Algorithms for Molecular Biology

RESEARCH Open Access

A topology-marginal composite likelihood via a generalized phylogenetic pruning algorithm

Seong-Hwan Jun[1†], Hassan Nasif[2†], Chris Jennings-Shaffer[3], David H Rich[3], Anna Kooperberg[3], Mathieu Fourment[4], Cheng Zhang[5], Marc A Suchard[6,7,8] and Frederick A Matsen IV[3,9,10,11*]

Abstract

Bayesian phylogenetics is a computationally challenging inferential problem. Classical methods are based on random-walk Markov chain Monte Carlo (MCMC), where random proposals are made on the tree parameter and the continuous parameters simultaneously. Variational phylogenetics is a promising alternative to MCMC, in which one fits an approximating distribution to the unnormalized phylogenetic posterior. Previous work fit this variational approximation using stochastic gradient descent, which is the canonical way of fitting general variational approximations. However, phylogenetic trees are special structures, giving opportunities for efficient computation. In this paper we describe a new algorithm that directly generalizes the Felsenstein pruning algorithm (a.k.a. sum-product algorithm) to compute a composite-like likelihood by marginalizing out ancestral states and subtrees simultaneously. We show the utility of this algorithm by rapidly making point estimates for branch lengths of a multi-tree phylogenetic model. These estimates accord with a long MCMC run and with estimates obtained using a variational method, but are much faster to obtain. Thus, although generalized pruning does not lead to a variational algorithm as such, we believe that it will form a useful starting point for variational inference.

†Seong-Hwan Jun, Hassan Nasif have contributed equally to this work.

*Correspondence:
Frederick A Matsen IV
matsen@fredhutch.org
[1] Department of Biostatistics and Computational Biology, University of Rochester, Rochester, USA
[2] Department of Statistics, University of Washington, Seattle, USA
[3] Public Health Sciences Division, Fred Hutchinson Cancer Research Center, Seattle, WA, USA
[4] Australian Institute for Microbiology and Infection, University of Technology Sydney, Ultimo, NSW, Australia
[5] School of Mathematical Sciences and Center for Statistical Science, Peking University, Beijing, China
[6] Department of Human Genetics, University of California, Los Angeles, USA
[7] Department of Computational Medicine, University of California, Los Angeles, USA
[8] Department of Biostatistics, University of California, Los Angeles, USA
[9] Department of Genome Sciences, University of Washington, Seattle, USA
[10] Howard Hughes Medical Institute, Fred Hutchinson Cancer Research Center, Seattle, Washington, USA
[11] Computational Biology Program, Fred Hutchinson Cancer Research Center, 1100 Fairview Ave. N., Mail stop: S2-140, Seattle, WA 98109-1024, USA

Introduction

Statistical phylogenetics is largely divided into maximum-likelihood based and Bayesian posterior-based approaches. The former searches for a tree that yields highest likelihood for the observed sequencing data, while the latter is typically approached using Markov chain Monte Carlo (MCMC) sampling to estimate the posterior probabilities of trees given the observed sequencing data. Both approaches involve exploration of the tree space using local tree rearrangements, such as nearest neighbor interchange or subtree pruning and regrafting [1], followed by computation of the likelihood of the sequence data given the tree using Felsenstein's pruning algorithm [2]. The Felsenstein pruning algorithm prunes out the ancestral states at the internal nodes of the tree and is the engine driving the advances in statistical phylogenetics, allowing estimation of branch lengths of a tree as well as the parameters of the evolutionary models. Specifically, the two-pass version of the algorithm allows for constant-time updates to branch lengths and local tree structures. In this paper, we propose a generalization of the two-pass Felsenstein pruning algorithm that marginalizes uncertain tree structures as well as the ancestral states, with a long-term goal of bringing efficient optimization strategies from maximum-likelihood phylogenetics to Bayesian inference.

To begin to appreciate the challenges of Bayesian phylogenetics, we start with an overview of likelihood-based phylogenetic models (see the Background and Notation section for a full development). Assume that we are given a multiple sequence alignment [3, 4] of DNA sequences as data $\mathbf{Y}$ that maps a sequence of molecular characters (i.e. DNA bases) to each leaf of the phylogenetic tree that generated it. This alignment is organized in terms of a list of *sites* such that the differences between sites are assumed to arise only due to substitution of one DNA base for another along the course of evolution. Specifically, per-site sequence change is formulated in terms of continuous-time Markov chain (CTMC) models of DNA sequence evolution along the branches of the tree, where the time parameter in the CTMC is called *branch length*. The CTMC may also have other parameters, such as the rate of change from one DNA base to another. For this paper, a *(phylogenetic) tree* is defined to be a rooted bifurcating tree structure τ with leaf labels that has been equipped with branch lengths on every edge. We follow common practice by using the word *topology* to describe the discrete component of this model, namely the tree without branch lengths.

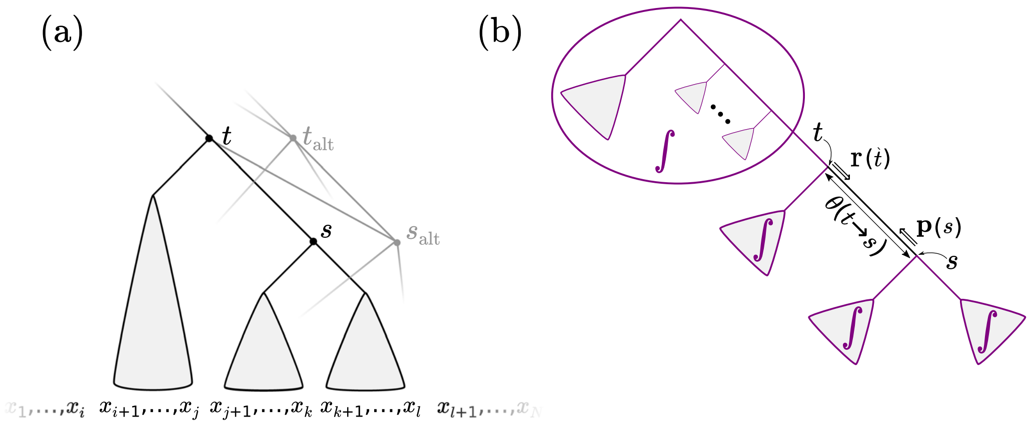
We make the typical independence-across-sites assumption for evolutionary processes conditioned on the tree, which enables efficient likelihood computation via a dynamic programming approach that integrates out the unobserved molecular characters (*ancestral states*) at all of the internal nodes. This approach is called the *pruning algorithm* [2] in phylogenetics, which is reviewed below, and is also known as the sum-product algorithm or belief propagation in other settings [5]. It enables linear-complexity calculation in the number of sequences for the phylogenetic likelihood $p(\mathbf{Y} \mid \tau, \boldsymbol{\theta})$, where τ is the topology and $\boldsymbol{\theta}$ is a corresponding vector of branch lengths, as well as constant-complexity updates for local modifications. Assume we are given a prior $p(\tau, \boldsymbol{\theta})$ on phylogenetic trees. We assume here that the prior factors into two easily-calculated terms: the prior $p(\tau)$ on topologies and the prior $p(\boldsymbol{\theta} \mid \tau)$ on branch lengths given a topology.

Recent work has fit reduced-dimension probabilistic models to the topological posterior [6–9]. Briefly, these methods break topologies into building blocks such that probabilistic models on these building blocks can be translated into probabilistic models on whole topologies themselves. We have shown that this translation provides a flexible distribution on topologies with good inductive biases [8].

One can think of these reduced-dimension models in terms of a structure we call a *subsplit directed acyclic graph* or *subsplit DAG* (introduced below). One can think of the nodes of the subsplit DAG as comprising the union of substructures, called *subsplits*, of a collection of topologies. The edges between these nodes represent compatibility of substructures (Fig. 1a). We arrive at a probability distribution on topologies by attaching probabilities to the edges of the DAG.

Stated in these terms, in [9] we used a variational approach, with modern yet general-purpose gradient estimators, to fit continuous parameters (i.e. edge probabilities and branch length distributions) to the subsplit DAG. This variational approach results in an excellent approximation to the phylogenetic posterior distribution, and converges in relatively few iterations. However, this variational approach struggles to be time-competitive with classical random-walk MCMC because of the stochasticity of the gradient estimator as well as the cost of evaluating the gradient of the phylogenetic likelihood function.

In this work, we begin addressing these difficulties via a new algorithm that performs dynamic programming directly on the subsplit DAG via a generalization of the Felsenstein algorithm. This *generalized pruning* (GP) algorithm marginalizes a likelihood function over ancestral states and topologies at the same time (Fig. 1). By forming a new type of partial likelihood vector that integrates out all of the topologies in the support that



(a)
t
t_{alt}
s
s_{alt}
$x_1,\ldots,x_i$ $x_{i+1},\ldots,x_j$ $x_{j+1},\ldots,x_k$ $x_{k+1},\ldots,x_l$ $x_{l+1},\ldots,x_N$
(b)
t
$\mathbf{r}(t)$
$\theta(t\to s)$
$\mathbf{p}(s)$
s

Fig. 1 A preview of the core components of the algorithm to give intuition; concepts will be introduced in the text. **a** The subsplit DAG, which encodes a collection of phylogenetic tree topologies on leaves $x_1,\ldots,x_N$. One such topology is partially shown in black, with alternate topologies indicated with gray lines. The nodes of this DAG are uniquely associated with "subsplits" that give the bipartition of taxa below them (subsplits corresponding to alternate topologies are marked with $\cdot_{\mathrm{alt}}$). For example, the subsplit t is $(\{x_{i+1},\ldots,x_j\},\{x_{j+1},\ldots,x_l\})$. Edges go between compatible subsplits, such as between t and s, and are directed towards the leaves. **b** Overview of method: given an edge of the DAG we integrate ($\int$) out all of the topologies in the DAG that contain the DAG edge $t\to s$. Branch lengths θ are associated to DAG edges. We perform efficient inference using "rootward" **r** and "leafward" **p** partial likelihood vectors marginalized over unknown structure of trees encoded in the DAG

contain a given DAG edge, we are able to perform constant-time updates to the branch length associated with that structure.

This generalized pruning algorithm based on integrated partial likelihood vectors relies on several modeling approximations. First, the likelihood we use is a composite-like model likelihood marginalized over topologies τ:

$$\prod_{k=1}^{K}\sum_{\tau\in\mathcal{D}} p_{\boldsymbol{\theta}}(Y_k \mid \tau)\,p(\tau) \tag{1}$$

where Y_k is the k-th column of the sequence alignment $\mathbf{Y}$ and $\mathcal{D}$ is a structure that contains many topologies described below. In contrast, exact likelihood computation marginalizes over the topologies, with each term being a product across sites. Second, we parameterize the continuous aspects of phylogenetic trees in a very simple way: one fixed branch length per DAG edge. This is unusual for Bayesian phylogenetics, in which one typically considers a distribution of branch lengths, however this style of modeling approximation achieves surprisingly good performance [10–12]. Furthermore, this is a necessary assumption for the efficient implementation of our GP algorithm, as anything else would require an integration over branch lengths. We use $\boldsymbol{\theta}$ to denote the vector of these parameters.

