VaiPhy: a Variational Inference Based Algorithm for Phylogeny

Anonymous Author(s)
Affiliation
Address
email

Abstract

Phylogenetics is a classical methodology in computational biology that today has become highly relevant for medical investigation of single-cell data, e.g., in the context of development of cancer. The exponential size of the tree space is unfortunately a formidable obstacle for current Bayesian phylogenetic inference using Markov chain Monte Carlo based methods since these rely on local operations. And although more recent variational inference (VI) based methods offer speed improvements, they rely on expensive auto-differentiation operations for learning the variational parameters. We propose VaiPhy, a remarkably fast VI based algorithm for approximate posterior inference in an *augmented tree space*. VaiPhy produces marginal log-likelihood estimates on par with the state-of-the-art methods on real data, and is considerably faster since it does not require auto-differentiation. Instead, VaiPhy combines coordinate ascent update equations with two novel sampling schemes: (i) *SLANTIS*, a proposal distribution for tree topologies in the augmented tree space, and (ii) the *JC sampler*, the, to the best of our knowledge, first ever scheme for sampling branch lengths directly from the popular Jukes-Cantor model. We compare VaiPhy in terms of density estimation and runtime. Additionally, we evaluate the reproducibility of the baselines. We provide our code on GitHub: `gitlink`.

1 Introduction

Phylogenetic software has for a longer time been applied in biological research and biological findings relying on phylogenetic trees are frequent. Moreover, due to the emergence of single-cell sequencing, phylogentic inference for bifurcating and multifurcating trees can now be utilized in medical studies of how cells differentiate during development and how tumor cells progress in cancer.

When attempting to infer the posterior distribution over phylogenetic trees for $|X|$ given taxa, represented as sequences, the main obstacle is the exponential size of the tree space, i.e., the set of candidate trees having those taxa as leaves. In Markov chain Monte Carlo (MCMC) based Bayesian phylogenetic inference, e.g., MrBayes [21] and RevBayes [17], this tree space is explored by performing local operations on the trees. In order to avoid local operations, Bouchard-Cote et al. [3] launched Sequential Monte Carlo (SMC) as an alternative methodology for Bayesian phylogenetic inference.

Probabilistic phylogenetic inference is a machine learning problem and the key to unlock this crucial application can be expected to be found among the machine learning methodology; in particular, considering the capacity of the VI methodology, [23, 1], to deliver impressive performance gains for Bayesian inference, e.g., compared to MCMC. Expectedly, such gains have recently been made within the realm of phylogenetic analysis.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

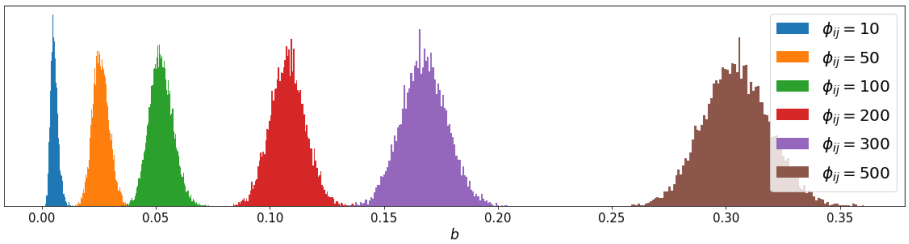

Figure 1: Visualization of branch lengths, b, sampled directly from the Jukes-Cantor model. The figure shows samples for different numbers of estimated mutations along an edge, $\phi_{i,j}$, when the total number of sites are $M = 2000$.

For instance, inference based on the computationally costly CAT model, which allows different equilibrium frequencies across sequence sites, have been made substantially more efficient by application of VI, [5]. Also [10] applied VI to render analysis of a sequence evolution model more efficient. They consider the efficiency improvements obtained by VI, in particular using the generic STAN software [10], when analysing a single fixed tree topology. Interestingly, the VI approach can also be used to analyse a large number of candidate phylogenetic trees. In [28], so-called subsplit Bayesian networks (SBNs) were introduced in order to represent the posterior over a given set of candidate trees. This approaches has subsequently been developed further by introducing an edge (a.k.a. branch) length distribution [29], and making the edge length distribution more expressive by taking advantage of normalizing flows [27]. In particular, [29] obtains posterior distributions that improve on the marginal likelihood obtained by using MrBayes, [18], in combination with the improved Stepping Stone (SS) method, [8].

The SBN approach evades the exponential sized tree space, by considering a given set of candidate trees and the partitions of the given taxa defined by those candidate trees. It is noteworthy that the VI-methodology earlier has been successfully applied to bipartite matching and a few other combinatorial problems [2]; the phylogeny problem, however, involves both combinatorial and continuous parameters.

Multifurcating trees are common in biology and biologists are well familiar with them. For instance, in a clone tree that represents the evolutionary relationships between the subclones of a tumor, nodes may have more than two children. Even in cases with no biological support for multifurcating trees, biologists are used to working with multifurcating trees. E.g., the consensus method, used to analyze the posterior of a MrBayes run, may produce a multifurcating tree.

Contributions: We present a novel framework for phylogenetic inference that produces marginal likelihood estimates on par with the state-of-the-art VI methods, while being up to 25 times faster than VBPI on real datasets. The framework is on the top level a mean-field-VI algorithm and is comprised of the following three novel methods:

- **SLANTIS** is a proposal distribution which proposes multifurcating spanning trees. Sampling from SLANTIS is performed following an approach suggested by [6], for sampling a perfect matching in bipartite graphs: it is fast, and the sampled tree's compensation weight is easily computed without additional cost of time.

