]} = \frac{\mathrm{Var}_{\mathrm{MC}}[\hat{I}^*]}{\mathrm{Var}_{\pi_0}[\hat{I}]} &\approx \frac{1}{1 + \mathrm{Var}_{\pi_0}[\tilde{w}_{1,1}]}, \\ &= \frac{1}{\mathbb{E}_{\pi_0}\left[(\tilde{w}_{1,1})^2\right]},\end{aligned}$$

where we used the fact that in this simple setting the distribution of one proposed particles is just π_0, so $\mathrm{Var}_{\mathrm{SMC}}[\cdot] = \mathrm{Var}_{\pi_0}[\cdot]$ and in the last line,

$$\mathbb{E}_{\pi_0}[\tilde{w}_{1,1}] = \int \pi_0(x_{0,1}) \frac{\pi_1(x_{0,1})}{\pi_0(x_{0,1})} \,\mathrm{d}x_{0,1} = 1.$$

(24) [[introduce general estimator of $\hat{Z}_r/Z_{r-1}$ earlier]]

In general, the normalization constant is not one, so for a general one-step AIS algorithm we get instead the approximation:

$$\begin{aligned}\frac{\mathrm{Var}_{\mathrm{MC}}[\hat{I}^*]}{\mathrm{Var}_{\mathrm{SMC}}[\hat{I}]} &\approx \left(\mathbb{E}_{\pi_0}\left[\left(\frac{\pi_1(x_{0,1})}{\pi_0(x_{0,1})}\right)^2\right]\right)^{-1} \\ &= \left(\mathbb{E}_{\pi_0}\left[\left(\frac{\gamma_1(x_{0,1})/Z_1}{\gamma_0(x_{0,1})/Z_0}\right)^2\right]\right)^{-1} \\ &= \left(\frac{Z_1}{Z_0}\right)^2 \Big/ \mathbb{E}_{\pi_0}\left[\left(\frac{\gamma_1}{\gamma_0}(x_{0,1})\right)^2\right].\end{aligned}$$

In the context of a general SMC algorithm, $\pi_1 = \pi_r$ plays the role of the current iteration, and π_0, of the previous iteration. However, since π_0 is not known in this case, we plug-in a particle approximation $\hat{\pi}_0 = \sum_{k=1}^{K} W_{0,k}\delta_{x_{0,k}}$ to get:

$$\frac{\mathrm{Var}_{\mathrm{MC}}[\hat{I}^*]}{\mathrm{Var}_{\mathrm{SMC}}[\hat{I}]} \approx \left(\frac{Z_1}{Z_0}\right)^2 \Big/ \mathbb{E}_{\hat{\pi}_0}\left[\left(\frac{\gamma_1}{\gamma_0}(x_{0,1})\right)^2\right] = \left(\frac{Z_1}{Z_0}\right)^2 \Big/ \sum_{k=1}^{K} W_{0,k}\left(\frac{\gamma_1}{\gamma_0}(x_{0,k})\right)^2.$$

The effect of this additional approximation is that it makes our estimator over-optimistic, by ignoring the error of the approximation $\hat{\pi}_0$ of π_0. It is nonetheless a useful tool to assess the degradation of performance over a small number of SMC iterations, which is what we will use ESS for.

Finally, since the ratio of normalization constants is also unknown, we also need to estimate it via Equation ***(25) [[TODO]]***, obtaining:

$$\frac{\mathrm{Var}_{\mathrm{MC}}[\hat{I}^*]}{\mathrm{Var}_{\mathrm{SMC}}[\hat{I}]} \approx \left(\sum_{k=1}^{K} W_{0,k}\frac{\gamma_1}{\gamma_0}(x_{0,k})\right)^2 \Big/ \sum_{k=1}^{K} W_{0,k}\left(\frac{\gamma_1}{\gamma_0}(x_{0,k})\right)^2. \tag{10}$$

More generally, to assess the effect on the particle population $(x_\cdot, w_\cdot)$ when performing a weight update $w'_k \leftarrow w_k u_k$, we define the relative conditional ESS:

$$\mathrm{rCESS}(W_\cdot, u_\cdot) = \left(\sum_{k=1}^{K} W_k u_k\right)^2 \Big/ \sum_{k=1}^{K} W_k u_k^2,$$

which is a normalized version of the CESS defined in ***(26) [[TODO]]***.

Dynamic resampling

As explained in Section , the construction of the proposal guarantees that as the difference $\phi_r - \phi_{r-1}$ goes to zero, the fluctuation of the weights vanishes. In this context, resampling at every iteration is wasteful. Fortunately, SMC algorithms can be modified to forgo a subset of the

resampling steps. From a theoretical stand-point, this is achieved by "grouping" the SMC proposals when they are not separated by a resampling round (and grouping similarly the intermediate distributions π_r). For example, to resample every other round, use a transformed SMC algorithm with proposal $K'_{r/2}(x_r, (x_{r+1}, x_{r+2})) = K_{r+1}(x_r, x_{r+1})K_{r+2}(x_{r+1}, x_{r+2})$, for each even r. For convenience, this can be implemented as an algorithm over R iterations instead of $R/2$, with two modifications: first, when resampling is skipped, we multiply the weights. This is implemented in Lines 13 and 23 of Algorithm 2. Second, we only use the weights corresponding to resampling rounds in the estimate of the normalization constant (Equation 5). This is implemented in Lines 17 and 20 of Algorithm 2.

Instead of specifying in advance the subset of iterations in which resampling should be performed, it is customary in the SMC literature ***(27) [[add cite]]*** to determine whether to resample or not in an adaptive fashion. To do so, the standard approach is to compute a measure of particle degeneracy at every iteration, and to perform resampling only when the particle degeneracy exceeds a pre-determined threshold.

The standard measure of particle degeneracy used for this purpose is called the relative Effective Sample Size (ESS), defined as:

$$\mathrm{rESS}(\tilde{W}_{r,\cdot}) = \left(K \sum_{k=1}^{K} \tilde{W}_{r,k}^2 \right)^{-1} . \tag{11}$$

The above formula can be shown to be a special case of Equation 10 as follows. Let r^* denote the iteration of the latest resampling round preceding the current iteration r. This implies

$W_{r^*,k} = 1/K$ for all k. Plugging in the weight update $u_k = \tilde{w}_{r,k}$ into Equation 10, we obtain

$$\begin{aligned}
\mathrm{rCESS}(W_{r^*,\cdot}, \tilde{w}_{r,\cdot}) &= \left(\sum_{k=1}^{K} \frac{1}{K}\tilde{w}_{r,k}\right)^2 \Big/ \sum_{k=1}^{K} \frac{1}{K}\tilde{w}_{r,k}^2 \\
&= \frac{1}{K} \frac{\left(\sum_{k=1}^{K} \tilde{w}_{r,k}\right)^2}{\sum_{k=1}^{K} \tilde{w}_{r,k}^2} \\
&= \left(K \sum_{k=1}^{K} \tilde{W}_{r,k}^2\right)^{-1}.
\end{aligned}$$

Adaptive determination of annealing parameters

In practice it is difficult and inconvenient to manually construct a sequence of annealing parameters (ϕ_r). Not only the number of distributions R to get a certain accuracy may depend on the number of taxa, the number of sites, and the complexity of the evolutionary model, but also the optimal spacing between consecutive annealing parameters is in general non-regular. To alleviate this, in the following we borrow an adaptive strategy from the Approximate Bayesian Computation literature ***(28) [[ABC SMC paper]]***, also generalized to Bayesian model selection in ***(29) [[model selection paper]]***.

To understand the adaptive annealing scheme, observe first in the SMC algorithm presented in the last section, the weight update, Line 12, depends on x_{r-1} only (whereas in general SMC algorithms, the weight update could depend on both x_{r-1} and x_r; here it does not because of cancellation explained in Appendix ***(30) [[TODO]]***). As a consequence, we can swap the order of Lines 11 and 12. In other words, we can compute weights before sampling from the proposal. So instead of computing the weights only for one pre-determined annealing parameter ϕ_r, we can search over several tentative values. For each tentative value, we can score the choice using a measure of weight degeneracy applied to the putative weights. Crucially, each choice can be quickly scored without having to propose particles, which is key since proposals are typically the

computational bottleneck: in a phylogenetic context, the cost of one proposal step scales linearly in the number of sites whereas the search over ϕ_t proposed in this section has a running time constant in the number of sites and taxa. This is because the search involves fixed values of $p(y|x_{r-1,k})$ cached from the last proposal step, which are exponentiated to different values.