This paper is focused on providing a complete description of the generalized pruning algorithm, as well as describing fast parameter estimation procedures for the composite-like model likelihood shown in Eq. (1). We perform experiments to demonstrate the efficacy of the procedures introduced in this paper, specifically comparing branch length estimation to a more established procedure [9], benchmarked in terms of computational effort. We emphasize that generalized pruning does not lead to a variational algorithm as such, however, we believe that it will form a useful companion for variational inference by providing initial parameter estimates.

We assume for this paper that the subsplit DAG is provided to the algorithm. Constructing this DAG is an interesting challenge in itself and the subject of ongoing research; we hope to use generalized pruning to infer the structure of the subsplit DAG using a procedure analogous to maximum-likelihood phylogenetic inference. However, one existing option is to build the subsplit DAG out of trees obtained by bootstrapping as practiced in maximum likelihood phylogenetics [9], or to use trees from an initial MCMC run [13].

Background and notation

We begin by introducing common phylogenetics notation (mostly following [14]) and the notion of subsplits as buildings blocks for modeling tree structures.

Setup for likelihood-based phylogenetics

We use the word *taxon* (plural *taxa*) to describe an entity associated with a molecular sequence. In classical evolutionary phylogenetics, taxa are commonly species, although they could be other entities such as samples of viruses. We assume that a taxon set X is given, and that we have a lexicographic order on it. Also assume that we are given a sequence alignment on our taxon set X: an arrangement of the molecular sequences for X into a rectangular array such that sequence differences between sites in a single column are assumed to be due to point mutation [4, 15]. Here we focus on rooted bifurcating trees, and as mentioned above use the terminology *topology* to refer to a rooted bifurcating tree structure with the tips of the tree being labeled in a 1-to-1 fashion with the set X.

We use $N = |X|$ to denote the cardinality of the taxon set under consideration. The observed sequences are denoted by $\mathbf{Y}$. We let Σ denote the set of states in the sequences. If the sequences are DNA, Σ is the set of nucleotide bases $\{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$; our implementation is specialized to this case. The length of the DNA sequences is denoted by M. We denote the m-th site over all taxa (i.e. the m-th column of the alignment) by $\mathbf{Y}^m \in \Sigma^N$. Topologies are denoted by τ. The collection of all branch length parameters is denoted by $\boldsymbol{\theta}$ with individual branch length by θ. The likelihood of the observed sequences is denoted by $\mathbb{P}(\mathbf{Y} \mid \tau, \boldsymbol{\theta})$, which under the standard site independence assumption can be expressed as

$$\mathbb{P}(\mathbf{Y} \mid \tau, \boldsymbol{\theta}) = \prod_{m=1}^{M} \mathbb{P}(\mathbf{Y}^m \mid \tau, \boldsymbol{\theta}). \tag{2}$$

The phylogenetic model underlying the likelihood computation is typically a continuous time Markov chain (CTMC) evolving along the branches. The rate matrix of a CTMC is denoted by $\mathbf{Q}$, which yields the transition matrix via matrix exponentiation, denoted $\mathbf{P}$:

$$\mathbf{P}(\theta) = \exp(\theta \mathbf{Q}).$$

The transition matrix plays a key role in the likelihood calculation. We follow the English-typical convention that probability transition matrices are right-stochastic, so that the (i, j)-th entry of $\mathbf{P}$ is $\mathbb{P}(j \mid i)$ for $i, j \in \Sigma$. We denote transposition of vectors and matrices by $^\top$ (in contrast to [14], which uses $'$). For brevity of notation, we may omit θ when referencing a specific entry of the transition matrix.

For simplicity of exposition and implementation, we assume the Jukes-Cantor model for DNA sequences, under which there are no CTMC model parameters other than branch lengths. Additional CTMC model parameters could be added and fit in a maximum-likelihood sense without much difficulty, but given that such parameter fitting is now standard, we focus on our novel tree-marginalization procedure. However, in the mathematical exposition we do not assume that the probability transition matrices are symmetric as they are in the Jukes-Cantor model.

Likelihood calculation over a tree using a two-pass algorithm

We briefly describe the two-pass version of the Felsenstein pruning algorithm [2, 14, 16–18] over a single tree using the notion of *partial likelihood vectors* (PLVs). For the rest of this section, **we will compute the likelihood of a single site** m without further specification, such that what we called $\mathbf{Y}^m$ will now be called $\mathbf{Y}$. We will return to the multiple-site case in the section "Composite-like marginal likelihood."

We follow the exposition and notation in [14], except that we express partial likelihood vectors in terms of subtrees, because in our setting we will deal with many trees and cannot unambiguously describe the algorithm in terms of nodes of a given fixed tree. We also compute partial likelihood vectors at a slightly different location on the tree. For an internal node v, $Y_v \in \Sigma$ will denote the state of v. We use the word "leafward" to refer to the direction in the tree towards the leaves; if the tree is displayed with the root on top and the leaves hanging down, leafward is down. $\mathbf{Y}_{\lfloor v \rfloor}$ will denote the sequences leafward of v. The direction towards the root of the tree will be called "rootward" and we let $\mathbf{Y}_{\lceil v \rceil} = \mathbf{Y} \backslash \mathbf{Y}_{\lfloor v \rfloor}$.

Assume we have a topology τ on all of the sequences $\mathbf{Y}$. Define $\tau_v^\downarrow$ to be the topology with all the nodes leafward of v, including v. Define $\tau_v^\uparrow$ to be the topology with all the nodes of the tree in the rootward direction of v, excluding v; that is, $\tau_v^\uparrow$ is the topology on all of the nodes not in $\tau_v^\downarrow$. The i-th element of the p- and r-PLVs at node v of the topology τ store

$$\mathbf{p}(\tau_v^\downarrow)_i := \mathbb{P}(\mathbf{Y}_{\lfloor v \rfloor} \mid \tau_v^\downarrow, Y_v = i), \tag{3}$$

$$\mathbf{r}(\tau_v^\uparrow)_i := \mathbb{P}(\mathbf{Y}_{\lceil v \rceil}, Y_{\mathsf{pa}(v)} = i \mid \tau_v^\uparrow), \tag{4}$$

where $i \in \Sigma$ and $\mathsf{pa}(v)$ is the parent node of v (towards the root). Note that we describe $\mathbf{p}$ and $\mathbf{r}$ in terms of topologies on subsets of the taxon set, rather than simply a node v, which will become important below when we allow τ to vary. Equation (3) can be interpreted as the marginal likelihood of the observed sequences "below" v, conditioned on $Y_v = i$ for $i \in \Sigma$. Similarly, Eq. (4) stores the likelihood of the data "above" v, however, this time we have the joint likelihood of the observed sequences along with the state at the parent of v.

Note that for any v, the likelihood is given by the product of the p- and r-vectors:

$$\mathbf{r}(\tau_v^\uparrow)^\top \mathbf{P}(\theta_{\mathsf{pa}(v),v}) \mathbf{p}(\tau_v^\downarrow) = \mathbb{P}(\mathbf{Y} \mid \boldsymbol{\theta}, \tau) = \sum_{i,j \in \Sigma} \mathbb{P}(\mathbf{Y}_{\lceil v \rceil}, Y_{\mathsf{pa}(v)} = i \mid \tau_v^\uparrow) \mathbb{P}(Y_{\mathsf{pa}(v)} = i, Y_v = j \mid \boldsymbol{\theta}) \mathbb{P}(\mathbf{Y}_{\lfloor v \rfloor} \mid \tau_v^\downarrow, Y_v = j).$$

Such a decomposition can be performed with respect to any internal node.

We now review the classical two-pass dynamic program showing how to calculate these partial likelihood vectors, mostly following [14] but with a slightly different formulation, which is more appropriate for our setting. Specifically, our $\mathbf{r}$ vectors are our "upper partial" vectors, taking the role of their $\mathbf{q}$ vectors, but calculated on the root-side rather than the leaf-side of a given edge.

Let x and y be the child nodes of v and $\theta_{v,x}, \theta_{v,y}$ denote the branch length between v, x and v, y respectively (Fig. 2). Dropping conditioning on tree structures and branch lengths for simplicity yields

$$\begin{aligned}\mathbb{P}(\mathbf{Y}_{\lfloor v \rfloor} \mid Y_v) &= \sum_{Y_x \in \Sigma} \sum_{Y_y \in \Sigma} \mathbb{P}(\mathbf{Y}_{\lfloor v \rfloor}, Y_x, Y_y \mid Y_v) \\ &= \left[\sum_{Y_x \in \Sigma} \mathbb{P}(Y_x \mid Y_v)\, \mathbb{P}(\mathbf{Y}_{\lfloor x \rfloor} \mid Y_x) \right] \\ &\quad \left[\sum_{Y_y \in \Sigma} \mathbb{P}(Y_y \mid Y_v)\, \mathbb{P}(\mathbf{Y}_{\lfloor y \rfloor} \mid Y_y) \right].\end{aligned}$$

In matrix notation,

$$\mathbf{p}(\tau_v^{\downarrow}) = \left(\mathbf{P}(\theta_{v,x})\, \mathbf{p}(\tau_x^{\downarrow}) \right) \circ \left(\mathbf{P}(\theta_{v,y})\, \mathbf{p}(\tau_y^{\downarrow}) \right) \tag{5}$$

where $\circ$ denotes element-wise multiplication.