- **The JC sampler** is a novel scheme for sampling branch lengths directly from the Jukes-Cantor (JC69) model. This is a significant contribution, as no previous sampling scheme is known for this task and the JC model arguably is a commonly used substitution model in phylogenetics.

- **VaiPhy** is a VI algorithm for obtaining posterior approximations over the set of branch lengths, the ancestral sequences and tree topologies, given the leaf set observations. VaiPhy utilizes SLANTIS and the JC sampler in order to compute the importance weighted evidence lower bound (IWELBO; [4]) and its update equations. Compared to existing VI methods in phylogenetics, VaiPhy extends the target distribution by considering an auxiliary variable, the ancestral sequences, and it does not require any input consisting of precomputed trees.

We demonstrate the flexibility of our framework by using its learned parameters to parameterize two new proposal distributions for the CSMC algorithm. We compare our CSMC with the other two main SMC-based ones, outperforming them in terms of marginal likelihood estimation on real data.

Finally, we contribute to the field of Bayesian phylogenetic inference by benchmarking the most prominent methods in the field on seven popular real datasets.

2 Background

In the Bayesian phylogenetic setting, we are interested in the posterior over phylogenies – pairs of tree topologies and sets of branch lengths – given observations on the leaves,

$$p_\theta(\tau, \mathcal{B}|X) = \frac{p_\theta(\tau, \mathcal{B}, X)}{p_\theta(X)}, \tag{1}$$

where τ, $\mathcal{B}$ and X denote the tree topology, the set of branch lengths and the observations, respectively. More specifically, $\mathcal{B}$ contains a single branch length per edge, e, in the set of edges in τ, i.e. $\mathcal{B} = \{b(e) : e \in E(\tau)\}$. Unfortunately, $p_\theta(\tau, \mathcal{B}|X)$ is intractable, so we need to approximate it using Bayesian inference. Before going into how to approximate this quantity, we first provide some crucial notation.

In phylogenetics, one is often concerned with bifurcating X-trees. These trees have labeled leaf nodes, and its internal vertices have either two (root in a rooted tree) or three degrees. Meanwhile, multifurcating trees' internal vertices can have higher degrees. We use M, Σ, and N to denote the sequence length, the set of nucleotides, and the number of number of vertices, respectively. Further, we let $|X|$ be the number of given taxa, V and E be the set of vertices and edges, while (V, E) is an acyclic graph.

Let Z_i^m be the probabilistic ancestral sequence at position m, and θ denote the model parameters, such as the branch length prior parameters or the parameter choices in the substitution model. Then, the evolutionary model along an edge, $e = (i, j)$, is $p_\theta(Z_i^m = \alpha | Z_j^m = \beta) = exp(b(e)\, Q_{\alpha\beta})$, where Q is the rate matrix, parameterized according to the JC model, and $\alpha, \beta \in \Sigma$ are nucleotides. The likelihood is defined as

$$p_\theta(X|\tau, \mathcal{B}) = \sum_{Z \in \Sigma^M} p_\theta(X, Z|\tau, \mathcal{B}),$$

and can be computed effectively using Felsenstein's pruning algorithm [9].

3 Proposed Framework

Existing Bayesian phylogentic approaches often perform inference in a sample space over bifurcating X-trees and branch length sets, Ω. In contrast, we propose to work in an augmented sample space, A. Firstly, we work in a tree topology space for multifurcating trees with labelled internal vertices. Secondly, we extend the target distribution in Eq. (1) by considering ancestral sequences on the internal vertices as an auxiliary variable, Z.

Apart from that our augmented space offers higher modelling flexibility (clearly, since $\Omega \subset \mathrm{A}$), it enables posterior estimation over Z, which further lets us estimate the number of mutations between all vertices in the sample space. This is a key insight which ultimately allows us to devise VaiPhy, a mean-field coordinate ascent VI [16] algorithm.

3.1 VaiPhy

In the augmented space, A, we wish to approximate the posterior distribution

$$p_\theta(\tau, \mathcal{B}, Z|X) \propto p_\theta(X, Z|\mathcal{B}, \tau) p_\theta(\mathcal{B}|\tau) p_\theta(\tau), \tag{2}$$

using a factorized variational distribution

$$q(\tau, \mathcal{B}, Z|X) = q(\mathcal{B}|X) q(\tau|X) q(Z|X), \tag{3}$$

where $q(Z|X) = \prod_{i \in I(\mathrm{A})} q(Z_i|X)$. Here, $I(\mathrm{A})$ denotes the set of internal vertices in the augmented space. We will use $q(Z|X)$ to compute $\phi \in [0, M]^{N \times N}$, the expected number of mutations over the edge (i, j). Notice that the maximum number of mutations between two vertices is the number of sites, M.

As is standard in classical VI, we alternatively update the distributions in Eq. (3) in order to maximize the evidence lower bound (ELBO),

$$\mathcal{L} = \mathbb{E}_{q(\tau, \mathcal{B}, Z|X)} \left[\log \frac{p_\theta(X, Z|\mathcal{B}, \tau) p_\theta(\mathcal{B}|\tau) p_\theta(\tau)}{q(\mathcal{B}|X) q(\tau|X) q(Z|X)} \right], \tag{4}$$

over iterations. Each iteration, we also update ϕ, following [11]

$$\phi_{ij} = \sum_{\alpha \neq \beta} F_{ij}^{\alpha,\beta}, \tag{5}$$

where $F_{ij}^{\alpha,\beta}$ is the estimated number of times a nucleotide mutates from α to β, defined as $F_{ij}^{\alpha,\beta} = \sum_{m=1}^{M} q(Z_i^m = \alpha|X^m) q(Z_j^m = \beta|X^m)$.