We show the procedure NextAnnealingParameter in Algorithm ***(31) [[TODO]]***. It is again based on relative conditional ESS, but this time, we are interested in the degeneracy of a single iteration, i.e. we do not trace back until the previous resampling step (since the optimization over the temperature difference can only impact the current iteration). As a corollary, the previous iteration's particles are not always equally weighted and the simplification in Equation 11 is not possible.

(32) [[TODO: move to appendix the alternative scheme]]

,

Algorithm 3 Procedure NextAnnealingParameter

1: **Inputs:** (a) Particle population from previous SMC iteration $(x_{r-1,\cdot}, w_{r-1,\cdot})$; (b) Annealing parameter ϕ_{r-1} of previous SMC iteration; (c) Global algorithmic tuning parameters: (i) Whether to use the relESS or relCESS criterion; (ii) A degeneracy decay target $\alpha \in (0, 1)$.
2: **Outputs:** automatic choice of annealing parameter ϕ_r.
3: Initialize the function f assessing the particle population quality associated to a putative annealing parameter ϕ:

$$f(\phi) = \text{relativeCESS}\left(W_{r-1,\cdot}, p(y|x_{r-1,\cdot})^{\phi-\phi_{r-1}}\right) = \frac{\left(\sum_{k=1}^{K} W_{r-1,k} p(y|x_{r-1,k})^{\phi-\phi_{r-1}}\right)^2}{\sum_{j=1}^{K} W_{r-1,j} p(y|x_{r-1,k})^{2(\phi-\phi_{r-1})}}$$

4: **if** $f(1) \geq \alpha f(\phi_{r-1}) = \alpha$ **then**
5: return $\phi_r = 1$
6: **else**
7: return $\phi^* \in (\phi_{r-1}, 1)$ such that $f(\phi^*) = \alpha f(\phi_{r-1}) = \alpha$. Such ϕ^* exists since $f(\phi_{r-1}) > 0$ and $\alpha \in (0, 1)$, so the continuous function $f(\phi) - \alpha f(\phi_{r-1})$ is positive at the left hand point of the search interval and negative at the right end point.
8: **end if**

The parameter α encodes the decay in particle population quality that we are aiming for. Based on our experiments we recommend values very close to one. For this reason, we

reparameterize the parameter α into $\beta = -\log_{10}(1-\alpha)$ and recommend a default value of $\beta = 5$ as a reasonable starting point. Increasing β improves the approximation accuracy.

- note: $f(\phi_{r-1}) \geq 0$, and for relCESS, $f(\phi_{r-1}) = 1$. - line search - pegasus solver

The distributions $\{\pi_r\}_{1,\cdots,R}$ are defined on a common measurable space $(E, \mathcal{E})$. The SMC sampler is a generalization of the standard SMC method (Doucet et al. 2001), in which the target distribution exists on a space of strictly increasing dimension. There are various ways of defining the sequence distributions. For instance, π_r is the posterior distribution of a parameter x given the data collected until time r, i.e. $\pi_r(x) = p(x|y_{1:r})$.

In this paper, we define the sequence distribution $\{\pi_r\}_{1,\cdots,R}$ as in simulated annealing (Neal 1996), and we call this SMC sampler an annealed SMC. The corresponding sequence of unnormalized distributions are denoted by $\{\gamma_r\}_{1,\cdots,R}$. This annealed SMC can be obtained by defining a sequence of distributions that admit the distribution of interest, $\pi_r(x_r)$, as the marginal of the recent iteration

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

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. We sample K particles at iteration r,

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

where K_r is a Markov kernel defined on $E \times \mathcal{E}$, with associated density $K_r(x_{r-1,k}, x_{r,k})$. The resulting sampler has a weight update

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

which is different from the one in a standard SMC.

Algorithm 4 summarizes the annealed SMC. A list of intermediate distributions $\pi_{1:R}$, is

Algorithm 4 An Annealed SMC

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

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

    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

introduced , each π_r is a tempered posterior distribution. At each iteration r, we first proposed the particles $x_{r,1:K}$ through the forward kernel K_{r-1} conditional on $\tilde{x}_{r-1,1:K}$. Then we compute the weights $w_{r,k}$ for each particle to compensate the discrepancy between the forward kernel $K_r(\tilde{x}_{r-1,k}, \cdot)$ and the intermediate target distribution π_r. Finally, resampling method is used to prune those particles with small weights. 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

$$L_{r-1}(x_r, x_{r-1}) = \frac{\pi_r(x_{r-1})K_r(x_{r-1}, x_r)}{\pi_r(x_r)}. \tag{12}$$

With this backward kernel, the incremental importance weight becomes

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

which doesn't involve particles at iteration r.

Phylogenetic MCMC Kernels for the Annealed SMC

The annealed SMC described in the previous subsection provides 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 subsection, we propose to use the standard MCMC kernels for Bayesian phylogenetics within an annealed SMC. The idea of the proposed SMC is to design a sequence of artificial intermediate distributions that goes from a tractable (easy-to-sample) distribution π_0 (e.g. prior distribution) to a distribution of interest, π_R. Each SMC iteration uses an MCMC kernel to propose artificially intermediate states, which are full trees.

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 a sequence of the artificially intermediate distributions

$$\pi_r(x) \propto \mathbb{P}(\mathcal{Y}|x)^{\phi_r} p(x), \tag{13}$$

where $p(x)$ is the prior density for x, and $\phi_0 = 0 \leq \phi_1 < \cdots < \phi_R = 1$. It is easy to see that the first distribution $\pi_0(x) = p(x)$, and $\pi_R(x) = \pi(x) = p(x|\mathcal{Y})$. In the prior distribution, we use the uniform distribution for the tree topology and an exponential distribution with rate 10.0 for the branch lengths.

We will use the annealed SMC (Algorithm 4) with the backward kernel in Equation (16), With this backward kernel, the incremental importance weight becomes $\gamma_r(x_{r-1})/\gamma_{r-1}(x_{r-1})$. More precisely, using Equation (13), we have

$$\gamma_r(x_{r-1})/\gamma_{r-1}(x_{r-1}) = \{\mathbb{P}(\mathcal{Y}|x_{r-1})\}^{\Delta_r},$$

where $\Delta_r = \phi_r - \phi_{r-1}$.

A common choice for the Markov kernels, $K_r(x_{r-1}, \cdot)$, is to use MCMC kernels (Del Moral et al. 2006, 2007). An MH kernel, a typical MCMC kernel, consists 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. 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, x_{r-1} remains.

In phylogenetics, there is a rich literature on using MCMC algorithms to sample the posterior phylogenetic trees. 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 (Tierney 1994; Andrieu et al. 2003): if each of the transition kernels $\{K^i\}, i = 1, \cdots, M$, has the 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 summarizes the annealed SMC for phylogenetics where the proposal K_r^i can be any MCMC kernel, including those proposed in Bayesian phylogenetics literature (Larget and Simon 1999; Lakner et al. 2008; Li et al. 2000; Holder and Lewis 2003). The proposal distributions used in this paper are defined in Appendix. Figure 1 provides an overview of the annealed SMC for phylogenetic trees algorithmic framework.