Now let u be the parent of v (Fig. 2). We have

$$\begin{aligned}\mathbb{P}(\mathbf{Y}_{\lceil y \rceil}, Y_v) &= \mathbb{P}(\mathbf{Y}_{\lfloor x \rfloor} | Y_v)\, \mathbb{P}(\mathbf{Y}_{\lceil v \rceil}, Y_v) \\ &= \left[\sum_{Y_x \in \Sigma} \mathbb{P}(Y_x | Y_v) \mathbb{P}(\mathbf{Y}_{\lfloor x \rfloor} | Y_x) \right] \\ &\quad \left[\sum_{Y_u \in \Sigma} \mathbb{P}(Y_v | Y_u) \mathbb{P}(\mathbf{Y}_{\lceil v \rceil}, Y_u) \right].\end{aligned}$$

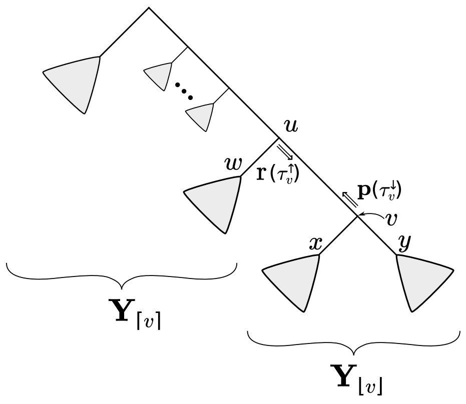
This recursion can be expressed in matrix form as

$$\mathbf{r}(\tau_y^{\uparrow}) = \left(\mathbf{P}(\theta_{v,x}) \mathbf{p}(\tau_x^{\downarrow}) \right) \circ \left(\mathbf{P}(\theta_{u,v})^{\top} \mathbf{r}(\tau_v^{\uparrow}) \right). \tag{6}$$

The subsplit directed acyclic graph

In addition to the phylogenetic likelihood computation, the other main ingredient for our algorithm is a structure on which one can marginalize over tree structures. We call this structure the *subsplit directed acyclic graph* (*subsplit DAG*), which encodes a collection of tree topologies, $\mathcal{T}$ (Fig. 3). This structure can be equipped with edge probabilities to give a probability distribution on tree topologies, and with branch lengths or distributions thereof to give a probability distribution on phylogenetic trees. We use the subsplit DAG to develop a formulation of variational distributions on tree structures as previously expressed in different language [8, 9]. This new language is necessary for the more complex traversals required here.

To define the subsplit DAG, we need a few concepts as follows. A *clade* W is subset of the taxa X, which in the context of a topology τ identifies a subset of the taxa deriving from a common ancestor (i.e.,



u
w
$\mathbf{r}(\tau_v^{\uparrow})$
$\mathbf{p}(\tau_v^{\downarrow})$
v
x
y
$\mathbf{Y}_{\lceil v \rceil}$
$\mathbf{Y}_{\lfloor v \rfloor}$

Fig. 2 Notation for the two-pass likelihood calculation on a tree. Nodes are denoted with *u*, *v*, *w*, *x*, and *y*, and partial likelihood vectors are denoted with **p** and **r**. The directions of double arrows indicate the flow of information to calculate the likelihood of a tree at a given edge

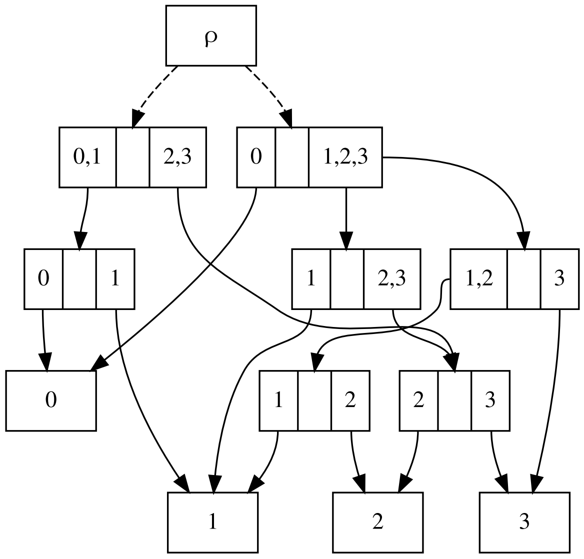


Fig. 3 The subsplit DAG containing three trees on four taxa. Written in Newick [19] parenthetical notation these are ((0, 1), (2, 3)), (0, ((1, 2), 3)), and (0, (1, (2, 3))). We obtain a tree by choosing a single edge out of every "clade" (e.g. {1, 2, 3}) from each subplit (e.g. {{0}, {1, 2, 3}}), and one of the dashed edges from the DAG root ρ

monophyletic group). The order on taxa induces a total order on the clades. A *subsplit* $s = \{V, Z\}$ partitioning a clade W is an unordered pair of disjoint subclades of W such that $V \cup Z = W$. V and Z are called the clades of the subsplit s. We define subsplits in terms of unordered pairs for when we need to express a subsplit in terms of set operations, but always draw subsplits in lexicographic order in figures and call the lexicographically smaller clade the "left" clade and the lexicographically larger clade the "right" clade. We require that the two clades forming a subsplit are each non-empty except in two special cases: "leaf subsplits" $\{\{x\}, \emptyset\}$ for some $x \in X$, and the "universal ancestor (UA) subsplit" $\rho := \{X, \emptyset\}$. Given a subsplit $s = \{V, Z\}$, define $U(s)$ to be $V \cup Z$, the set of taxa in the subsplit.

With the above definitions, we can define the subsplit DAG for a collection of topologies $\mathcal{T}$ as a graph with nodes being the set of subsplits for those topologies, and where there is an edge from t to s if $U(s)$ corresponds to one of the two clades of t. (One can define a subsplit DAG in a more abstract way without referring to a set of topologies $\mathcal{T}$, but this definition is sufficient here.) For the purposes of this paper we assume that $\mathcal{T}$ is supplied. In the long run, and as described in the Discussion, our goal with generalized pruning is to enable algorithms that will allow us to infer the structure of the subsplit DAG.

We will also need a notion of a subsplit where we are focusing attention on one of the clades of the subsplit, which we will call a *subsplit-clade*. This is useful, for example, in describing a collection of edges descending from a single "side" of a subsplit (e.g. the two edges coming from clade $\{1, 2, 3\}$ in the subsplit $\{\{0\}, \{1, 2, 3\}\}$ in Fig. 3). Given a subsplit s and a clade Z of s, we let $(s, \underline{Z})$ denote the subsplit-clade focusing attention on the clade Z. We use $\acute{s}$ to denote the left (i.e. lexicographically smaller) subsplit-clade of subsplit s and $\grave{s}$ to denote the right (i.e. lexicographically larger) subsplit-clade of subsplit s. We say a topology τ contains a subsplit $s = \{V, Z\}$ if V, Z, and $V \cup Z$ are all clades of the topology τ. We denote the DAG on the subsplits by $\mathcal{D}$ and define ρ to be the DAG root; we will also use ρ to signify the subsplit at the DAG root: $\{X, \emptyset\}$. We say an edge $t \to s \in \mathcal{D}$ iff there is an edge $(t, \underline{Z}) \to s$ in the DAG for one of the clades Z of t. Similarly, we say an edge $t \to s \in \tau$ iff s, t are contained in τ and s appears as one of the clades Z of t. We can also say $\tau \in \mathcal{D}$ iff $t \to s \in \mathcal{D}$ for all $t \to s \in \tau$.

Parameterizing the subsplit DAG

We can equip the subsplit DAG with parameters that turn it into a true probability distribution on phylogenetic trees [8, 9]. First, we have probability distributions for resolving a subsplit-clade $(t, \underline{Z})$, which we will write as $\mathbb{P}(s \mid (t, \underline{Z}))$, such that

$$\mathbb{P}(s \mid (t, \underline{Z})) \geq 0,$$
$$\sum_{s:(t,\underline{Z})\to s} \mathbb{P}(s \mid (t, \underline{Z})) = 1.$$

We can simplify notation by defining $\mathbb{P}(s \mid t)$ for an edge $t \to s$ of the DAG to be whichever version makes sense: for example if $\acute{t} \to s$, then $\mathbb{P}(s \mid t) := \mathbb{P}(s \mid \acute{t})$. Note that this is an abuse of notation because $\mathbb{P}(s \mid t)$ is not a normalized probability distribution across all possible s because they are allowed to resolve either clade of a given subsplit.

These conditional probabilities combine to give a normalized probability distribution on topologies [8]:

$$\mathbb{P}(\tau) = \prod_{t\to s\in\tau} \mathbb{P}(s \mid t). \tag{7}$$

Given probabilities on $\mathbb{P}(s \mid t)$, we can recursively compute the probability of a sub-topology descending from a subsplit. For example, imagine that we have two topologies (τ_1, τ_2) on disjoint taxon subsets of X, and that t is the subsplit consisting of the taxa of τ_1 for one clade and the taxa of τ_2 for the other. Assume that the taxon set of τ_1 is lexicographically smaller than that of τ_2. There is a 1-to-1 correspondence between all such (τ_1, τ_2) ordered pairs on those taxon sets and topologies τ_t on the union of the taxon sets: if we are given (τ_1, τ_2) we simply join them together to make τ_t. We express the conditional probability of such a topology recursively as

$$\mathbb{P}(\tau_t \mid t) = \mathbb{P}(s_1 \mid \acute{t})\, \mathbb{P}(\tau_1 \mid s_1)\, \mathbb{P}(s_2 \mid \grave{t})\, \mathbb{P}(\tau_2 \mid s_2), \tag{8}$$

where each s_i is the subsplit on the taxa appearing in subtrees τ_i for $i = 1, 2$. We utilize such recursive definitions of the tree probability in the development of generalized pruning algorithm.

However, in contrast to previous work [8, 9] in which these probabilities express an approximation to the posterior distribution, in this case they express a prior on tree topologies, which we will combine with a likelihood to get a posterior kernel. For this paper we assume that the topological prior can be expressed in terms of such $\mathbb{P}(s \mid t)$. This exact criterion is not strictly necessary, but we do need a way of computing it in terms of rootward and leafward components. Further details on prior computation are given in the section "Prior on subsplit parameters."

As described above, we will attach a single branch length parameter to each edge of the DAG. The branch length for edges originating in the root node ρ have no meaning and can be ignored.

The immediate goal of our work is to optimize these branch lengths via marginalization of tree topologies; the exact meaning of this will be made explicit in the next section.

Methods

In this section we describe the two-pass generalized pruning (GP) algorithm. We assume a DAG $\mathcal{D}$ is given, and all statements about DAG edges are with respect to that given DAG. The two-pass GP algorithm can efficiently compute the marginal likelihood for a single site, where marginalization is over the topologies and the states of the internal nodes of the trees in $\mathcal{D}$. Utilizing the two-pass algorithm, we formulate the *composite marginal likelihood* as a means to estimate the branch length parameters.

We index transition probability matrices by DAG edges $t \to s$. To avoid deep subscripting, we will use a function-type representation $\mathbf{P}(\boldsymbol{\theta}(t \to s))$, where $\boldsymbol{\theta}$ is the vector of branch lengths indexed by branch $t \to s$. These transition probabilities are completely defined by the branch length vector $\boldsymbol{\theta}$, because we assume no parameters of the substitution model other than branch lengths. With this assignment of branch lengths to edges, each rooted topology has a unique assignment of branch lengths and thus a well-defined likelihood.

DAG nodes can be unambigiously labeled with their subsplits, and so we will treat DAG nodes and subsplits interchangeably. We associate p- and r-PLVs to each node of the subsplit DAG. The p-PLVs are computed in the *rootward traversal* whereas the r-PLVs are computed in the *leafward traversal*. We describe details of the two passes in the subsequent sections.