The update equations for the variational distributions are derived in a mean-field fashion. Starting with $q(Z_i|X)$, we find that

$$\begin{aligned} \log q^*(Z_i|X) &\stackrel{+}{=} \mathbb{E}_{q(\tau, \mathcal{B}, Z_{\neg i}|X)} \left[\log p_\theta(X, Z, \mathcal{B}, \tau) \right] \\ &\stackrel{+}{=} \mathbb{E}_{q(\tau, \mathcal{B}|X)} \left[\sum_{Y_j \in Y_{N_\tau(i)}} q(Y_j|X) \log p_\theta(Y_j|Z_i, b(i,j)) \right], \end{aligned} \tag{6}$$

where $\stackrel{+}{=}$ denotes equal up to an additive constant and $Y_{N_\tau(i)}$ is the set of sequences of node i's neighbours in τ. Continuing with the two other approximations, we have

$$\begin{aligned} \log q^*(\tau|X) &\stackrel{+}{=} \mathbb{E}_{q(\mathcal{B}, Z|X)} \left[\log p_\theta(X, Z, \mathcal{B}, \tau) \right] \\ &\stackrel{+}{=} \mathbb{E}_{q(\mathcal{B}|X)} \left[\sum_{\substack{(i,j) \in E(\tau) \\ Z_i, Z_j}} q(Z_i, Z_j|X) \log p_\theta(Z_i|Z_j, b(i,j)) \right] \end{aligned} \tag{7}$$

and

$$\begin{aligned} \log q^*(b(i,j)|X) &\stackrel{+}{=} \mathbb{E}_{q(\tau, \mathcal{B}_{\neg b(i,j)}, Z|X)} \left[\log p_\theta(X, Z, \mathcal{B}, \tau) \right] \\ &\stackrel{+}{=} \mathbb{E}_{q(\tau|X)} \left[\sum_{Z_i, Z_j} q(Z_i, Z_j|X) \log p_\theta(Z_i|Z_j, b(i,j), \tau) \right] + \log p_\theta(b(i,j)). \end{aligned} \tag{8}$$

Exact computation of the expectations in Eq. (6) and (8) above involves summing over all possible trees, which is computationally intractable. Luckily, we can easily approximate the expectation using importance sampling and our new proposal distribution, $s_\phi(\tau)$, is outlined in the subsequent Sec. 3.2. We direct the reader to Appendix A for the derivations of above variational distributions and the details of the algorithm.

To obtain good starting points for the variational distributions, we initialize ϕ with trees obtained using the Neighbor Joining algorithm [13]. The details of the initialization is provided in Appendix A.2. Algorithm 1 is a high-level algorithmic description of VaiPhy. The proof regarding the natural gradient is presented in Appendix H.

Algorithm 1 VaiPhy in pseudocode – ι is the total number of iterations and η is the natural gradient step size

Input: X, ϕ, η, ι
for iter $= 1, \ldots, \iota$ **do**
  Approximate $\log q^*(Z|X)$ using $s_\phi(\tau)$ {Eq. (6)}
  $q(Z_i) \leftarrow (1-\eta)q(Z_i) + \eta q^*(Z_i)$
  $\phi_{ij} \leftarrow \sum_{\alpha \neq \beta} F_{ij}^{\alpha,\beta}$ {Eq. (5)}
  $q(\tau|X) \leftarrow q^*(\tau|X)$ {Eq. (7)}
  $q(\mathcal{B}|X) \leftarrow q^*(\mathcal{B}|X)$ using $s_\phi(\tau)$ {Eq. (8)}
end for
return $q(\tau, \mathcal{B}, Z|X)$, ϕ

3.2 SLANTIS

We propose the method Sequential Look-Ahead Non-local Tree Importance Sampling (SLANTIS), which enables sampling of a tree, τ, from a distribution, $s_\phi(\tau)$. SLANTIS has support over multifurcating trees with labeled internal vertices, and is parameterized by ϕ. To describe the algorithm, we next introduce some notation.

Using ϕ, SLANTIS immediately constructs W, an $N \times N$ matrix describing all pairwise edge weights based on the substitution model,

$$W_{ij} = \sum_{\alpha,\beta:\alpha\neq\beta} \phi_{ij}\ (b(i,j)\ Q_{\alpha\beta}) + \sum_{\alpha,\beta:\alpha=\beta} (M - \phi_{ij})\ (b(i,j)\ Q_{\alpha\beta}). \tag{9}$$

Moreover, let $\mathcal{G}$ be a graph with vertex set $\{N\}$ with edge weights specified by W. Let $\mathcal{G}_\mathcal{I} = \mathcal{G} \backslash \Lambda$, where Λ is the (V, E) of the leaves, i.e., the graph induced by the internal vertices of our trees. The pseudo code of the algorithm is presented in Algorithm B. The main idea, which is borrowed from [6], is to order the edges of the graph and for each edge in the order make a choice whether to include it in the sampled tree or not. The choice is made based on a Bernoulli trial, where the two probabilities are proportional to the weight of the maximum spanning tree (MST) containing the already selected edges and the currently considered edge and that of the MST containing the already selected edges but not the currently considered edge. The action associated with the former outcome is to include the edge in the sampled tree and the other outcome is to not include it.