Temperature scheduling

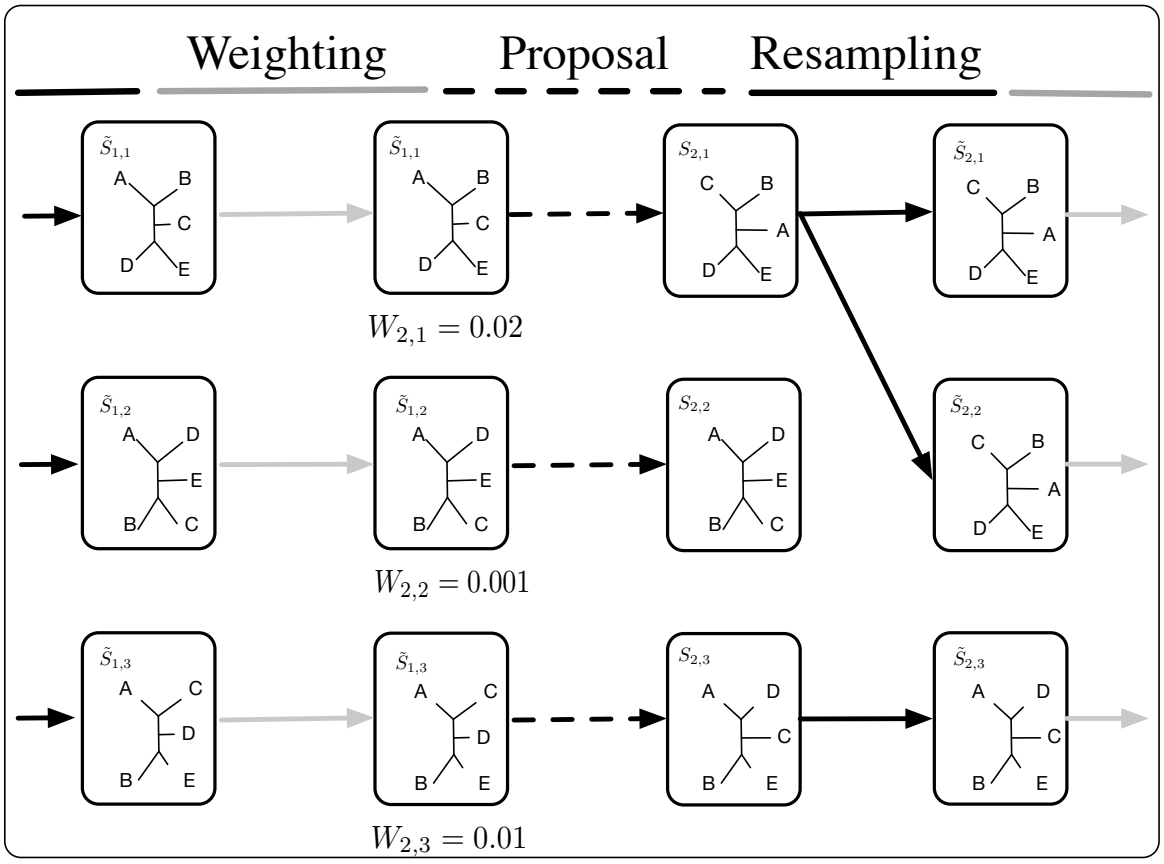


Figure 1: An overview of the annealed SMC algorithmic framework for phylogenetic trees. The algorithm iterates the following three steps: (i) compute the weights using samples from the previous iteration, (ii) perform MCMC moves to propose new samples, and (iii) resample from the weighted samples to obtain an unweighted set of samples.

Algorithm 5 An annealed SMC for phylogenetic trees

1: $x_{1,k} \leftarrow \perp$, $\forall k \in \{1, \cdots, K\}$
2: $w_{1,k} \leftarrow 1/K$
3: **for** $r \in 2, \ldots, R$ **do**
4: Sample $x_{r,k} \sim \sum_{i=1}^{M} p_i K_r^i(x_{r-1,k}, \cdot)$, $\sum_{i=1}^{M} p_i = 1$
5: $w_{r,k} \leftarrow \{\mathbb{P}(\mathcal{Y}|x_{r-1,k})\}^{\phi_r - \phi_{r-1}}$
6: Normalize weights $W_{r,k} \propto w_{r,k}$, and resample $\{x_{r,k}, W_{r,k}\}$ if $\text{ESS}_r < \epsilon$, where $\epsilon \in [1, K]$. (ESS will be introduced in the next section *Temperature scheduling.)*
7: **end for**

The sequence of the artificially intermediate distributions in Equation (13) is determined by the choice of the temperature scheduling, $\{\phi_r\}$, which relies on choosing the successive temperature difference Δ_i. Ideally, the sequence of the intermediate distributions changes gradually from the prior distribution to the posterior distribution so that the propagated particles from the current iteration can well approximate the next intermediate distribution.

A simple choice for the temperature sequence is to use a deterministic temperature schedule (Friel and Pettitt 2008). For instance, we choose $\phi_r = (r/R)^3$, where R is the total number of SMC iterations. In this case, the difference between successive temperatures is $\Delta_r = (3r^2 - 3r + 1)/R^3$. An annealed SMC with a larger number of R is computationally more expensive but has a better performance. However, it is hard to determine the total number of SMC iterations. Therefore, we focus on the adaptive schemes for temperature scheduling.

Adaptive scheme based on ESS

The effective sample size (ESS) (Del Moral et al. 2012) at time r is

$$\mathrm{ESS}_r = \frac{1}{\sum_{k=1}^{K}\left(\frac{W_{r-1,k}w_{r,k}}{\sum_{j=1}^{K} W_{r-1,j}w_{r,j}}\right)^2} = \frac{(\sum_{k=1}^{K} W_{r-1,k}w_{r,k})^2}{\sum_{j=1}^{K} W_{r-1,j}^2 w_{r,j}^2},$$

which takes values between 1 and K. ESS_r represents the number of perfect samples we are approximating π_r. A high ESS value is a necessary condition for good SMC approximation. If we choose Δ_r that is too large, then with high probability most of the particles will have very small or zero weights, which will lead to low ESS and collapse of the annealed SMC algorithm. A smaller Δ_r can help improve the performance of algorithm, but the computational cost is higher, the particles may move too slowly to the target distribution.

Inspired by Del Moral et al. (2012), we aim to control the ESS over iterations by selecting the differences of successive temperatures Δ_r such that

$$\mathrm{ESS}_r(\Delta_r) = \alpha \mathrm{ESS}_{r-1},$$

where $0 < \alpha < 1$, and it is close to 1 (for example, 0.999). The advantage of this adaptive scheme is that we can automatically determine the temperatures to prevent the algorithm from being collapsed. Note that $w_{r,k} = \{\mathbb{P}(\mathcal{Y}|x_{r-1,k})\}^{\Delta_r}$, where $\Delta_r = \phi_r - \phi_{r-1}$. Since $\text{ESS}_r(\Delta_r)$ does not involve the particles at time r, we can use the bisection method to find an approximate solution for Δ_r.

Adaptive scheme based on Conditional ESS

If the resampling step is not conducted at iteration $r-1$, the ESS is not able to reflect the discrepancy between two successive intermediate distributions π_{r-1} and π_r. Zhou et al. (2016) propose to use the conditional ESS (CESS) to measure the discrepancy. The CESS can be written in the following form

$$\text{CESS}_r = \left[\sum_{k=1}^{K} KW_{r-1,k}\left(\frac{w_{r,k}}{\sum_{k=1}^{K} KW_{r-1,k}w_{r,k}}\right)^2\right]^{-1} = \frac{K(\sum_{k=1}^{K} W_{r-1,k}w_{r,k})^2}{\sum_{k=1}^{K} W_{r-1,k}(w_{r,k})^2}.$$

With a fixed α, we solve the equation $\text{CESS}_r = \alpha$ numerically to obtain a solution for Δ_r. Note that the CESS will be equal to the ESS when resampling is conducted at every iteration. We introduce a new notation $\beta = -\log_{10}(1-\alpha)$ to ease presentation, a larger value of β refers to an α value closer to 1. More detailed comparison of the adaptive scheme using ESS and CESS is described in Appendix.

Normalizing Constant

Estimate from SMC

A byproduct of the SMC algorithm is an estimate of the normalizing constant $\mathbb{P}(\mathcal{Y})$ in Equation (1). We denote the normalizing constant by Z for simplicity. An estimate of the normalizing constant Z is

$$\widehat{Z_{R,K}} = \prod_{r=1}^{R}\left(\frac{1}{K}\sum_{k=1}^{K} w_{r,k}\right) = \prod_{r=1}^{R}\left(\frac{1}{K}\sum_{k=1}^{K}\{\mathbb{P}(\mathcal{Y}|x_{r-1,k})\}^{\phi_r-\phi_{r-1}}\right). \tag{14}$$

If resampling is not conducted at each iteration r, an alternative form is provided by

$$\widehat{Z_{R,K}} = \prod_{j=1}^{t_{R-1}+1} \left(\sum_{k=1}^{K} W_{n_j-1,k} \prod_{m=n_{j-1}+1}^{n_j} \{\mathbb{P}(\mathcal{Y}|x_{m-1,k})\}^{\phi_m-\phi_{m-1}} \right), \tag{15}$$

where n_j is the SMC iteration index at which we do the jth resampling, t_{R-1} is the number of resampling steps between 1 and $R-1$. Moreover, when the temperature scheme is deterministic, Equation (14) and Equation (15) provide unbiased estimators of the marginal likelihood $\mathbb{P}(\mathcal{Y})$ (Del Moral 2004).