We will extend the notation in the two-pass algorithm in Eqs. (3) and (4). Previously, we defined $\mathbf{p}$ and $\mathbf{r}$ as partial likelihoods of the observed sequences as functions of (partial) topologies for a given node v: $\mathbf{p}(\tau_v^{\downarrow})$ and $\mathbf{r}(\tau_v^{\uparrow})$. Below we will extend this notation by defining $\mathbf{p}(t)$ as a partial likelihood vector for a subsplit t, which is obtained via marginalization of possible topologies involving the subsplit t using the partial likelihoods $\mathbf{p}(\tau_t^{\downarrow})$. Similarly, we will define a $\mathbf{r}((s, \underline{Z}))$ (where $(s, \underline{Z})$ will be either $\acute{s}$ or $\grave{s}$) using a similar marginalization involving $\mathbf{r}(\tau_{s,\underline{Z}}^{\uparrow})$. In order to compute these terms we introduce intermediate sums $\check{\mathbf{p}}(\acute{s})$, $\check{\mathbf{p}}(\grave{s})$, and $\check{\mathbf{r}}(s)$.

Rootward traversal

In the rootward traversal we assume that we visit a node after visiting all of its descendants, such as via a postorder traversal.

Given a subsplit $t = \{A, B\}$, let the *leafward topologies of* t, denoted $\mathcal{T}_{\mathrm{leaf}}(t)$, be the set of rooted topologies on the taxon set $A \cup B$ with t as the bipartition at the root of the topology. As above, we will use the notation $\tau^{\downarrow}$ to emphasize that these are *not* topologies on the entire taxon set X, but only specify structure for a topology "below" a subsplit. We define the partial likelihood vector $\mathbf{p}(t)$ of a non-leaf subsplit t as

$$\mathbf{p}(t) := \sum_{\tau_t^{\downarrow} \in \mathcal{T}_{\mathrm{leaf}}(t)} \mathbf{p}(\tau_t^{\downarrow})\, \mathbb{P}(\tau_t^{\downarrow} \mid t), \tag{9}$$

where $\mathbf{p}(\tau_t^{\downarrow})$ is the partial likelihood vector as defined by (3) at the root of a topology $\tau_t^{\downarrow}$ on $U(t)$ and $\mathbb{P}(\tau_t^{\downarrow} \mid t)$ is the prior probability of $\tau_t^{\downarrow}$ given t as described above. If t is a leaf subsplit $(\{x\}, \emptyset)$, then $\mathbf{p}(t)$ is the tip partial likelihood vector for the taxon x, i.e., the vector with one entry corresponding to the observed nucleotide base set to 1 and the remaining entries set to 0.

We can calculate $\mathbf{p}(t)$ via a dynamic program generalizing the single-tree case:

Lemma 1 *Given a subsplit t,*

$$\mathbf{p}(t) = \left(\sum_{s_1 \leftarrow \acute{t}} \mathbf{P}(\boldsymbol{\theta}(\acute{t} \to s_1))\mathbf{p}(s_1)\, \mathbb{P}(s_1 \mid \acute{t}) \right) \circ \left(\sum_{s_2 \leftarrow \grave{t}} \mathbf{P}(\boldsymbol{\theta}(\grave{t} \to s_2))\mathbf{p}(s_2)\, \mathbb{P}(s_2 \mid \grave{t}) \right). \tag{10}$$

Here and below we use $s_1 \leftarrow \acute{t}$ as an abbreviation for $\{s_1 : \acute{t} \to s_1\}$, i.e. the set of subsplits s_1 of $\mathcal{D}$ that are direct descendants of the left subsplit-clade of t.

Proof We start with the right-hand side of Eq. (10) and show equality with the left-hand side. For each $j = 1, 2$, substitute in

$$\mathbf{p}(s_j) = \sum_{\tau_j^{\downarrow} \in \mathcal{T}_{\mathrm{leaf}}(s_j)} \mathbf{p}(\tau_j^{\downarrow})\, \mathbb{P}(\tau_j^{\downarrow} \mid s_j).$$

By pulling out sums and rearranging the order of terms, we have the sum of the following quantity over subsplits $s_1 \leftarrow \acute{t}$ and $s_2 \leftarrow \grave{t}$:

$$\sum_{\substack{\tau_1^{\downarrow} \in \mathcal{T}_{\text{leaf}}(s_1) \\ \tau_2^{\downarrow} \in \mathcal{T}_{\text{leaf}}(s_2)}} \left(\mathbf{P}(\boldsymbol{\theta}(\acute{t} \to s_1))\mathbf{p}(\tau_1^{\downarrow})\right) \circ \left(\mathbf{P}(\boldsymbol{\theta}(\grave{t} \to s_2))\mathbf{p}(\tau_2^{\downarrow})\right) \mathbb{P}(s_1 \mid \acute{t})\, \mathbb{P}(\tau_1^{\downarrow}|s_1)\mathbb{P}(s_2 \mid \grave{t})\, \mathbb{P}(\tau_2^{\downarrow}|s_2).$$

We define $\tau_t^{\downarrow}$ to be the topology built by joining $\tau_1^{\downarrow}$ and $\tau_2^{\downarrow}$ using t, so that

$$\mathbf{p}(\tau_t^{\downarrow}) = \left(\mathbf{P}(\boldsymbol{\theta}(\acute{t} \to s_1))\mathbf{p}(\tau_1^{\downarrow})\right) \circ \left(\mathbf{P}(\boldsymbol{\theta}(\grave{t} \to s_2))\mathbf{p}(\tau_2^{\downarrow})\right)$$

by Eq. (5), and

$$\mathbb{P}(\tau_t^{\downarrow} \mid t) = \mathbb{P}(s_1|\acute{t})\, \mathbb{P}(\tau_1^{\downarrow}|s_1)\, \mathbb{P}(s_2|\grave{t})\, \mathbb{P}(\tau_2^{\downarrow}|s_2)$$

by Eq. (8). Under this construction for $\tau_t^{\downarrow}$, the sums over subsplits $s_1 \leftarrow \acute{t}$ and $s_2 \leftarrow \grave{t}$ combined with the sums over the leafward topologies of s_1 and s_2 are equivalent to a single sum over the leafward topologies of t. This concludes the proof by the definition of $\mathbf{p}(t)$. □

Rootward topologies

We have equivalent notions of partial likelihood for leafward traversals. Although conceptually quite similar, we will require additional definitions and notation.

First, we need a notion of a rooted topology where the structure of the tree is unspecified for a subset of the taxa that appear together in the tree; this will generalize the notion of $\tau_v^{\uparrow}$ defined above. We formalize this by expressing a partially-specified topology as the set of subsplits it contains. For $Z \subset X$, define a *Z-unspecified topology* as a set of subsplits of the form $C \setminus D$, where C is the subsplit representation of a topology τ on X with Z as a clade, and D is the subsplit representation of the sub-topology of τ on Z. Furthermore, there is a natural definition of the probability of such a partially-specified topology as the product of the corresponding set of probabilities in Eq. (7).

Given a subsplit-clade $(s, \underline{Z})$, let the *rootward topologies of* $(s, \underline{Z})$, denoted by $\mathcal{T}_{\text{root}}((s, \underline{Z}))$, be the set of Z-unspecified topologies on X containing s. We emphasize that such a topology does specify topological structure for the non-Z side of the subsplit (often called the "sister clade" of Z). For example in Fig. 4, $\mathcal{T}_{\text{root}}(\grave{s})$ would be all of the C-unspecified topologies in the subsplit DAG containing s (such trees do specify structure for the sister clade of C, which in this case is the union of B_1 and B_2).

We can combine a leafward topology and a rootward topology and evaluate its probability dynamically using subsplits analogous to (8). Say we have subsplits t, s, and u such that that $\acute{s} \to u$ and $\grave{t} \to s$ (Fig. 4). Assume we are given $\tau_{\grave{t}}^{\uparrow} \in \mathcal{T}_{\text{root}}(\grave{t})$ and $\tau_u^{\downarrow} \in \mathcal{T}_{\text{leaf}}(u)$, and form $\tau_{\grave{s}}^{\uparrow} \in \mathcal{T}_{\text{root}}(\grave{s})$ by joining together $\tau_{\grave{t}}^{\uparrow}$ and $\tau_u^{\downarrow}$ using s. By definition of the structures, we have

$$\mathbb{P}(\tau_{\grave{s}}^{\uparrow}) = \mathbb{P}(\tau_{\grave{t}}^{\uparrow})\, \mathbb{P}(s \mid \grave{t})\, \mathbb{P}(u \mid \acute{s})\, \mathbb{P}(\tau_u^{\downarrow} \mid u). \tag{11}$$

The equivalent equation holds (using a suitable definition of u) when replacing $\grave{t}$ with $\acute{t}$, or $\grave{s}$ with $\acute{s}$. This is analogous to (8), but there are important differences. For example, (8) gives a probability $\mathbb{P}(\tau|t)$ conditioned on a subsplit t, but no such conditioning is present for (11).

Leafward traversal

We now use our PLVs $\mathbf{p}$ marginalizing over leafward trees to build the PLVs $\mathbf{r}$ marginalizing over rootward trees. This happens in a reverse post-order traversal, which we call the "leafward traversal".

First we need a version of the $\mathbf{r}$ vector that is defined for an element of $\tau_{s,\underline{Z}}^{\uparrow} \in \mathcal{T}_{\text{root}}((s, \underline{Z}))$:

$$\mathbf{r}(\tau_{s,\underline{Z}}^{\uparrow})_i := \mathbb{P}(\mathbf{Y}_{X \setminus Z}, Y_s = i \mid \tau_{s,\underline{Z}}^{\uparrow}) \tag{12}$$

where $\mathbf{Y}_{X \setminus Z}$ is the data for all taxa outside of Z and Y_s is the state of the node corresponding to subsplit s in the tree. This is equivalent to the definition (4) of $\mathbf{r}$ for any tree containing $\tau_{s,\underline{Z}}^{\uparrow}$ such that v is the root node of the subtree containing all the taxa of Z.