To facilitate fast computation of the required MSTs, we consider an order of the edges where first in the order of the edges of an MST T_1 of $\mathcal{G}$ occurs in some order. After those edge, the edges of an MST T_2 of $\mathcal{G} \setminus T_1$ can be found. In general, T_s is an MST of $\mathcal{G} \setminus \cup_{r=1}^{s-1} T_r$, and after the edges of $\cup_{r=1}^{s-1} T_r$, the edges of T_s can be found. In the algorithm, we repeatedly take advantage of the property that for any tree T and edge e not in T, $T \cup e$ contains a unique cycle and the removal of its heaviest edge different from e produces the minimum weight sub-tree of $T \cup e$ containing e. If the considered edge is in T and we choose to include it in the sampled tree, τ, we recursively update $\log s_\phi(\tau)$ by adding the Bernoulli log probability corresponding to the edge. See Fig. 2 for an illustration.

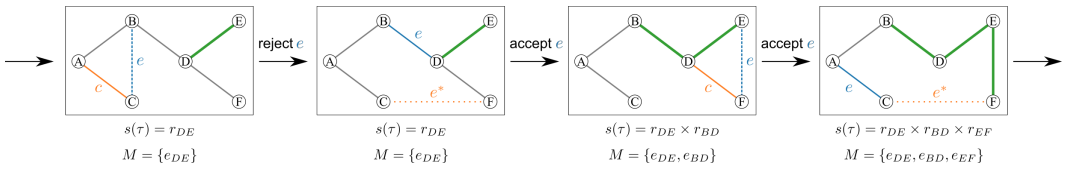

Figure 2: Propagation step of SLANTIS. Solid line is τ so far, bold **green** lines are accepted edges. Dashed and dotted lines are the edges not in τ. Blue color indicates the edge we consider (e) and orange color indicates the alternative edge; either c or e^*. $r_e = W(e)/(W(e) + W(c))$ or $r_e = W(e)/(W(e) + W(e^*))$. At the end of SLANTIS, the spanning tree with bold green lines is reported.

3.3 The JC Sampler

The JC model is arguably one of the most frequently used substitution models in Bayesian phylogenetics. However, as far as we are aware, the model has only been used for obtaining maximum likelihood estimates of branch lengths. Here, we observe that the JC model can be seen as a binomial distribution, and leverage the Beta-Binomial conjugacy for obtaining a normalized distribution over branch lengths in variational inference. Below we show how to sample from this distribution, and how to evaluate its likelihood using the change-of-variables formula.

Given a branch length $b(e) = b$, the probability of the m'th nucleotide *not* mutating along edge $e = (i, j)$ is, according to the JC model,

$$p = \left(\frac{1}{4} + \frac{3}{4}e^{-\frac{4}{3}b}\right)^{1\{Z_i^m = Z_j^m\}}. \tag{10}$$

As all mutations are equiprobable for this model, the probability of the nucleotide mutating is thus $1 - p$, giving rise to the following Binomial distribution when all M estimated mutations and non-mutations are observed and assuming mutations occur independently across sites

$$\text{Bin}\left(\phi_{ij}|p\right) = p^{M-\phi_{ij}}(1-p)^{\phi_{ij}}. \tag{11}$$

Assuming an uninformative Beta prior on the Binomial parameter, p, and recalling that Beta is a conjugate prior for the Binomial distribution, we get a posterior distribution over p, $\text{Beta}(p|M - \phi_{ij} + 1, \phi_{ij} + 1)$. Since we are ultimately interested in sampling b, not p, we find a transform, $f(\cdot)$, mapping p to b, using Eq. (10)

$$f^{-1}(b) = \frac{1}{4} + \frac{3}{4}e^{-\frac{4}{3}b} = p, \;\; f(p) = -\frac{3}{4}\log\left(\frac{4}{3}\left(p - \frac{1}{4}\right)\right) = b. \tag{12}$$

In all, sampling branch lengths from the JC model includes two straightforward steps, (i) draw p from the Beta posterior distribution above, and (ii) pass it through the transform, $b = f(p)$.

Of course, we are also interested in evaluating the likelihood of the sampled branch length. To do so, we use the change-of-variables formula, we let

$$s_\phi(b) = \text{Beta}(p|M - \phi_{ij} + 1, \phi_{ij} + 1)\left|\frac{df^{-1}(b)}{db}\right| \tag{13}$$

$$= \frac{p^{M-\phi_{ij}}(1-p)^{\phi_{ij}}e^{-\frac{4}{3}b}}{\text{B}(M - \phi_{ij} + 1, \phi_{ij} + 1)}, \tag{14}$$

where $\text{B}(\cdot)$ is the Beta function. This provides an efficient way of evaluating the likelihood of a branch length. In Algorithm 2 we summarize the JC sampler with an algorithmic description.