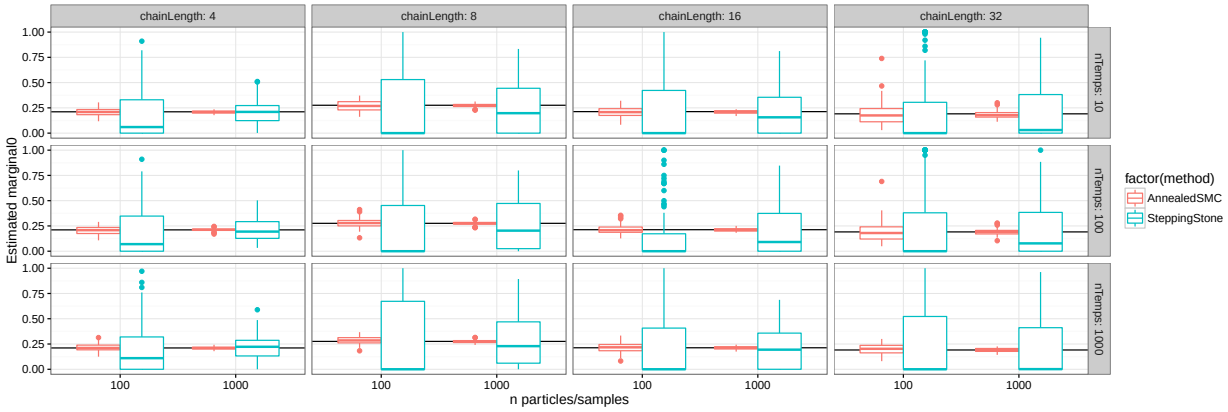
Simulation Studies

Simulation setup and tree distance

We evaluate the proposed annealed SMC using some simulation studies. In order to simulate datasets, we first generate a set of random unrooted trees, including topology and branch lengths, as the reference trees. The tree topology is sampled from a uniform distribution (Desper and Gascuel 2004). Each branch length is generated from an exponential distribution with rate 10.0.

Then, for each reference tree, we simulate DNA sequences using the K2P model with parameter $\kappa = 2.0$. We choose an arbitrary point on the simulated reference tree as its root (the model is reversible). The data generation starts from the root of a tree by randomly sampling from the stationary distribution of the CTMC. Assuming site independence, we generate the data for the children of the root using the transition probability computed with the rate matrix Q. This procedure is recursively implemented until reaching the leaves. We discard the data at the internal nodes and take the data on leaves as the simulated observations.

We summarize the sample of phylogenetic trees from the annealed SMC and MCMC using the *majority-rule consensus tree* which consists of clades that are present in no less than a



half of the trees (Felsenstein 1981). Then we measure the distance between a reference tree and an estimated consensus tree using three types of tree distance metrics: Robinson Foulds (RF) metric (Robinson and Foulds 1981), the partition metric (PM) (Felsenstein 2003), and Kuhner Felsenstein (KF) metric (Kuhner and Felsenstein 1994). A small tree distance between an estimated consensus tree and its reference tree indicates good performance of the estimation method.

An example for stepping stone in Hidden Markov Model

In this section, we use an example of hidden Markov model to illustrate that the marginalized likelihood estimates provided by stepping stone are biased. The methodology we considered to estimate the marginalized likelihood are Annealed SMC algorithm and stepping stone. **(Add description of experiment here.)**

Comparison of normalizing constant estimates $\hat{Z}$

In this section, we emphasize the marginal likelihood estimates provided by adaptive annealed SMC (ASMC), debiased adaptive annealed SMC (DASMC), deterministic annealed SMC (DSMC), LIS and SS. In DASMC, the temperature scheme is determined before running

annealed SMC using the same temperatures obtained from the ASMC. In DSMC, we use temperature scheme $\phi_r = (r/R)^3$ with a predetermined R.

In the first experiment, we focus on evaluating the marginal likelihood estimates using ASMC, DASMC, LIS and SS with same computing budget. We simulate unrooted trees of various numbers of taxa: 5, 10, 15, 20, and 25. For each tree, we generate one data set of DNA sequences and each sequence has length 100. Every algorithm for each data set is repeated 100 times with different random seeds. We set $\beta = 5$ for adaptive annealed SMC, and the number of particles is set to 1000. In stepping stone and linked importance sampling, we set the total number of heated chains $D = 50$, and the temperature scheme is $\phi_d = (d/D)^3$, where $d = 1, 2, \ldots, D$. We set $K_{SMC}R_{SMC} = N_{SS}D_{SS} = N_{LIS}D_{LIS}$ in order to make a fair comparison.

Figure 2 shows the comparison of the performance of the four algorithms in terms of the normalizing constants in the log scale as the number of taxa increases. For the unbiased estimators of the marginal likelihood, it is underestimated in the log scale by Jensen's inequality. Therefore, a higher value of the marginalized likelihood implies a more accurate estimate. The results show that ASMC and DASMC can achieve higher marginalized likelihood estimates in the log scale than SS and LIS with the same computational cost. The performances of the two SMC algorithms are quite similar, while the marginal likelihood estimates provided by LIS and SS are close.

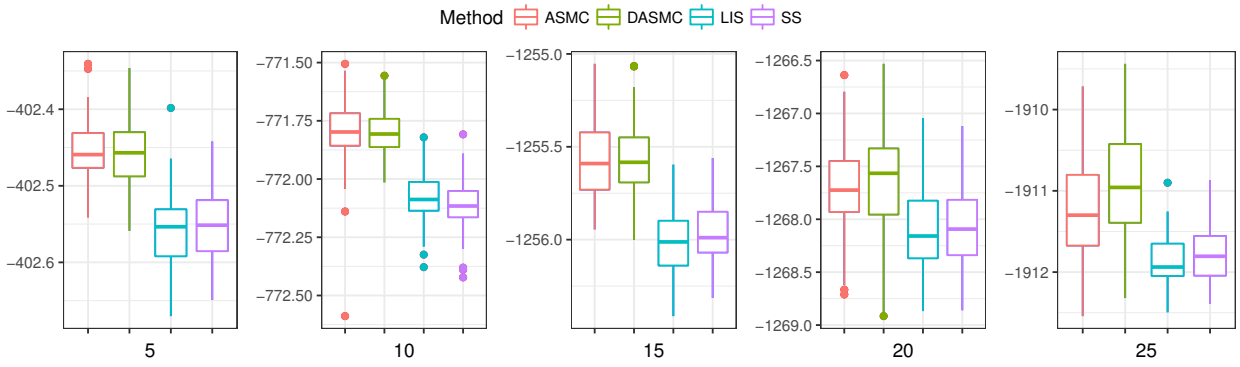


Figure 2: Normalizing constant (in log scale) for different number of taxa with fixed computational budget.

Another experiment is conducted to measure the variability of the marginal likelihood estimates from each algorithm by comparing the coefficient of variation (CV) for different number of taxa with the same setting. The coefficients of variation is defined as $CV = sd(\hat{Z})/E(\hat{Z})$. We simulate 10 data sets for each number of taxa from 10 to 40. For each data set, we repeat each algorithm 20 times with different random seeds. The upper bound of CV equals $\sqrt{n-1}$, where n represents the number of replicates in experiments. In our setting, this upper bound is $\sqrt{20-1} \approx 4.359$. In ASMC, the computational cost is fixed at $K = 200$, and $\beta = 6$. In DSMC, we use the same number of particles, and the temperature scheme is $\phi_r = (r/R)^3$, where the total number of temperature R is fixed to be the one obtained from running ASMC with $K = 200$ and $\beta = 6$ for a tree with 10 taxa.

Figure 3 displays the CV for ASMC and DSMC as a function of the number of taxa. The error bars in figures represent the 95% confidence intervals. When the number of taxa is smaller than 20, there is no much difference between the CV of two SMC algorithms. However, the CV of DSMC increases faster than ASMC as the number of taxa gets larger than 25. It gradually converge to the upper bound of CV as number of taxa reaches 40. The CV of ASMC increases slowly as the number of taxa increases.