We formalize our topology-marginal version of $\mathbf{r}$ for subsplit-clades $(s, \underline{Z})$:

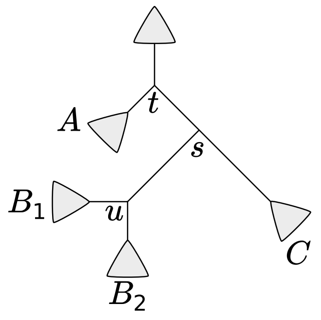


Fig. 4 The setting for dynamic computation during the leafward pass. Assume $A < B_1 < B_2 < C$ in the lexicographic ordering. Here $u = \{B_1, B_2\}$, $s = \{B_1 \cup B_2, C\}$, and $t = \{A, B_1 \cup B_2 \cup C\}$. We will use $\tau_{\grave{t}}^{\uparrow}$ to represent the partially-specified topology with all subsplits in clade A, t, and those towards the root from t. We will use $\tau_u^{\downarrow}$ to represent the topology leafward of u (including u itself)

$$\mathbf{r}((s,\underline{Z})) := \sum_{\tau_{s,\underline{Z}}^{\uparrow} \in \mathcal{T}_{\text{root}}((s,\underline{Z}))} \mathbf{r}(\tau_{s,\underline{Z}}^{\uparrow})\, \mathbb{P}(\tau_{s,\underline{Z}}^{\uparrow}). \tag{13}$$

The summand is the joint probability of observing $\tau_{s,\underline{Z}}^{\uparrow}$ in the DAG and $\mathbf{Y}_{X\setminus Z}$ (all of the sequences other than those in Z). There are important differences between this and (9): we use an unconditional $\mathbb{P}(\tau_{s,\underline{Z}}^{\uparrow})$ of a partially specified topology $\tau_{s,\underline{Z}}^{\uparrow}$ where before we used a conditional $\mathbb{P}(\tau_t^{\downarrow}|t)$ on a fully specified topology for the subset of taxa in t.

Rewriting (6) in DAG notation, we have

$$\mathbf{r}(\tau_{\grave{s}}^{\uparrow}) = \left(\mathbf{P}(\boldsymbol{\theta}(\acute{s} \to u))\mathbf{p}(\tau_u^{\downarrow})\right) \circ \left(\mathbf{P}^{\top}(\boldsymbol{\theta}(\grave{t} \to s))\mathbf{r}(\tau_{\grave{t}}^{\uparrow})\right). \tag{14}$$

With this we can derive the leafward-traversal version of (10). Recall that $\mathbf{r}(\tau_{\grave{s}}^{\uparrow})$ is of the form $\mathbf{r}((s,\underline{Z}))$, so the goal is to show that (13) decomposes into the element-wise product ($\circ$) of two terms.

Lemma 2

$$\mathbf{r}(\grave{s}) = \left(\sum_{u:\acute{s}\to u} \mathbf{P}(\boldsymbol{\theta}(\acute{s} \to u))\mathbf{p}(u)\, \mathbb{P}(u \mid \acute{s})\right) \circ \left(\sum_{\grave{t}:\grave{t}\to s} \mathbf{P}^{\top}(\boldsymbol{\theta}(\grave{t} \to s))\mathbf{r}(\grave{t})\, \mathbb{P}(s \mid \grave{t}) + \sum_{\acute{t}:\acute{t}\to s} \mathbf{P}^{\top}(\boldsymbol{\theta}(\acute{t} \to s))\mathbf{r}(\acute{t})\, \mathbb{P}(s \mid \acute{t})\right). \tag{15}$$

The same holds true exchanging $\grave{s}$ *and* $\acute{s}$.

We emphasize that this is a strict generalization of Felsenstein's pruning algorithm, because in the single-tree case the $\mathbb{P}$ are indicator functions so the sum collapses to a single term.

Proof We start with the right-hand side and show equality with the left-hand side. Because $\circ$ is linear, we can split (15) into the sum of two terms. We focus on the product of sums indexed by u and $\grave{t}$, and substitute in

$$\mathbf{p}(u) = \sum_{\tau_u^{\downarrow} \in \mathcal{T}_{\text{leaf}}(u)} \mathbf{p}(\tau_u^{\downarrow})\, \mathbb{P}(\tau_u^{\downarrow}|u)$$

and

$$\mathbf{r}(\grave{t}) = \sum_{\tau_{\grave{t}}^{\uparrow} \in \mathcal{T}_{\text{root}}(\grave{t})} \mathbf{r}(\tau_{\grave{t}}^{\uparrow})\, \mathbb{P}(\tau_{\grave{t}}^{\uparrow}).$$

Upon pulling sums out and rearranging the order of terms, we have

$$\sum_{\substack{u\,:\,\acute{s} \to u \\ \grave{t}\,:\,\grave{t} \to s}} \ \sum_{\substack{\tau_u^{\downarrow} \in \mathcal{T}_{\text{leaf}}(u) \\ \tau_{\grave{t}}^{\uparrow} \in \mathcal{T}_{\text{root}}(\grave{t})}} \left(\mathbf{P}(\boldsymbol{\theta}(\acute{s} \to u))\mathbf{p}(\tau_u^{\downarrow})\right) \circ \left(\mathbf{P}^{\top}(\boldsymbol{\theta}(\grave{t} \to s))\mathbf{r}(\tau_{\grave{t}}^{\uparrow})\right) \mathbb{P}(\tau_{\grave{t}}^{\uparrow})\, \mathbb{P}(s \mid \grave{t})\, \mathbb{P}(u \mid \acute{s})\, \mathbb{P}(\tau_u^{\downarrow}|u).$$

Similar to the proof of Lemma 1, there is a 1-to-1 correspondence between such $(\tau_{\grave{t}}^{\uparrow}, \tau_u^{\downarrow})$ pairs and topologies $\tau_{\grave{s}}^{\uparrow} \in \mathcal{T}_{\text{root}}(\grave{s})$, when restricted to those topologies $\tau_{\grave{s}}^{\uparrow}$ where $\grave{s}$ descends as the right subsplit of its parent: if we are given $(\tau_{\grave{t}}^{\uparrow}, \tau_u^{\downarrow})$ we simply join them together using s to make a $\tau_{\grave{s}}^{\uparrow} \in \mathcal{T}_{\text{root}}(\grave{s})$. This concludes the proof for the first term by (11), (14), and the definition of $\mathbf{r}(\grave{s})$. The second term, using $\acute{t}$, follows in exactly the same way, as does the statement when exchanging $\acute{s}$ and $\grave{s}$. Note that we need both of these terms because of where the subsplit s appears relative to the tree above it: it could appear as either a left or a right subsplit of the parent split. ☐

Implementing efficient computation

We now introduce additional notation to represent and store intermediate computations.

$$\check{\mathbf{p}}(\grave{s}) := \sum_{u:\grave{s}\to u} \mathbf{P}(\boldsymbol{\theta}(\grave{s} \to u))\mathbf{p}(u)\, \mathbb{P}(u \mid \grave{s}), \tag{16}$$

$$\check{\mathbf{p}}(\acute{s}) := \sum_{u:\acute{s}\to u} \mathbf{P}(\boldsymbol{\theta}(\acute{s} \to u))\mathbf{p}(u)\, \mathbb{P}(u \mid \acute{s}), \tag{17}$$

$$\begin{tikzcd}
& \text{$\mathbf{r}$-PLVs from parent subsplits} \arrow[d] & \\
\mathbf{r}(\grave{s}) & \breve{\mathbf{r}}(s) \arrow[l] \arrow[r] & \mathbf{r}(\acute{s}) \\
\breve{\mathbf{p}}(\acute{s}) \arrow[u] \arrow[r] & \mathbf{p}(s) & \breve{\mathbf{p}}(\grave{s}) \arrow[u] \arrow[l] \\
\text{$\mathbf{p}$-PLVs from children subsplits of $\acute{s}$} \arrow[u] & & \text{$\mathbf{p}$-PLVs from children subsplits of $\grave{s}$} \arrow[u]
\end{tikzcd}$$

Fig. 5 Dependency graph of partial likelihood vectors associated with a subsplit s, where $x \to y$ means that y depends on x

$$\breve{\mathbf{r}}(s) := \sum_{\grave{t}:\grave{t}\to s} \mathbf{P}^\top(\boldsymbol{\theta}(\grave{t} \to s))\mathbf{r}(\grave{t})\,\mathbb{P}(s \mid \grave{t}) + \sum_{\acute{t}:\acute{t}\to s} \mathbf{P}^\top(\boldsymbol{\theta}(\acute{t} \to s))\mathbf{r}(\acute{t})\,\mathbb{P}(s \mid \acute{t}). \tag{18}$$

With these definitions, (10) and (15) become

$$\mathbf{p}(s) = \breve{\mathbf{p}}(\acute{s}) \circ \breve{\mathbf{p}}(\grave{s}), \tag{19}$$

$$\mathbf{r}(\grave{s}) = \breve{\mathbf{p}}(\acute{s}) \circ \breve{\mathbf{r}}(s), \tag{20}$$

$$\mathbf{r}(\acute{s}) = \breve{\mathbf{p}}(\grave{s}) \circ \breve{\mathbf{r}}(s). \tag{21}$$

We use the dependency graph to calculate the ensemble of partial likelihood vectors (Fig. 5) associated with each subsplit s. We perform the *rootward traversal* to compute $\breve{\mathbf{p}}(\acute{s})$ and $\breve{\mathbf{p}}(\grave{s})$ using $\mathbf{p}(u)$ for $u : \acute{s} \to u$ and $u : \grave{s} \to u$, giving $\mathbf{p}(s)$ by (16), (17), and (19). Given these vectors our second pass, the *leafward traversal*, computes $\breve{\mathbf{r}}(s)$ via (18), (20), and (21).

The per-edge marginal likelihood

For a given edge $t \to s$, define the *per edge marginal likelihood* to be

$$\ell(t \to s; \boldsymbol{\theta}) := \sum_{\tau \ni (t \to s)} \mathbb{P}(\mathbf{Y} \mid \tau, \boldsymbol{\theta})\,\mathbb{P}(\tau \mid t \to s). \tag{22}$$

Here $t \to s$ is interpreted as $\grave{t} \to s$ or $\acute{t} \to s$, whichever is correct (as above). We are introducing a new notion, $\mathbb{P}(\tau \mid t \to s)$, which is a normalized probability restricted to the trees that contain an edge $t \to s$ of the DAG:

$$\mathbb{P}(\tau \mid t \to s) := \frac{\mathbb{P}(\tau)\,\mathbb{1}[t \to s \in \tau]}{\sum_{\tau' \in \mathcal{D}} \mathbb{P}(\tau')\,\mathbb{1}[t \to s \in \tau']}.$$

The numerator defines the joint probability of observing a tree and an edge,

$$\mathbb{P}(\tau, t \to s) := \mathbb{P}(\tau)\,\mathbb{1}[t \to s \in \tau].$$

The denominator is the unconditional probability of sampling a tree containing an edge $t \to s$:

$$\mathbb{P}(t \to s) := \sum_{\tau \in \mathcal{D}} \mathbb{P}(\tau)\,\mathbb{1}[t \to s \in \tau], \tag{23}$$

note that we sum over τ so this quantity is independent of the tree topologies. Now, (23) can be expressed as

$$\mathbb{P}(t \to s) = \mathbb{P}(t) \cdot \mathbb{P}(s|t), \tag{24}$$

where $\mathbb{P}(t)$ is the unconditional probability of sampling a tree containing a subsplit t.