Algorithm 2 The JC Sampler

1: **Input:** ϕ, i, j
2: Sample $p \sim \text{Beta}(p|M - \phi_{ij} + 1, \phi_{ij} + 1)$
3: Transform $b = f(p)$ {Eq. (12)}
4: Evaluate $s_\phi(b)$ {Eq. (13)}
5: **return** $b, s_\phi(b)$

3.4 Assessing the Framework

The framework learns its parameters and distributions by maximizing the ELBO in Eq. (4). However, the IWELBO offers a tighter lower bound on the marginal log-likelihood. Using SLANTIS and the JC sampler we can easily evaluate our framework using the IWELBO:

$$\mathcal{L}_L = \mathbb{E}_{\mathcal{B}_\ell, \tau_\ell \sim s_\phi(\mathcal{B}, \tau)}\left[\log \frac{1}{L}\sum_{\ell=1}^{L}\frac{p_\theta(X, \mathcal{B}_\ell, \tau_\ell)}{s_\phi(\mathcal{B}_\ell, \tau_\ell)}\right], \tag{15}$$

where we may factorize the denominator as $s_\phi(\mathcal{B}, \tau) = s_\phi(\mathcal{B}|\tau)s_\phi(\tau)$, and $s_\phi(\mathcal{B}|\tau) = \prod_{e \in E(\tau)} s_\phi(b(e))$. Note that we have marginalized out the auxiliary variable, Z, in the target in Eq. (15), as we do not wish to compute $\mathcal{L}_L$ using our variational distributions.

4 Using VaiPhy to Parameterize the CSMC

As the tree topology space in A is considerably larger than the bifurcating X-tree space, obtaining competitive log-likelihood (LL) estimates is not straightforward. Namely, the uniform prior over tree the tree space in A, $p_\theta(\tau) = 1/N^{N-2}$, turns into a heavily penalizing term. When compared to baselines in smaller sample spaces, this will become a considerable negative factor for VaiPhy. C.f. the prior when considering unrooted, bifurcating X-trees (which is the case for VBPI): $p'_\theta(\tau) = 1/(2|X|-3)!!$, where $|X|$ is the number of taxa. As an example, for $|X| = 5$ and $N = 8$, $p_\theta(\tau)$ distributes mass uniformly over 262144 trees, while the latter one merely considers 15 trees.

Inspired by [20], we instead seek to use VaiPhy to parameterize a CSMC algorithm, thereby, obtaining a procedure for projecting our trees to the lower-dimensional leaf-labeled tree space. Here, the prior is instead $p''_\theta(\tau) = 1/(2|X|-5)!!$.

Since ϕ is not constrained to multifurcating trees, we can utilize the estimated number of mutations between vertices learned in A, also in the bifurcating X-tree space, Ω. Here arises an interesting obstacle. In the CSMC algorithm, the internal vertices are *unlabeled*. Hence we are effectively blindfolded, and need to estimate which entry in ϕ to use for a given edge. We achieve this by scoring edges based on how frequently the corresponding split of the leaf-set X occur in trees sampled from $s_\phi(\tau)$. See Appendix C.2 for more information.

At each rank $\rho \in [1, R]$, the CSMC proposes (i) a forest, $\mathcal{T}_\rho$, by *merging* the roots of two subtrees in $\mathcal{T}_{\rho-1}$, and (ii) a corresponding set of branch lengths, $\mathcal{B}_\rho$. This is done in parallel for K *particles*, resampling particles with probability proportional to their importance weights, w_ρ^k, at every rank[1]. However, the reason to take advantage of an CSMC is that it allows us to estimate the marginal log-likelihood in the bifurcating X-tree space, using the well-known formula,

$$\log p_\theta(X) \approx \log \prod_{\rho=1}^{R} \frac{1}{K} \sum_{k=1}^{K} w_\rho^k. \tag{16}$$

We let ϕ parameterize our two novel CSMC proposal distributions, $r_\phi(\mathcal{B}_\rho|\mathcal{B}_{\rho-1}, \mathcal{T}_\rho)$ and $r_\phi(\mathcal{T}_\rho|\mathcal{T}_{\rho-1})$. The former constructs a mixture of JC samplers, and evaluates the proposal's likelihood in a multiple importance sampling fashion [7], while the latter forms a categorical distribution over the roots in $\mathcal{T}_{\rho-1}$, parameterized by a function of the likelihoods of trees sampled by SLANTIS. The in-detail descriptions can be found in Appendix C.

5 Experiments

5.1 Visualizing the JC Sampler

We conduct a toy experiment, investigating how the estimated number of mutations along an edge affects the behaviour of the JC sampler. We would expect that a greater number of mutations should generate a larger branch length. This since the involved vertices are supposedly farther apart in terms of an evolutionary distance.

As demonstrated in Fig. 1, this advantageous behaviour is indeed displayed. As ϕ_{ij} grows, the corresponding JC sampling distribution is shifted towards larger branch lengths. Interestingly, the JC sampler distributes its mass more symmetrically than the log-Normal or Exponential distributions, which are typically used for modelling branch length distributions [24, 29, 27, 20].

It follows from Eq. (12) that branch lengths grow infinitely large as p goes to 0.25. This phenomenon in phylogenetics is referred to as saturation. Saturation occurs for distantly related lineages, and implies that the states of the nucleotides in these ancestral sequences are mutually independent. This is not an interesting case, hence the non-exhaustive parameter-search of ϕ_{ij} in Fig. 1.

5.2 Density Estimation

Here we benchmark our methods, VaiPhy and ϕ-CSMC, in terms of LL estimates on seven real world datasets, which we refer to as DS1-DS7 ([14, 12, 26, 15, 19, 30, 22]; in Appendix D we provide

[1]For more details and background of the CSMC algorithm, we point the reader to [24] and [20] for excellent descriptions.