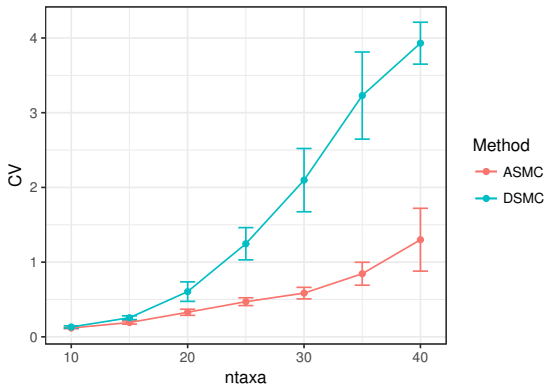


Figure 3: Coefficient of variation (CV) for the marginal likelihood estimates versus the number of taxa for ASMC and DSMC with fixed number of particles.

Comparison of tree distance metrics

In this section, we compare the performance of ASMC and MCMC in terms of reconstructing the phylogenetic trees using the log likelihood of the consensus tree and the tree distance between the true tree and the consensus tree. We simulate one unrooted tree, the true tree, with 50 taxa and then generate one data set of DNA sequences of length 2000 from this tree. The ASMC is run with $\beta = 6$ and $K = 100$. The MCMC algorithm is initialized with a random tree from the prior distribution. To make a fair comparison, we set the number of MCMC iterations to be no less than $K_{SMC}R_{SMC}$. Table 1 summarizes the iteration numbers, the log likelihood of the consensus tree and tree distance metrics from running ASMC and MCMC. Although the computational cost of MCMC is set to about twice as expensive as ASMC, the log-likelihood of the consensus tree is much higher than that from MCMC. In addition, ASMC can achieve much lower RF and KF metrics. Further, to confirm that both the ASMC and MCMC can converge to the same posterior distribution, MCMC is rerun with a better starting value, the consensus tree obtained after running ASMC. This run of MCMC is denoted as MCMC2 in Table 1. The computational cost of MCMC2 is set the same as the ASMC algorithm. This time MCMC can achieve similar log-likelihood of the consensus tree and tree distance metrics compared with ASMC, which indicates that MCMC may be trapped at a local mode and can only converge to the true posterior with a good initialization.

Influence of Number of Threads, β, and K

The speed and performance of the ASMC can be affected by the number of threads, β, and K used in the algorithm. In this section, we focus on investigating the effects of these factors on ASMC. We simulate an unrooted tree with 30 taxa, and then generate DNA sequences of length 1500.

We first use an example to show one advantage of using the ASMC algorithm over MCMC algorithm. Figure 4 displays the computing time versus different number of threads. We run ASMC 100 times using $K = 1000$ and $\beta = 2$ for each number of threads. The results indicate

Method	R	K	Metric	Value
ASMC	54876	100	ConsensusLogLL	-72787.99
	54876	100	BestSampledLogLL	-72826.17
	54876	100	PartitionMetric	0
	54876	100	RobinsonFouldsMetric	0.70623
	54876	100	KuhnerFelsenstein	0.00990
MCMC	1.0E+07		ConsensusLogLL	-72833.82
	1.0E+07		PartitionMetric	0
	1.0E+07		RobinsonFouldsMetric	0.92031
	1.0E+07		KuhnerFelsenstein	0.03138
MCMC2	5.49E+06		ConsensusLogLL	-72784.86
	5.49E+06		PartitionMetric	0
	5.49E+06		RobinsonFouldsMetric	0.73644
	5.49E+06		KuhnerFelsenstein	0.01066

Table 1: Comparison of tree distance using SMC and MCMC.

that by increasing the number of cores, the speed of the ASMC algorithm can be increased notably. In our experiments, the propagation step in the ASMC algorithm is paralleled.

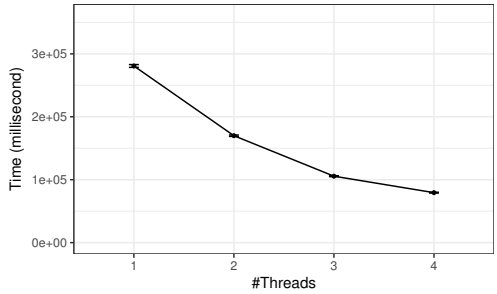


Figure 4: Computing time of ASMC using multiple threads.

In Figure 5, we compare the performance of ASMC algorithm as a function of K, with β fixed at 5. We choose four different particle values $K = 100, 300, 1000, 3000$. Both the marginal likelihood estimate and tree metrics improved when we increase K. Figure 6 displays the performance of ASMC algorithm as a function of β, with $K = 1000$. We select five distinct β values, $\beta = 3, 4, 4.3, 5, 5.3$. The marginal likelihood estimates and tree metrics can be improved when β increases, they tend to be stable when β reaches 5. A larger value of β can improve the performance of ASMC more significantly than K.

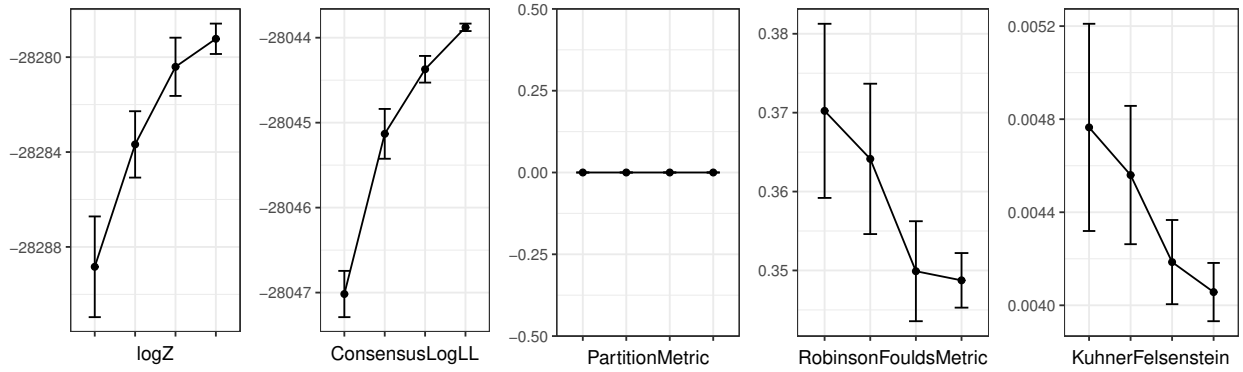


Figure 5: Comparison of adaptive SMC algorithm with different number of particles, from left to right $K = 100, 300, 1000, 3000$.

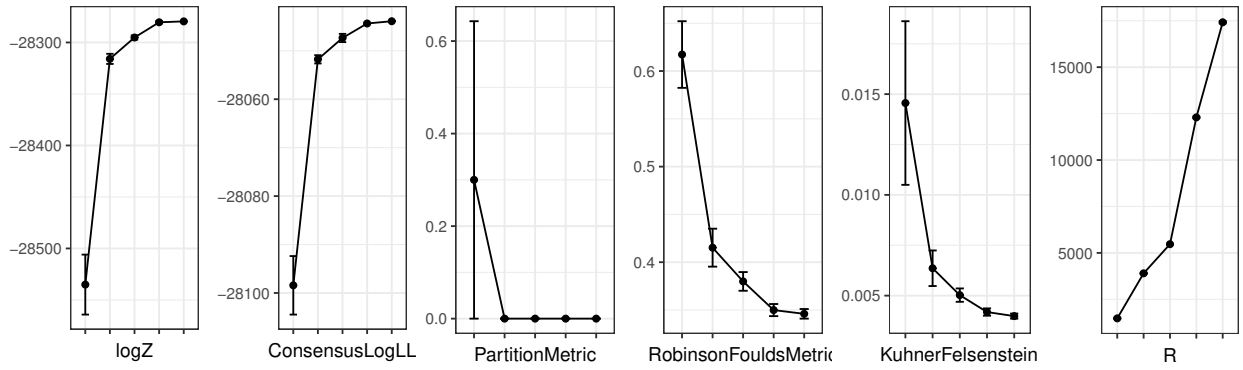


Figure 6: Comparison of adaptive SMC algorithms with different β, from left to right $\beta = 3, 4, 4.3, 5, 5.3$. Here R is the total number of SMC iterations.

Real datasets

We analyze two difficult real data sets from TreeBASE: M336 and M1809 in Table 1 of Lakner et al. (2008). M336 contains DNA sequences of length 1949 for 27 species. In M1809, there are 59 species and the length of DNA sequence is equal to 1824. We compare the marginal likelihood estimates, log-likelihood of the consensus tree, and tree distance metrics provided by ASMC and MrBayes (default setting) with the same computational cost. The reference trees used to compute tree distance is obtained by running MrBayes for very long time. Hence, the convergence of MCMC is assumed to be guaranteed. Note that the comparison of ASMC and MrBayes is not totally fair since the tree moves in MrBayes are more complicated and advanced. The evolutionary model we consider in real data analysis is the JC69 model.