We now show how to calculate $\ell(t \to s; \boldsymbol{\theta})$ efficiently using the components already described. A key part is that we can expand $P(\mathbf{Y}|\tau, \boldsymbol{\theta})$ over any edge $t \to s \in \tau$ as per the usual Felsenstein pruning algorithm by

$$\mathbb{P}(\mathbf{Y}|\tau, \boldsymbol{\theta}) = \mathbf{r}(\tau_t^\uparrow)^\top \mathbf{P}(\boldsymbol{\theta}(t \to s))\mathbf{p}(\tau_s^\downarrow); \tag{25}$$

marginalization leads to the following lemma.

Lemma 3

$$\ell(\grave{t} \to s; \boldsymbol{\theta}) = \mathbf{r}(\grave{t})^\top \mathbf{P}(\boldsymbol{\theta}(\grave{t} \to s))\mathbf{p}(s) \,/\, \mathbb{P}(\grave{t}). \tag{26}$$

The same holds true exchanging $\grave{t}$ for $\acute{t}$.

Proof First recall that for any τ containing $\grave{t} \to s$,

$$\mathbb{P}(\tau) = \mathbb{P}(\tau_{\grave{t}}^\uparrow)\,\mathbb{P}(s \mid \grave{t})\,\mathbb{P}(\tau_s^\downarrow|s)$$

where $\tau_{\grave{t}}^\uparrow$ is the $U(s)$-unspecified topology rootward of t, and $\tau_s^\downarrow$ is the topology on $U(s)$ leafward of s. Thus

$$\mathbb{P}(\tau \mid \grave{t} \to s) = \frac{\mathbb{P}(\tau, \grave{t} \to s)}{\mathbb{P}(\grave{t})\mathbb{P}(s|\grave{t})} = \frac{\mathbb{P}(\tau_{\grave{t}}^\uparrow)\mathbb{P}(s|\grave{t})\mathbb{P}(\tau_s^\downarrow|s)}{\mathbb{P}(\grave{t})\mathbb{P}(s|\grave{t})} = \frac{\mathbb{P}(\tau_{\grave{t}}^\uparrow)\mathbb{P}(\tau_s^\downarrow|s)}{\mathbb{P}(\grave{t})},$$

where $\mathbb{P}(\tau, \grave{t} \to s)$ is the joint probability of observing a tree τ and having it contain the edge $\grave{t} \to s$. Using this identity and Equation (25) to expand (22), we have

$$
\begin{aligned}
&\sum_{\tau \ni (\grave{t} \to s)} \mathbf{r}(\tau_{\grave{t}}^{\uparrow})^\top \mathbf{P}(\boldsymbol{\theta}(\grave{t} \to s))\mathbf{p}(\tau_s^{\downarrow})\, \mathbb{P}(\tau \mid \grave{t} \to s) \\
&= \sum_{\substack{\tau_{\grave{t}}^{\uparrow} \in \mathcal{T}_{\text{root}}(\grave{t}) \\ \tau_s^{\downarrow} \in \mathcal{T}_{\text{leaf}}(s)}} \mathbf{r}(\tau_{\grave{t}}^{\uparrow})^\top \mathbf{P}(\boldsymbol{\theta}(\grave{t} \to s))\mathbf{p}(\tau_s^{\downarrow})\, \mathbb{P}(\tau_{\grave{t}}^{\uparrow})\, \mathbb{P}(\tau_s^{\downarrow} \mid s) \,/\, \mathbb{P}(\grave{t}) \\
&= \left(\sum_{\tau_{\grave{t}}^{\uparrow} \in \mathcal{T}_{\text{root}}(\grave{t})} \mathbf{r}(\tau_{\grave{t}}^{\uparrow})\, \mathbb{P}(\tau_{\grave{t}}^{\uparrow}) \right)^\top \mathbf{P}(\boldsymbol{\theta}(\grave{t} \to s)) \left(\sum_{\tau_s^{\downarrow} \in \mathcal{T}_{\text{leaf}}(s)} \mathbf{p}(\tau_s^{\downarrow})\, \mathbb{P}(\tau_s^{\downarrow} | s) \right) \Big/ \mathbb{P}(\grave{t}) \\
&= \mathbf{r}(\grave{t})^\top \mathbf{P}(\boldsymbol{\theta}(\grave{t} \to s))\mathbf{p}(s) \,/\, \mathbb{P}(\grave{t}).
\end{aligned}
$$

□

Composite-like marginal likelihood

So far we have described computations for a single site and dropped the site from the notation. At this point we shift to notation that is explicit about the site under consideration.

As described at the beginning, assume we have M sites, which are indexed by $m \in \{1, \ldots, M\}$. Our work so far enables efficient computation of

$$\ell^m(\boldsymbol{\theta}) := \mathbb{P}(\mathbf{Y}^m \mid \boldsymbol{\theta}) = \sum_{\tau \in \mathcal{D}} \mathbb{P}(\mathbf{Y}^m \mid \tau, \boldsymbol{\theta})\mathbb{P}(\tau).$$

Namely, we can evaluate the exact marginal likelihood for each site by summing over the rootsplits t for fixed values of $\boldsymbol{\theta}$:

$$
\begin{aligned}
\ell^m(\boldsymbol{\theta}) &= \sum_{\tau \in \mathcal{D}} \mathbb{P}(\mathbf{Y}^m \mid \tau, \boldsymbol{\theta})\, \mathbb{P}(\tau) \\
&= \sum_{\text{rootsplits } t} \sum_{\tau_t^{\downarrow} \in \mathcal{T}_{\text{leaf}}(t)} \boldsymbol{\pi}^\top \mathbf{p}^m(\tau_t^{\downarrow})\, \mathbb{P}(t)\, \mathbb{P}(\tau_t^{\downarrow} \mid t) \\
&= \sum_{\text{rootsplits } t} \mathbb{P}(t)\, \boldsymbol{\pi}^\top \mathbf{p}^m(t).
\end{aligned}
$$

Here $\boldsymbol{\pi}$ is the distribution at the root and $\mathbf{p}^m$ is the p-PLV for the m-th site, which was simply $\mathbf{p}$ in the previous section. Similarly, we use $\mathbf{r}^m$ for the r-PLV at the m-th site.

One approach is to combine the per-site marginal likelihood to form a composite-like objective,

$$\ell(\boldsymbol{\theta}) := \prod_{m=1}^{M} \ell^m(\boldsymbol{\theta}). \tag{27}$$

In fact, we define a related notion of *per-edge composite likelihood*, and use it as an objective function when optimizing the branch length parameters associated with each edge $t \to s$. This per-edge composite likelihood is defined as the product of the per-edge marginal likelihoods over the sites,

$$\ell(t \to s; \boldsymbol{\theta}) := \prod_{m} \ell^m(t \to s; \boldsymbol{\theta}). \tag{28}$$

We can optimize the per edge composite marginal likelihood over branch lengths via a gradient-free method such as Brent optimization [20] since evaluating (28) is fast given the PLVs using Lemma 3 (for numerical stability, we instead optimize $\log \ell(t \to s; \boldsymbol{\theta})$). Next we note how gradient-based optimization is also possible.

Gradient-based optimization

Let $\partial_{t \to s}$ be the partial derivative of the component of the branch length vector $\boldsymbol{\theta}$ corresponding to $t \to s$. Thus

$$\partial_{t \to s} \log \ell(t \to s; \boldsymbol{\theta}) = \sum_{m=1}^{M} \frac{\partial_{t \to s}\, \ell^m(t \to s; \boldsymbol{\theta})}{\ell^m(t \to s; \boldsymbol{\theta})}.$$

By Lemma 3, we have,

$$\partial_{t \to s} \ell^m(t \to s; \boldsymbol{\theta}) = \frac{1}{\mathbb{P}(t)} (\mathbf{r}^m(t))^\top (\partial_{t \to s} \mathbf{P}(\boldsymbol{\theta}(t \to s)))\mathbf{p}^m(s).$$

Returning to the full likelihood optimization, we have

$$\begin{aligned}\partial_{t\to s}\log\ell(\boldsymbol{\theta}) &= \sum_{m=1}^{M}\frac{1}{\ell^m(\boldsymbol{\theta})}\left[\partial_{t\to s}\,\ell^m(\boldsymbol{\theta})\right]\\ &= \frac{\mathbb{P}(t\to s)}{\mathbb{P}(t)}\sum_{m=1}^{M}\frac{1}{\ell^m(\boldsymbol{\theta})}\left[(\mathbf{r}^m(t))^\top(\partial_{t\to s}\mathbf{P}(\boldsymbol{\theta}(t\to s)))\mathbf{p}^m(s)\right].\end{aligned}$$

Optimization in analogy with the single tree case

In the case of a single tree, it is typical for maximum-likelihood phylogenetic algorithms to proceed from edge to edge of the tree, optimizing the branch length for each edge. This is made efficient by the two-pass algorithm on the tree, where partial likelihood vectors on the tree can be calculated and then used for a constant-time function evaluation in the inner optimization loop. By optimizing the likelihood as parameterized by a single branch, they optimize the likelihood of data given the entire tree.

The setting for the DAG is related but somewhat different. We also have partial likelihood vectors, which can be calculated via a dynamic program, enabling efficient local inference. However, we do not have the guarantee that sequentially maximizing the per-edge marginal likelihood (22) will maximize the full composite likelihood (27). This is because improving the per-edge marginal likelihood for a given edge may decrease the per-edge marginal likelihood for another. In fact, this does occur in practice. Nevertheless, we have found that this algorithm works well for our purposes.

Optimization schedule

We begin by describing a slightly simplified version of the algorithm, and will then describe the full version. Given an initial branch length $\boldsymbol{\theta}$, we perform a rootward traversal to populate all of the $\mathbf{p}$-PLVs followed by a leafward traversal to populate $\mathbf{r}$-PLVs. We then perform a depth-first traversal as follows to optimize the branch lengths while keeping track of visited nodes with a set S so as to not re-visit nodes more than once. When we visit a given node t, we begin by updating $\breve{\mathbf{r}}(t)$ as the marginal of the $\mathbf{r}$ -PLVs of the parent subsplits. This is an important step as each subsplit node s updates $\breve{\mathbf{r}}(s)$ and passes down the results to its children, so that optimization for a edge has up-to-date information from other parts of the graph. When a non-trivial subsplit is visited, we descend into the left subsplit-clade followed by the right subsplit-clade (or vice versa) to keep it consistent. Specifically, we update $\mathbf{r}(\acute{t})$ using $\breve{\mathbf{r}}(t)$ and $\breve{\mathbf{p}}(\grave{t})$ computed from the previous optimization iteration. Then for each $\acute{t}\to s\in\mathcal{D}$, we optimize all branches below s and update $\mathbf{p}(s)$. Upon returning from the recursion on child s, we optimize $\boldsymbol{\theta}(\acute{t}\to s)$ by maximizing $\mathbf{r}(\acute{t})^\top\mathbf{P}(\theta)\mathbf{p}(s)$. We accumulate $\breve{\mathbf{p}}(\acute{t})$ using the optimized $\boldsymbol{\theta}(\acute{t}\to s)$ for each child s. Once the recursion on left subsplit-clade completes, we repeat the same procedure for $\grave{t}$; finally, we update $\mathbf{p}(t)=\breve{\mathbf{p}}(\acute{t})\circ\breve{\mathbf{p}}(\grave{t})$. The optimization is run until convergence is reached. Pseudocode for this simplified procedure is outlined in Algorithm 1.