Table 1: Real dataset LL estimates for the different methods. Our ϕ-CSMC is on average the best-performing CSMC method (red). Overall, MrBayes SS achieves the highest LL estimates (**bold**). *Not available due to unresolved memory issues.

DATA SET	(TAXA, SITES)	MRBAYES SS	VBPI-NF	VCSMC (JC)	VANILLA CSMC	VAIPHY	ϕ-CSMC
DS1	(27, 1949)	**-7032.45 ± 0.15**	-7108.40 ± 0.11	-9180.34 ± 170.27	-8306.76 ± 166.27	-7490.54 ± 0	-7290.36 ± 7.23
DS2	(29, 2520)	**-26363.85 ± 0.33**	-26367.70 ± 0.03	-28700.7 ± 4892.67	-27884.37 ± 226.60	-31203.44 ± $3E^{-12}$	-30568.49 ± 31.34
DS3	(36, 1812)	**-33729.60 ± 0.74**	-33735.09 ± 0.05	-37211.20 ± 397.97	-35381.01 ± 218.18	-33911.13 ± $7E^{-12}$	-33798.06 ± 6.62
DS4	(41, 1137)	**-13292.37 ± 1.42**	-13329.93 ± 0.11	-17106.10 ± 362.74	-15019.21 ± 100.61	-13700.86 ± 0	-13582.24 ± 35.08
DS5	(50, 378)	**-8192.96 ± 0.71**	-8214.61 ± 0.40	-9449.65 ± 2578.58	-8940.62 ± 46.44	-8464.77 ± 0	-8367.51 ± 8.87
DS6	(50, 1133)	**-6571.02 ± 0.81**	-6724.36 ± 0.37	-9296.66 ± 2046.70	-8029.51 ± 83.67	-7157.84 ± 0	-7013.83 ± 16.99
DS7	(64, 1008)	**-8438.78 ± 73.66**	-8650.49 ± 0.53	N/A*	-11013.57 ± 113.49	-9462.21 ± $1E^{-12}$	-9209.18 ± 18.03

additional information about the datasets). Additionally, to the extent possible, we have used the open-source code of our baselines to survey the benchmarks on these popular datasets. We only considered baselines that employ the JC model. Moreover, we assumed an Exp(10) prior over branch lengths, and a uniform prior over tree topologies. This was aligned with the generative models used in all the original works of the baselines.

For all methods, we use the exact parameter configurations reported in the corresponding papers or specified in their available code. We let VBPI-NF [27] and VCSMC [20] train for the number of iterations as specified in their works. For MrBayes [21] with the SS method, we follow the protocol used in [29].

As we are benchmarking sophisticated versions of the phylogenetic-inference CSMC [24], it seems relevant to include a vanilla CSMC. We define the vanilla CSMC as one that proposes branch lengths using the model prior, and merges subtrees uniformly. There are no learnable parameters in the vanilla CSMC.

Indeed, there exists particle MCMC-based approaches for parameterizing the CSMC, e.g. [24] or [25], however these have not previously been benchmarked on the datasets used here, and the latter uses the K2P substitution model. As such, we leave these comparisons for future work and constrain ourselves to VI-based CSMCs.

Following [20], we give all CSMC methods $K = 2048$ particles. The parameters of VCSMC, VaiPhy and ϕ-CSMC are selected based on their best iteration, measured in LL. Using these parameters, we compute the LL estimates averaged over ten random seeds. We run VaiPhy and ϕ-CSMC for 200 iterations, and evaluate them on Eq. (15) and Eq. (16), respectively. Vanilla CSMC is also evaluated using Eq. (16).

In Table 1 we provide the mean LL scores and standard deviations. On all datasets except DS2, our ϕ-CSMC is the superior CSMC method (highlighted in red). It also produces significantly smaller variations in its LL estimates compared to VCSMC and the Vanilla CSMC. On DS2, the Vanilla CSMC is surprisingly the winning CSMC.

Overall, MrBayes SS is the superior method on these datasets followed by VBPI-NF. VaiPhy, despite its heavily penalizing prior distribution over tree topologies (see Sec. 4), still outperforms VCSMC and Vanilla CSMC on all datasets but DS2, with extremely low-variance estimates of the LL.

5.3 Wall-Clock Time and Memory Comparisons

Being a mean-field VI algorithm, VaiPhy truly enjoys the related benefits in terms of speed, which we demonstrate in this section. The experiments are performed on a high performance computing cluster node with two Intel Xeon Gold 6130 CPUs with 16 CPU cores each. Each node in the cluster has 96 GiB RAM.

We run VaiPhy for 200 iterations and 128 trees are sampled using SLANTIS to compute the expectations in Eq. (6) and (8). To compute $\mathcal{L}_L$, 3000 trees are sampled from SLANTIS after 200 iterations. The ϕ-CSMC results include the aforementioned VaiPhy runs and 10 additional independent CSMC runs with 2048 particles. VBPI-NF runs for 400,000 iterations and uses 100,000 trees to estimate the lower bound after training. Each experiment is repeated 10 times. The wall-clock time comparison of the methods on DS1 is presented in Figure 3.