Dataset M336

We set $K = 500$ and $\beta = 5.3$ for the ASMC algorithm. The log normalizing constant estimated from ASMC is -7103.73, which is higher than the log normalizing constant provided by MrBayes using stepping stone (-7114.04). Table 2 displays the log-likelihood of the consensus tree and tree distance metrics provided by ASMC and MrBayes. In the table, R represents the number of temperatures in ASMC and the total number of MCMC iterations in MrBayes respectively. The consensus log-likelihood estimated from ASMC is slightly lower than MrBayes. The RF and KF metrics estimated from MrBayes are slightly higher than ASMC. The Majority rule consensus tree provided by ASMC and MrBayes are identical, which coincides with the reference tree.

Dataset M1809

We set $K = 1000$ and $\beta = 5$ for the ASMC algorithm. The log marginal likelihood estimated from ASMC is -37542.25, the one estimated by MrBayes using stepping stone is -37335.73. Table 3 displays the tree metrics provided by ASMC and MrBayes. The Consensus log-likelihood and PM metric provided by ASMC is higher than MrBayes, and RF, KF metrics

Method	R	K	Metric	Value
ASMC	15706	500	ConsensusLogLL	-6892.16
	15706	500	BestSampledLogLL	-6901.31
	15706	500	PartitionMetric	0
	15706	500	RobinsonFouldsMetric	0.01269
	15706	500	KuhnerFelsenstein	5.55E-06
MrBayes	8.0E+06		ConsensusLogLL	-6889.52
	8.0E+06		PartitionMetric	0
	8.0E+06		RobinsonFouldsMetric	0.01832
	8.0E+06		KuhnerFelsenstein	2.25E-5

Table 2: Comparison of running ASMC and MrBayes for M336 from TreeBASE.

estimated from ASMC is lower.

Method	R	K	Metric	Value
ASMC	17639	1000	ConsensusLogLL	-36972.513
	17639	1000	BestSampledLogLL	-36991.443
	17639	1000	PartitionMetric	2.0
	17639	1000	RobinsonFouldsMetric	0.13741
	17639	1000	KuhnerFelsenstein	3.95E-4
MrBayes	1.76E+07		ConsensusLogLL	-36996.13
	1.76E+07		PartitionMetric	16.0
	1.76E+07		RobinsonFouldsMetric	0.513285
	1.76E+07		KuhnerFelsenstein	0.01137

Table 3: Comparison of running ASMC and MrBayes for M1809 from TreeBASE.

Conclusion

We have proposed an annealed SMC algorithm with the adaptive temperature scheduling based on the conditional ESS for Bayesian phylogenetics.

The estimation of the marginal likelihood is an important but challenging task in Bayesian phylogenetics. One of the advantages of SMC algorithms is that the estimate of the marginal likelihood can be obtained for free. Further, the estimate is unbiased when the temperature scheduling is deterministic. Our simulation studies have shown that ASMC can give a similar

marginal likelihood estimate as the one obtained from the SMC with the same but deterministic sequence of temperatures. We have also investigated the marginal likelihood estimates using methods LIS and SS. With the same computing budget, ASMC has been demonstrated to result in more accurate estimates. Moreover, the ASMC algorithm requires less tuning than the other methods. All of the DSMC, LIS, and SS need a predetermined sequence of temperatures, which is often inconvenient to choose in practice. In addition, LIS and SS demands extra and expensive computation for the marginal likelihood estimates, and it is nontrivial to decide the number of iterations. In contrast, ASMC only requires choosing the number of particles and the value of β. With the same setting, ASMC leads to a more stable estimate for the marginal likelihood than the other three methods when the problem gets more complicated, for example, when the number of taxa increases.

As we have illustrated in our simulation studies, the parallelism of SMC is one advantage over MCMC algorithms. Even though the parallelism can also be achieved in some parallel tempering algorithms (e.g. parallel Metropolis coupled MCMC Altekar et al. (2004)), there exists difference between the parallelism of SMC and parallel tempering. The computing time decreases as the number of cores increases in annealed SMC, it is well-adaptive to the parallel structure. However, in parallel tempering, more swaps are required between the main chain and the most heated chain as we increase the number of chains, gains from parallelism will decrease.

MCMC imposes relatively strict constraints on the types of proposals that can be used. More precisely, to alleviate the problem of high rejection rate, only small moves are allowed in proposals, making it challenge to design fast mixing algorithms. In future, it is desirable to design more bold MCMC moves that are more suitable for annealed SMC. For example, we can use the automatic specification of distributions within SMC algorithms (Zhou et al. 2016) to improve the MCMC tree moves in the annealed SMC.

FUNDING

ACKNOWLEDGMENTS

*

References

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

Andrieu, C., N. de Freitas, A. Doucet, and M. I. Jordan. 2003. An introduction to MCMC for machine learning. Machine Learning 50:5–43.

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

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

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

Chan, H. P. and T. L. Lai. 2013. A general theory of particle filters in hidden Markov models and some applications. The Annals of Statistics 41:2877–2904.

Del Moral, P. 2004. Feynman-Kac Formulae: Genealogical and Interacting Particle Systems with Applications. Springer, New York.

Del Moral, P., A. Doucet, and A. Jasra. 2006. Sequential Monte Carlo samplers. Journal of the Royal Statistical Society B 68:411–436.

Del Moral, P., A. Doucet, and A. Jasra. 2007. Sequential Monte Carlo for Bayesian computation. Bayesian Statistics 8:1–34.

Del Moral, P., A. Doucet, and A. Jasra. 2012. An adaptive sequential Monte Carlo method for approximate Bayesian computation. Statistics and Computing 22:1009–1020.

Desper, R. and O. Gascuel. 2004. Theoretical foundation of the balanced minimum evolution method of phylogenetic inference and its relationship to weighted least-squares tree fitting. Molecular Biology and Evolution 21:587–598.

Dinh, V., A. E. Darling, I. Matsen, and A. Frederick. 2016. Online Bayesian phylogenetic inference: theoretical foundations via Sequential Monte Carlo. Systematic biology .

Doucet, A., N. de Freitas, and N. Gordon. 2001. Sequential Monte Carlo methods in practice. Springer-Verlag, New York.

Drummond, A. and A. Rambaut. 2007. Beast: Bayesian evolutionary analysis by sampling trees. BMC Evolutionary Biology 7:214.

Drummond, A. and M. Suchard. 2010. Bayesian random local clocks, or one rate to rule them all. BMC biology 8:114.

Everitt, R. G., R. Culliford, F. Medina-Aguayo, and D. J. Wilson. 2016. Sequential Bayesian inference for mixture models and the coalescent using sequential Monte Carlo samplers with transformations. arXiv preprint arXiv:1612.06468 .

Fan, Y., R. Wu, M.-H. Chen, L. Kuo, and P. O. Lewis. 2010. Choosing among partition models in bayesian phylogenetics. Molecular biology and evolution 28:523–532.

Felsenstein, J. 1981. Evolutionary trees from DNA sequences: a maximum likelihood approach. J. Mol. Evol. 17:368–376.

Felsenstein, J. 2003. Inferring phylogenies. Sinauer Associates.

Fourment, M., B. C. Claywell, V. Dinh, C. McCoy, I. Matsen, A. Frederick, and A. E. Darling. 2017. Effective online Bayesian phylogenetics via sequential Monte Carlo with guided proposals. Systematic biology .

Friel, N. and A. N. Pettitt. 2008. Marginal likelihood estimation via power posteriors. Journal of the Royal Statistical Society: Series B (Statistical Methodology) 70:589–607.

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

Görür, D. and Y. W. Teh. 2009. An efficient sequential Monte Carlo algorithm for coalescent clustering. *in* Advances in Neural Information Processing Systems (NIPS).

Höhna, S., M. Defoin-Platel, and A. Drummond. 2008. Clock-constrained tree proposal operators in Bayesian phylogenetic inference. Pages 1–7 *in* 8th IEEE International Conference on BioInformatics and BioEngineering.

Höhna, S. and A. J. Drummond. 2012. Guided tree topology proposals for Bayesian phylogenetic inference. Syst. Biol. 61:1–11.

Holder, M. and P. Lewis. 2003. Phylogeny estimation: Traditional and Bayesian approaches. Nat. Rev.: Genet. 4:275–284.