However, this simple version has a shortcoming in that, because of the structure of the DAG, modification of one branch length can invalidate some of the PLVs used later in the procedure. To handle this, we perform a modified depth-first traversal on a version of the DAG in which each node can be marked with a "dirty" status. If a node s is dirty, then we assume that $\mathbf{p}(s)$, $\breve{\mathbf{p}}(\acute{s})$, and $\breve{\mathbf{p}}(\grave{s})$ are invalid. If we modify a branch length for a given edge of the subsplit DAG, then we must mark all of the ancestors of the nodes of that edge as dirty. If we require the PLV for a dirty node, then we perform a traversal down, performing updates without optimizing branch lengths, until the node is clean. Note that because we do not re-optimize branch lengths when "cleaning" a node, this cannot lead to optimization loops.

We draw parallels between our algorithm to the loopy belief propagation (LBP) [5]. Although the convergence is not guaranteed for LBP and there may be oscillations, the algorithm is known to produce accurate results through empirical studies, especially when the subgraphs have tree-like properties [21, 22]. In the experiments section, we show that the proposed parameter estimation scheme recovers branch lengths that are close to the truth.

Algorithm 1 Simplified generalized pruning algorithm optimization

```
1: procedure DepthFirst($t, S$)    ▷ $S$ is the set of visited nodes
2:     if $t$ is not a rootsplit then
3:         Update $\breve{\mathbf{r}}(t)$ (Equation (18))
4:     end if
5:     $\mathbf{r}(\acute{t}) \leftarrow \breve{\mathbf{r}}(t) \circ \breve{\mathbf{p}}(\grave{t})$
6:     $\breve{\mathbf{p}}(\acute{t}) \leftarrow 0$
7:     for $\{s : \acute{t} \to s \in \mathcal{D}, s \notin \text{leaf}(\mathcal{D}), s \notin S\}$ do
8:         DepthFirst($s, S$)
9:         $\boldsymbol{\theta}(\acute{t} \to s) \leftarrow \text{argmax}_\theta\, \mathbf{r}(\acute{t})^\top \mathbf{P}(\theta)\mathbf{p}(s)$
10:        $\breve{\mathbf{p}}(\acute{t}) \leftarrow \breve{\mathbf{p}}(\acute{t}) + \mathbf{P}(\boldsymbol{\theta}(t \to s))\mathbf{p}(s)\mathbb{P}(s|\acute{t})$
11:        $S \leftarrow S \bigcup \{s\}$
12:    end for
13:    $\mathbf{r}(\grave{t}) \leftarrow \breve{\mathbf{r}}(t) \circ \breve{\mathbf{p}}(\acute{t})$
14:    $\breve{\mathbf{p}}(\grave{t}) \leftarrow 0$
15:    for $\{s : \grave{t} \to s \in \mathcal{D}, s \notin \text{leaf}(\mathcal{D}), s \notin S\}$ do
16:        DepthFirst($s, S$)
17:        $\boldsymbol{\theta}(\grave{t} \to s) \leftarrow \text{argmax}_\theta\, \mathbf{r}(\grave{t})^\top \mathbf{P}(\theta)\mathbf{p}(s)$
18:        $\breve{\mathbf{p}}(\grave{t}) \leftarrow \breve{\mathbf{p}}(\grave{t}) + \mathbf{P}(\boldsymbol{\theta}(t \to s))\mathbf{p}(s)\mathbb{P}(s|\grave{t})$
19:        $S \leftarrow S \bigcup \{s\}$
20:    end for
21:    $\mathbf{p}(t) \leftarrow \breve{\mathbf{p}}(\grave{t}) \circ \breve{\mathbf{p}}(\acute{t})$
22: end procedure
23: Initialize $\boldsymbol{\theta}$
24: $(\mathbf{p}, \breve{\mathbf{p}}) \leftarrow$ RootwardTraversal($\mathcal{D}, \boldsymbol{\theta}$)
25: $(\mathbf{r}, \breve{\mathbf{r}}) \leftarrow$ LeafwardTraversal($\mathcal{D}, \mathbf{p}, \breve{\mathbf{p}}, \boldsymbol{\theta}$)
26: while Not Converged do
27:     $S \leftarrow \emptyset$
28:     for Rootsplits $t$ do
29:         DepthFirst($t, S$)
30:     end for
31: end while
32: return $\boldsymbol{\theta}$
```

Prior on subsplit parameters

As a prior on subsplit parameters, the simplest choice would be to set $\mathbb{P}(s \mid t) \propto 1$. However, this does not correspond to any previously described prior on topologies. Another option is to initialize $\mathbb{P}$ so as to induce a uniform prior on topologies appearing in the DAG. To that end, we must first compute $n(t)$, the number of topologies in the DAG descending from the subsplit t (we use the same notation for n applied to a subsplit-clade). Using this we can take a prior p_{pr} on rootsplits and edges:

$$\mathbb{P}_{\text{prior}}(t) := \frac{n(\acute{t})n(\grave{t})}{n_{\text{total}}} \qquad \mathbb{P}_{\text{prior}}(s|(t,\underline{Z})) := \frac{n(s)}{n((t,\underline{Z}))} \quad (29)$$

where n_{total} is the total number of topologies in the DAG. One can see that this leads to a uniform prior on topologies by expanding an arbitrary tree into its component subsplits. Computation of $n(t), n(\tilde{t}), n_{\text{total}}$ can be performed using a simple recursive algorithm.

Implementation and experiments

We implemented the generalized pruning algorithm in our Python-interface C++ library `bito` (https://github.com/phylovi/bito). In this section, we test accuracy, speed, and scalability of the generalized pruning implementation on phylogenetic datasets of various sizes. Our

Table 1 The Pearson correlation and mean absolute error between GP and VBPI estimates to the MCMC posterior means. VBPI outliers were defined as branch length estimates outside the 95% quantile

Dataset	Correlation			Mean absolute error		
	GP	VBPI	VBPI outliers omitted	GP	VBPI	VBPI outliers omitted
DS1	0.991	0.102	0.909	0.0009	0.0032	0.0012
DS3	0.999	0.999	0.999	0.0015	0.0007	0.0007
DS4	0.999	0.166	0.970	0.0012	0.0085	0.0019
DS5	0.990	0.930	0.920	0.0044	0.0050	0.0047
DS6	0.995	− 0.017	0.858	0.0011	0.0240	0.0013
DS7	0.999	0.999	0.999	0.0010	0.0003	0.0003
DS8	0.995	0.975	0.950	0.0015	0.0013	0.0012

experiments can be re-run as described in the section *Availability of data and materials* below.

Accuracy

We want to know if optimization of the composite likelihood using the generalized pruning algorithm could give a rapid and accurate estimate of part of the phylogenetic model. To that end, we compared branch length estimates between GP and those observed in posterior samples of a standard MCMC run on empirical data sets. The data sets, which we call DS1 and DS3-DS8, are standard data sets for evaluating MCMC methods on phylogenies (e.g., [1, 6, 7, 23]). We excluded DS2 from this study because it has an almost trivial posterior distribution [23]. The data sets consist of sequences from 27 to 67 species and are fully described in [1].

We used MrBayes 3.2 [24] to obtain posterior samples to serve as the ground truth for accuracy assessments. For each data set we ran 4 chains of 1,000,000,000 generations; sampled every 1000 generations; and used the first 100,000,000 generations as a burn-in to yield 900,000 samples from the posterior. The posterior branch length estimates were obtained by rooting topologies in the posterior and calculating the average branch length across topologies for each observed DAG edge. As we are interested in the accuracy of the estimation procedure developed here, rather than construction of the DAG, we constructed the DAG using the topologies found in the MrBayes posterior samples for the experiments.

We also assessed how competitive GP is against a state-of-the-art variational Bayesian phylogenetic inference (VBPI) method [9]. VBPI postulates variational approximations of the posterior using a product of two distributions, one for the topology and the other for the branch lengths. The topology component of the variational approximation is parameterized using subsplits; by supplying the same set of topologies as an input to GP and VBPI, we can ensure fairness in evaluation. VBPI optimizes the parameters of the two distributions iteratively using stochastic gradient descent, where each iteration involves a sampling step to draw phylogenetic trees (topology and branch lengths) from the variational distributions followed by a gradient step to optimize variational parameters. Upon convergence, the fitted variational distributions have shown to approximate the posterior distribution with high accuracy.

We applied VBPI on the same DS datasets to fit the parameters of the variational distribution and compared against GP for both speed and accuracy. We ran the VBPI implementation on each data set for 200,000 iterations, with 10 trees sampled per iteration for stochastic gradient descent, and output 1,000,000 trees sampled from the converged distribution. The 1,000,000 trees were processed to give average branch lengths in the same manner as the 900,000 trees sampled from the posterior.

We compared the posterior means to the estimates output by loading the same sample of topologies into GP and VBPI to obtain branch length estimates per edge of the DAG. Additionally, we calculated the coverage of the estimates to the 95% credible intervals, measured as the percentage of DAG edge branch length estimates that fall between the 2.5 and 97.5 percentiles in the posterior samples for each dataset. To ensure reliability of the posterior means and credible intervals, we discarded all edges appearing in fewer than 10 posterior samples.