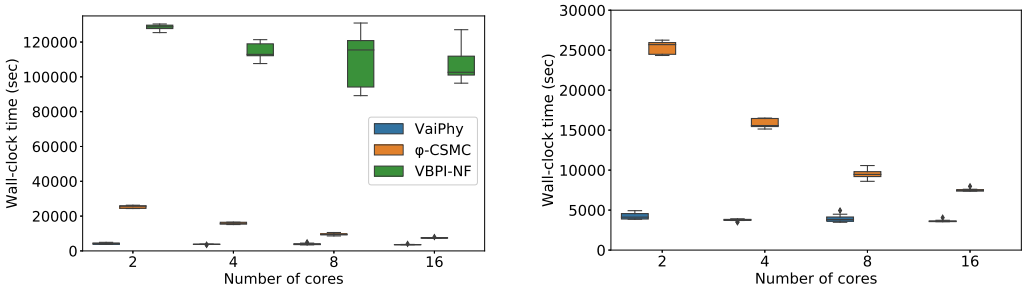

Figure 3: Wall-clock runtime results on DS1 data set with $\{2, 4, 8, 16\}$ cores. Left) Runtime results for VaiPhy, ϕ-CSMC and VBPI-NF. Right) A zoomed in version of the top figure showing runtime results for VaiPhy and ϕ-CSMC. Clearly, VaiPhy is consistently faster than its benchmarks.

We were not able to run VCSMC on less than 16 cores due to the method producing memory-issues on DS1. However, for 16 cores their runtime was around 6840 seconds, with too small variations to show in a a boxplot. VCSMC is thus faster than ϕ-CSMC on 16 cores, but still slower than VaiPhy.

VaiPhy is remarkably much faster than all other methods for all numbers of cores (except for MrBayes in special cases, see further down). Although VBPI-NF produces impressive LL estimates, it is considerably slower than VaiPhy and the CSMC methods (all CSMCs are not shown in the Fig.). Additionally, using two cores, the wall-clock time required by VBPI-NF to arrive at VaiPhy's approximation accuracy was close to 11x that of VaiPhy on DS1. For completeness we compared wall-clock times with MrBayes, although this is not appropriate given that the software has been highly optimized over decades and is written in C. And, indeed, MrBayes is the fastest algorithm when allowed multiple cores. However, for a single core on DS1 MrBayes ran for approximately 100 minutes while VaiPhy only needed impressive 83 minutes on average for the same setup.

Finally we discuss memory requirements. Our main variational baseline, VBPI-NF, used 8.4 million neural network weights and biases on DS1, while the number of learnable parameters (i.e. the dimensions of ϕ and $q(Z|X)$) in VaiPhy were 0.8 million; less than 10% of the former.

6 Conclusion

We have presented a modular framework for Bayesian phylogenetic inference, building on three novel methods: *VaiPhy*, *SLANTIS*, and the *JC sampler*. At the heart of our framework lies ϕ, the estimated number of mutations matrix. Although ϕ was learned in a multifurcating tree space with labeled internal vertices, we showed that the evolutionary distances also served useful in the bifurcating X-tree space. We demonstrated this by using ϕ to parameterize two new proposal distributions for the CSMC, also introduced by us in this work; we referred to this new algorithm as ϕ-CSMC. Our ϕ-CSMC proved to be the best-performing CSMC in terms of marginal log-likelihood estimation on real datasets. Additionally, VaiPhy was up to *25 times faster* than its baselines on real data.

Speed is essential for making Bayesian phylogenetics a reality for future large data sets, in particular, single cell data sets in medical investigations. Moreover, speed and the opportunity to leverage the VI methodology for phylogeny, as provided by our framework, i.e., our graph-based representation of the VI distribution over trees, ϕ, opens up novel research opportunities.

For many potential applications, other models than the JC model will be more suitable models of the evolution that takes place over edges of the phylogenetic tree. However, our JC sampler provides a great starting point. Reasonably, our JC sampler should serve as a better base distribution for normalizing flows than those currently employed [27].

In conclusion, we have presented a VI algorithm for Bayesian phylogeny that, together with its building blocks, unlocks the potential of VI for fast Bayesian analysis in phylogeny and integrated phylogenetic analysis. The by now 25 year old methodology of MCMC based Bayesian phylogentic analysis is certainly a mature and optimized methodology that has great accuracy. However, VaiPhy is faster than MrBayes on a single core, and the underlying methodology is bound to lead to a sequence of increasingly faster and more accurate VI based phylogenetic software. Moreover, it paves the way for integrated VI based phylogenetic inference.

References

[1] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

[2] A. Bouchard-Côté and M. I. Jordan. Variational Inference over Combinatorial Spaces. *NIPS*, 2010.

[3] A. Bouchard-Côté, S. Sankararaman, and M. I. Jordan. Phylogenetic inference via sequential Monte Carlo. *Systematic Biology*, 61(4):579–593, July 2012.

[4] Y. Burda, R. B. Grosse, and R. Salakhutdinov. Importance Weighted Autoencoders. *ICLR*, 2016.

[5] T. Dang and H. Kishino. Stochastic Variational Inference for Bayesian Phylogenetics: A Case of CAT Model. *Molecular Biology and Evolution*, 36(4):msz020, 2019.

[6] P. Diaconis. *Sequential Importance Sampling for Estimating the Number of Perfect Matchings in Bipartite Graphs: An Ongoing Conversation with Laci*, pages 223–233. 01 2019.

[7] V. Elvira, L. Martino, D. Luengo, and M. F. Bugallo. Generalized multiple importance sampling. *Statistical Science*, 34(1):129–155, 2019.