Huelsenbeck, J. P. and F. Ronquist. 2001. MRBAYES: Bayesian inference of phylogenetic trees. Bioinformatics 17:754–755.

Jow, H., C. Hudelot, M. Rattray, and P. G. Higgs. 2002. Bayesian phylogenetics using an RNA substitution model applied to early mammalian evolution. Mol. Biol. Evol. 19:1591–1601.

Jukes, T. H., C. R. Cantor, et al. 1969. Evolution of protein molecules. Mammalian protein metabolism 3:132.

Kimura, M. 1980. A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. J. Mol. Evol. 16:111–120.

Kuhner, M. K. and J. Felsenstein. 1994. A simulation comparison of phylogeny algorithms under equal and unequal evolutionary rates. Mol. Biol. Evol. 11:459–468.

Lakner, C., P. van der Mark, J. P. Huelsenbeck, B. Larget, and F. Ronquist. 2008. Efficiency of Markov chain Monte Carlo tree proposals in Bayesian phylogenetics. Syst. Biol. 57:86–103.

Larget, B. and D. Simon. 1999. Markov chain Monte Carlo algorithms for the Bayesian analysis of phylogenetic trees. Mol. Biol. Evol. 16:750–759.

Lemey, P., A. Rambaut, A. J. Drummond, and M. A. Suchard. 2009. Bayesian phylogeography finds its roots. PLoS Computational Biology 5:e1000520.

Li, S., D. Pearl, and H. Doss. 2000. Phylogenetic tree construction using Markov chain Monte Carlo. J. Am. Stat. Assoc. 95:493–508.

Mau, B., M. Newton, and B. Larget. 1999. Bayesian phylogenetic inference via Markov chain Monte Carlo. Biometrics 55:1–12.

Miller, J. C. and C. J. Maloney. 1963. Systematic Mistake Analysis of Digital Computer Programs. Commun. ACM 6:58–63.

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

Neal, R. M. 2005. Estimating ratios of normalizing constants using linked importance sampling. arXiv preprint math/0511216 .

Olsson, J. and R. Douc. 2017. Numerically stable online estimation of variance in particle filters. arXiv:1701.01001 [stat] ArXiv: 1701.01001.

Rannala, B. and Z. Yang. 1996. Probability distribution of molecular evolutionary trees: a new method of phylogenetic inference. J. Mol. E 43:304–311.

Rannala, B. and Z. Yang. 2003. Bayes estimation of species divergence times and ancestral population sizes using DNA sequences from multiple loci. Genetics 164:1645–1656.

Robinson, D. and L. Foulds. 1981. Comparison of phylogenetic trees. Mathematical Biosciences 53:131–147.

Ronquist, F. and J. P. Huelsenbeck. 2003a. MrBayes 3: Bayesian phylogenetic inference under mixed models. Bioinformatics 19:1572–1574.

Ronquist, F. and J. P. Huelsenbeck. 2003b. Mrbayes 3: Bayesian phylogenetic inference under mixed models. Bioinformatics 19:1572–1574.

Ronquist, F., M. Teslenko, P. van der Mark, D. L. Ayres, A. Darling, S. Hohna, B. Larget, L. Liu, M. A. Suchard, and J. P. Huelsenbeck. 2012. MrBayes 3.2: Efficient Bayesian phylogenetic inference and model choice across a large model space. Syst. Biol. 61:539–542.

Smith, R. A., E. L. Ionides, and A. A. King. 2017. Infectious disease dynamics inferred from genetic data via sequential Monte Carlo. Molecular Biology and Evolution Page msx124.

Suchard, M. A. and B. D. Redelings. 2006. BAli-Phy: simultaneous Bayesian inference of alignment and phylogeny. Bioinformatics 22:2047–2048.

Teh, Y. W., H. Daumé III, and D. M. Roy. 2008. Bayesian agglomerative clustering with coalescents. *in* Advances in Neural Information Processing Systems (NIPS).

Thorne, J. L., H. Kishino, and I. S. Painter. 1998. Estimating the rate of evolution of the rate of molecular evolution. Mol. Biol. Evol. 15:1647–1657.

Tierney, L. 1994. Markov chains for exploring posterior distributions. Annals of Statistics 22:1701–1762.

Wang, L., A. Bouchard-Côté, and A. Doucet. 2015. Bayesian phylogenetic inference using a combinatorial sequential Monte Carlo method. Journal of the American Statistical Association 110:1362–1374.

Wingate, D., N. Goodman, A. Stuhlmueller, and J. M. Siskind. 2011. Nonstandard Interpretations of Probabilistic Programs for Efficient Inference. Pages 1152–1160 *in* Advances in Neural Information Processing Systems 24 (J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, eds.). Curran Associates, Inc.

Xie, W., P. O. Lewis, Y. Fan, L. Kuo, and M.-H. Chen. 2010. Improving marginal likelihood estimation for Bayesian phylogenetic model selection. Systematic biology 60:150–160.

Yang, Z. and B. Rannala. 1997. Bayesian phylogenetic inference using DNA sequences: a Markov chain Monte Carlo method. Mol. Biol. Evol. 14:717–724.

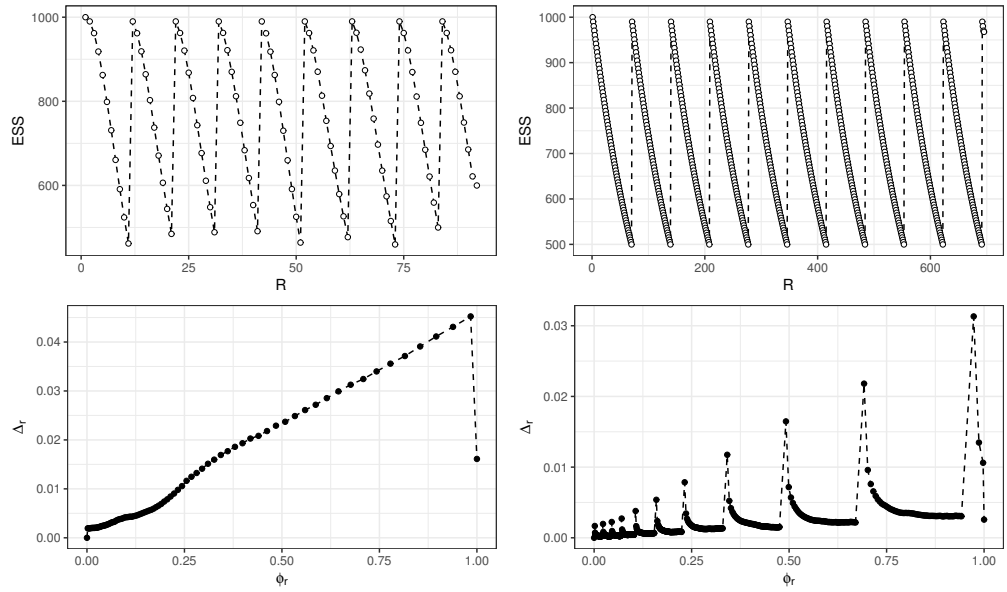
Zhou, Y., A. M. Johansen, and J. A. Aston. 2016. Toward automatic model comparison: An adaptive sequential Monte Carlo approach. Journal of Computational and Graphical Statistics 25:701–726.

SUPPLEMENTARY

Comparison of ESS and CESS

The ESS can reflect the discrepancy between the proposal distribution and the intermediate target distribution since last resampling time. However, if resampling is not conducted at every iteration, the ESS may not be able to show the mismatch in adaptive scheme annealed SMC. The CESS proposed in (Zhou et al. 2016) can improve the performance of adaptive SMC compared with the ESS. Two experiments are used to illustrate the benefits of using CESS. In the first experiment, we simulate one unrooted tree of 10 taxa, and generate one data set of DNA sequences and each sequence has length 100. The setup of tree simulation is same as section *Simulation Studies*. We run adaptive annealed SMC algorithm in two schemes: (a) $CESS_r = 0.99$; (b) $ESS_r(\Delta_r) = 0.99 ESS_{r-1}$. We use $K = 1000$ particles. Resampling of particles is triggered when $ESS < 0.5K$. Figure 7 displays the advantage of using CESS over ESS in adaptive SMC. The temperature difference Δ_r increases smoothly in the CESS scheme, while in the ESS scheme there is a big gap in temperature increment after doing resampling, then the temperatures decrease gradually until the next resampling time. The number of iterations R for adaptive SMC using ESS is much larger than using CESS.