[8] Y. Fan, R. Wu, M.-H. Chen, L. Kuo, and P. O. Lewis. Choosing among Partition Models in Bayesian Phylogenetics. *Molecular Biology and Evolution*, 28(1):523–532, 2011.

[9] J. Felsenstein. Evolutionary trees from dna sequences: a maximum likelihood approach. *Journal of molecular evolution*, 17(6):368–376, 1981.

[10] M. Fourment and A. E. Darling. Evaluating probabilistic programming and fast variational Bayesian inference in phylogenetics. *PeerJ*, 7(1):e8272, 2019.

[11] N. Friedman, M. Ninio, I. Pe'er, and T. Pupko. A structural EM algorithm for phylogenetic inference. *Journal of computational biology : a journal of computational molecular cell biology*, 9(2):331–353, 2002.

[12] J. R. Garey, T. J. Near, M. R. Nonnemacher, and S. A. Nadler. Molecular evidence for acanthocephala as a subtaxon of rotifera. *Journal of Molecular Evolution*, 43(3):287–292, 1996.

[13] O. Gascuel. BIONJ: an improved version of the NJ algorithm based on a simple model of sequence data. *Molecular Biology and Evolution*, 14(7):685–695, 07 1997.

[14] S. B. Hedges, K. D. Moberg, and L. R. Maxson. Tetrapod phylogeny inferred from 18s and 28s ribosomal rna sequences and a review of the evidence for amniote relationships. *Molecular Biology and Evolution*, 7(6):607–633, 1990.

[15] D. A. Henk, A. Weir, and M. Blackwell. Laboulbeniopsis termitarius, an ectoparasite of termites newly recognized as a member of the laboulbeniomycetes. *Mycologia*, 95(4):561–564, 2003.

[16] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 2013.

[17] S. Höhna, M. J. Landis, T. A. Heath, B. Boussau, N. Lartillot, B. R. Moore, J. P. Huelsenbeck, and F. Ronquist. Revbayes: Bayesian phylogenetic inference using graphical models and an interactive model-specification language. *Systematic biology*, 65(4):726–736, 2016.

[18] J. P. Huelsenbeck and F. Ronquist. Mrbayes: Bayesian inference of phylogenetic trees. *Bioinformatics*, 17(8):754–755, 2001.

[19] C. Lakner, P. Van Der Mark, J. P. Huelsenbeck, B. Larget, and F. Ronquist. Efficiency of markov chain monte carlo tree proposals in bayesian phylogenetics. *Systematic biology*, 57(1):86–103, 2008.

[20] A. K. Moretti, L. Zhang, C. A. Naesseth, H. Venner, D. Blei, and I. Pe'er. Variational combinatorial sequential monte carlo methods for bayesian phylogenetic inference. *arXiv preprint arXiv:2106.00075*, 2021.

[21] F. Ronquist, M. Teslenko, P. Mark, D. Ayres, A. Darling, S. Höhna, B. Larget, L. Liu, M. Suchard, and J. Huelsenbeck. Mrbayes 3.2: Efficient bayesian phylogenetic inference and model choice across a large model space. *Systematic biology*, 61:539–42, 03 2012.

[22] A. Y. Rossman, J. M. McKemy, R. A. Pardo-Schultheiss, and H.-J. Schroers. Molecular studies of the bionectriaceae using large subunit rdna sequences. *Mycologia*, 93(1):100–110, 2001.

[23] M. J. Wainwright and M. I. Jordan. Graphical Models, Exponential Families, and Variational Inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2007.

[24] L. Wang, A. Bouchard-Côté, and A. Doucet. Bayesian phylogenetic inference using a combinatorial sequential monte carlo method. *Journal of the American Statistical Association*, 110(512):1362–1374, 2015.

[25] S. Wang and L. Wang. Particle gibbs sampling for bayesian phylogenetic inference. *Bioinformatics*, 37(5):642–649, 2021.

[26] Z. Yang and A. D. Yoder. Comparison of likelihood and bayesian methods for estimating divergence times using multiple gene loci and calibration points, with application to a radiation of cute-looking mouse lemur species. *Systematic biology*, 52(5):705–716, 2003.

[27] C. Zhang. Improved Variational Bayesian Phylogenetic Inference with Normalizing Flows. *arXiv*, 2020.

[28] C. Zhang and F. A. Matsen IV. Generalizing tree probability estimation via bayesian networks. *arXiv preprint arXiv:1805.07834*, 2018.

[29] C. Zhang and F. A. Matsen IV. Variational bayesian phylogenetic inference. In *International Conference on Learning Representations*, 2018.

[30] N. Zhang and M. Blackwell. Molecular phylogeny of dogwood anthracnose fungus (discula destructiva) and the diaporthales. *Mycologia*, 93:355–365, 03 2001.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] We discuss the extension to labelled multifurcating tree space and the requirement of the post-processing step that is required to make fair comparisons in bifurcating space.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] We show the conjugacy of JC sampler and we provide proof of natural gradient in the Appendix.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We will provide the codes in the supplementary materials.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] In the Experiments section we mention the choice of parameters. In Appendix, we provide code segments to show how we ran the benchmark methods.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] In Table 1 we include the standard deviations and in Figure 3 we include the error bars.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] We wrote the hardware specifications in the runtime experiments; however, we didn't include the total amount of time required for all datasets.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We cited all software and also provided GitHub links to the repositories in the Appendix.
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] The code is available in the supplementary material.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]