In our second experiment, we compare the performance of adaptive annealed SMC using CESS and ESS in terms of tree metrics and normalizing constant. We simulate one unrooted tree of 15 taxa, and generate one data set of DNA sequences and each sequence has length 200. The setup of tree simulation is same as section *Simulation Studies*. We repeat adaptive annealed SMC algorithm 20 times with $CESS_r = 0.999$ and $ESS_r(\Delta_r) = 0.978 ESS_{r-1}$ respectively. The number of particles is $K = 500$. Under this setting, the computational cost for two schemes is quite close, as for same number of particles the number of temperatures (R) for CESS scheme and ESS scheme is close. (see Figure 8). Figure 8 displays the normalizing constant estimates, consensus likelihood and tree metrics provided by adaptive SMC using CESS and ESS. The log normalizing constant estimates and log consensus likelihood provided by adaptive SMC using CESS is higher and admits smaller variation. The PF, RF and KF metrics provided by two schemes are quite close, while the metrics provided by CESS scheme admits smaller variation.



(a) CESS (b) ESS

Figure 7: Comparison of CESS and ESS in terms of Δ_r as a function of ϕ_t and ESS as a function of R_t.

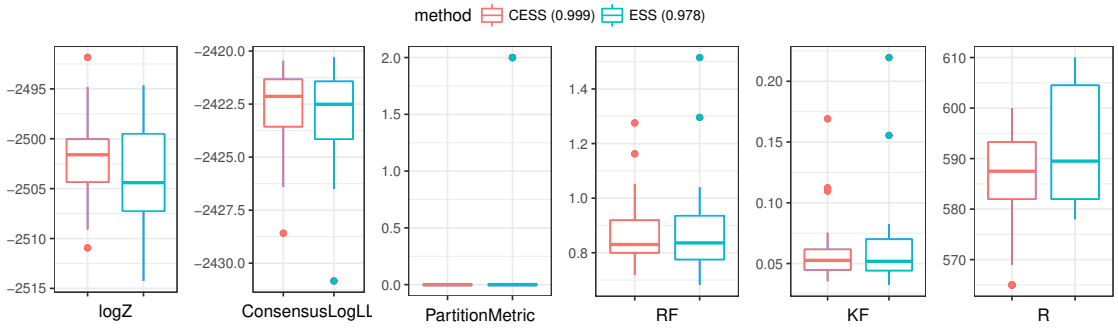


Figure 8: Comparison of adaptive SMC using ESS and CESS

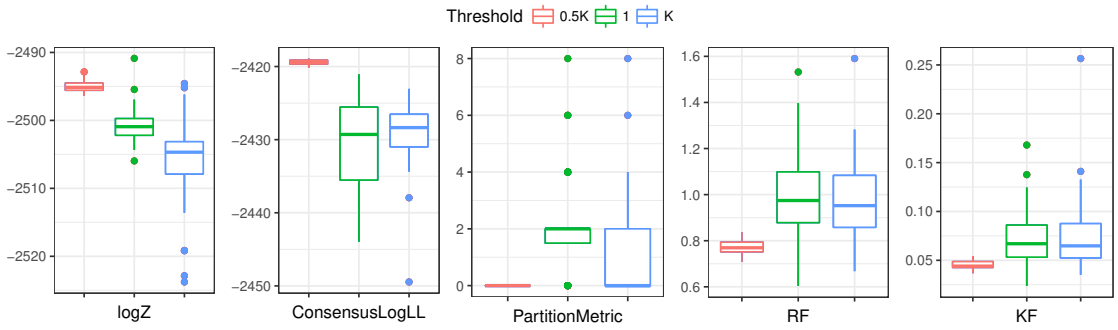


Figure 9: Comparison of three resampling thresholds $0.5K$, 1 and K.

Comparison of Resampling Strategies

We use an experiment to compare the performance of adaptive annealed SMC with three different Resampling thresholds. The first resampling threshold is $\epsilon_1 = 1$. In this case, we never resample our particles. The second resampling threshold is $\epsilon_2 = K$. At every SMC iteration, we prune particles with small weights. The third resampling threshold is $\epsilon_3 = 0.5K$. This is the threshold we used in our numerical experiments. We simulate one unrooted tree of 15 taxa, and generate one data set of DNA sequences and each sequence has length 200. The setup of tree simulation is same as section *Simulation Studies*. We run adaptive annealed SMC algorithm 20 times with the three resampling thesholds described above. We set $CESS_r = 0.99999$. The number of particles is $K = 100$. Figure 9 displays the advantage of resampling triggered by $ESS = 0.5K$ over NEVER perform resampling and resampling at every SMC iteration. The normalizing constant estimates, consensus likelihood and tree metrics provided by adaptive SMC using three different ϵ are displayed in Figure 9. The log normalizing constant estimates and log consensus likelihood provided by adaptive SMC using $\epsilon_3 = 0.5K$ is higher and admits smaller variation. The PF, RF and KF metrics provided by adaptive SMC using $\epsilon_3 = 0.5K$ are lowest.

MCMC proposals for Bayesian Phylogenetics

In this paper, we used the proposals q_r^i defined as follow:

1. q_r^1: the *multiplicative branch proposal*.

   This 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) Lakner et al. (2008).

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

3. q_r^3: the *stochastic NNI proposal*. We consider the nearest neighbor interchange (NNI) (Jow et al. 2002) to propose a new tree topology.

4. q_r^4: the *stochastic NNI proposal with resampling the edge* that uses the above NNI proposal in (3) and the multiplicative branch proposal in (1) for the edge under consideration.

5. q_r^5: the *Subtree Prune and Regraft (SPR) move* that selects and removes a subtree from the main tree and reinserts it elsewhere on the main tree to create a new tree.

Note that here we only describe the MCMC kernels for phylogenetic trees. For estimating evolutionary parameters θ, we just need to use $\{q_r^i\}$ to propose θ.

Backward kernel construction for annealed SMC

The annealed SMC has a weight update function

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

which is different from the one in a standard SMC. A common approach 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

$$L_{r-1}(x_r, x_{r-1}) = \frac{\pi_r(x_{r-1})K_r(x_{r-1}, x_r)}{\pi_r(x_r)}. \tag{16}$$

With this backward kernel, the incremental importance weight becomes

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

which doesn't involve particles at iteration r. This weight update function also doesn't involve pointwise evaluation of the proposal $K_r(x_{r-1,k}, x_{r,k})$. This is different from standard SMC algorithm.

Estimates of normalizing constants from LIS

We described the LIS procedure as follows:

1. Sample an index v_0 randomly from $\{1, 2, \cdots, N\}$, and sample $x_{0,v_1} \sim \pi_0(\cdot)$.

2. For $d = 0, 1, \cdots, D$, sample N states from π_d as follows:

   (a) If $d > 0$: sample an index v_d from $\{1, 2, \cdots, N\}$, and set $x_{d,v_d} = x_{d-1*d}$.

   (b) For $k = v_d + 1, \cdots, N$, sample $x_{d,k}$ from the forward kernel $x_{d,k} \sim K_d(x_{d,k-1}, \cdot)$.

   (c) For $k = v_d - 1, \cdots, 1$, sample $x_{d,k}$ from the backward kernel $x_{d,k} \sim L_d(x_{d,k+1}, \cdot)$.

   (d) If $d < D$, sample μ_d from $\{1, 2, \cdots, N_d\}$ according to the following probabilities:

$$p(\mu_d | x_d) = \frac{\gamma_{d-1*d}(x_{d,\mu_d})}{\gamma_d(x_{d,\mu_d})} \Big/ \sum_{k=1}^{N_d} \frac{\gamma_{d-1*d}(x_{d,k})}{\gamma_d(x_{d,k})},$$

and set x_{d*d+1} to x_{d,μ_d}.

3. Compute the likelihood estimate

$$\hat{Z}_{LIS} = \prod_{d=1}^{D} \left[\frac{1}{N} \sum_{k=1}^{N} \frac{\gamma_{d-1*d}(x_{d-1,k})}{\gamma_{d-1}(x_{d-1,k})} \Big/ \frac{1}{N} \sum_{k=1}^{N} \frac{\gamma_{d-1*d}(x_{d,k})}{\gamma_{d}(x_{d,k})} \right].$$

Note that if the backward kernel is reversible, then the forward kernel is the same as backward kernel. In this paper, we use the MCMC kernel as backward and forward kernels in LIS.