Some care was required to ensure a fair comparison to the posterior for either method. The VBPI implementation places an exponential prior on the branch lengths, while our current implementation of GP implicitly assumes a uniform prior. We thus use MrBayes posterior samples that match these prior assumptions for the comparison. For comparison of GP to MrBayes, we use

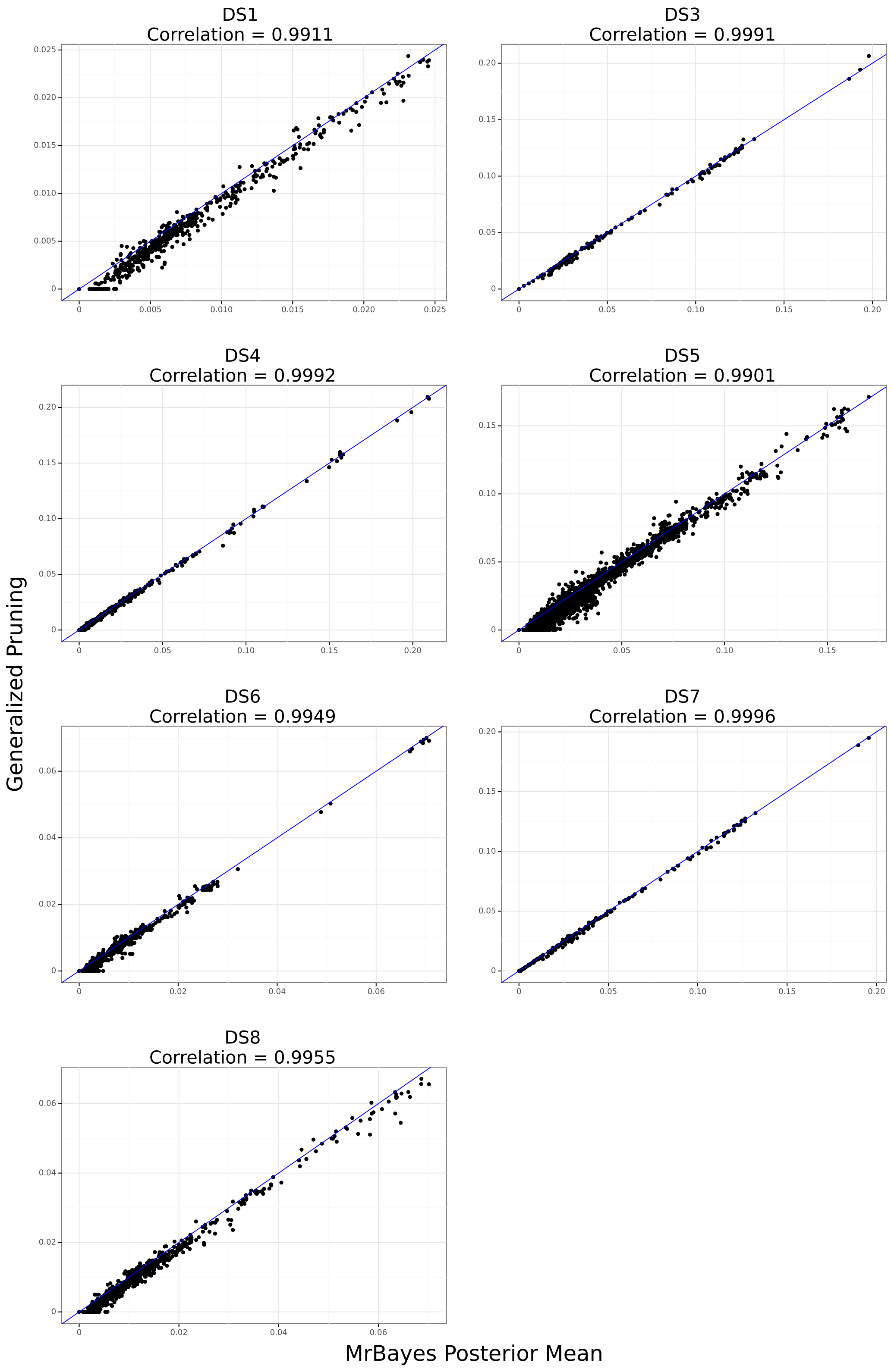


Fig. 6 Scatter plot of GP estimates vs MrBayes posterior means on the branch lengths for DS1, DS3, DS4, DS5, DS6, DS7, and DS8 for sDAG edges that appear in at least 10 posterior samples. The MrBayes posterior was sampled with a Uniform(0, 1) prior on branch lengths

Table 2 Coverage of GP and VBPI estimates in the 95% credible intervals

Dataset	GP	VBPI
DS1	0.946	0.926
DS3	0.983	1.000
DS4	0.958	0.915
DS5	0.907	0.921
DS6	0.709	0.870
DS7	0.849	0.997
DS8	0.893	0.950

Table 3 Timing results for executing GP and VBPI given a set of trees to build the subsplit DAG. GP runtimes are averages over 10 replicates of DAG initialization and branch length estimation

Dataset	DAG initialization (s)	GP estimation runtime (s)	VBPI runtime (s)
DS1	0.58	1.72	11, 336
DS3	0.19	0.38	15, 273
DS4	1.29	1.81	16, 867
DS5	62.11	4.90	19, 648
DS6	40.05	2.64	20, 420
DS7	1.72	0.67	25, 587
DS8	6.81	1.04	25, 725

posterior samples obtained with the MrBayes specifications as described earlier with the Uniform (0, 1) prior on the branch lengths. For comparison of VBPI to MrBayes, we use posterior samples obtained with those same settings, except with the Exponential (10) prior on the branch lengths. We do note that by aggregating branch length results by DAG edge, GP may have a slight inherent advantage because it optimizes branch lengths on a per-edge basis. In contrast, VBPI has a different and more complex parameterization for unrooted trees [9].

We compared the correlation and mean absolute error values between the estimates, shown in Table 1. We see that GP estimates closely align with those obtained from MCMC and offer improvements on both metrics compared to VBPI. Table 2 shows that GP and VBPI achieve similar coverage of the 95% credible interval across most datasets, with VBPI showing notably higher coverage for DS6. The scatter plots of GP vs the posterior means for each dataset are provided in Fig. 6. There is strong concordance between the GP estimates and the posterior means, from which we conclude that the composite likelihood optimization yields accurate estimates of the branch lengths. VBPI also mostly reports accurate estimates, although are some outliers for DS1, DS4, DS5, and DS6 (Additional file 1: Figure S1).

Speed and scalability

We wanted to evaluate the speed of the generalized pruning algorithm. We should note that VBPI is written in Python and calls PyTorch, whereas GP is written in C++, which can explain some of the speed difference. However recent benchmarking [25] shows that the automatic differentiation gradients used in VBPI are within an order of magnitude of carefully optimized implementations [26].

VBPI requires significantly longer runtime to converge compared to GP as shown in Table 3, but it is worth noting that VBPI learns branch length distributions rather than single point estimates, and additionally learns subsplit probabilities. We ran GP 10 times on each of the data set to obtain average run times estimates for the initialization of the DAG and estimation of the branch lengths. We measured DAG initialization as the time needed to build the subsplit DAG from the set of unique topologies found in the very large MrBayes posterior sample. This does not include inference of the trees used to build the subsplit DAG, nor does it include processing and deduplicating the raw MrBayes output into the format needed to build the DAG.

In order to understand the opportunities for scaling generalized pruning to large data sets, we measured the actual computation time for generalized pruning on a large subsample of sequences of the Makona variant Ebola virus [27]. We ran a single-threaded implementation of generalized pruning on an Intel Xeon E5-2667 Processor running Ubuntu 18.04 with 256 GB of RAM. This sample contained 1570 sequences and the subsplit DAG was built from a set of 1,000 trees obtained from an Ultrafast Bootstrap approximation using IQ-Tree [28]. The resulting subsplit DAG contained 42,305 nodes and 92,148 edges, which required 120 GB of virtual memory (allocated via a call to Linux `mmap`). Building the subsplit DAG required 25 min and 6 s, while branch length estimation required 2 full DAG traversals in 21 min and 26 s. There are further opportunities for efficient computations on large data sets, such as parallelization of branch length optimization in suitably distant edges in the DAG.

Availability of data and materials

All data, scripts, and instructions for reproducing the results presented are available at https://github.com/matsengrp/gp-benchmark-1. Readers can independently reproduce the MrBayes posterior samples, branch length estimates on the DS datasets for both VBPI and GP, and rerun GP on the Makona dataset. We note that MrBayes posterior samples on each dataset required multiple days. The repository includes copies of the posterior samples for reproduction without requiring the full MrBayes runs. Additionally, the repository includes copies of the raw output after VBPI and GP estimation used for figures and results presented in this manuscript.

Discussion

In this paper, we extend the Felsenstein pruning algorithm to a multi-tree phylogenetic model represented with a subsplit DAG. By using a dynamic program, we calculate partial likelihood vectors on the internal nodes of the DAG by marginalizing over topologies and molecular character states. This enables efficient calculation of a composite-like marginal likelihood that represents the per-site conditional probability of states given branch length parameters on the topologies observed in the DAG. We modify this likelihood by instead only marginalizing over topologies that contain a specific edge, defining the per edge marginal likelihood and the per edge composite likelihood under a standard assumption of independence over sites. The per edge composite likelihood serves as the objective function that allows us to make point estimates for branch lengths on the DAG edges. Under a Bayesian setting, these estimates were previously only accessible through a long MCMC run or, more recently, from related work on variational Bayesian phylogenetic inference [9].

We find that the branch length point estimates we obtain from generalized pruning accord with those obtained from a MrBayes posterior sample on standard phylogenetic datasets. Greater accordance is seen when we subset on edges that appear in a greater number of posterior topologies. Generalized pruning thus trades off accuracy on the low posterior probability edges for speed, leading to estimates in seconds compared to the hours required for convergence of MCMC.

One clear application to the branch length point estimates is for parameter initialization in variational Bayesian phylogenetic inference. Under that framework, branch lengths are estimated from a log-normal distribution with mean and variance parameters that are randomly initialized and updated during stochastic gradient descent. By replacing this random initialization with the point estimates obtained from generalized pruning, we should see faster convergence of the branch length variational distributions. In practice, the mean and variance parameters for a parent–child subsplit pair are themselves parameterized by subsplit-specific components. It remains to be determined how to appropriately initialize these subsplit components such that they result in edge branch length estimates given by the generalized pruning estimates.

The generalized pruning algorithm is, to our knowledge, the first algorithm to marginalize over tree topologies in a dynamic program. However, there are aspects of this work that connect with the "structural EM" algorithm [29], which leveraged the Chow-Liu approach [30] for phylogenetics. Specifically, that work approximates the full likelihood with the product of quantities, each of which can be evaluated edge-wise. Generalized pruning also enables per-edge optimization. However, the structural EM algorithm calculates an expected log-likelihood score marginalizing over ancestral states for pairs of nodes connected by a current tree, whereas generalized pruning marginalizes over alternative topologies and ancestral states simultaneously.

Our next step is to use generalized pruning as a means of inferring the subsplit DAG itself. In the same way that maximum-likelihood algorithms consider tree arrangements following nearest-neighbor interchanges (NNIs) to find the maximum-likelihood phylogeny, we may use a similar strategy for considering NNIs on the subsplit DAG. In this case, the per edge composite likelihoods could be used to accept or reject NNIs to append onto an initial DAG built from a small number of high likelihood topologies.

We also note that there is no obstruction to using non-reversible models for this algorithm— we do not need nor use the pulley principle— and the algorithms of [17] could be adapted here as well.

Supplementary Information

The online version contains supplementary material available at https://doi.org/10.1186/s13015-023-00235-1.

Additional file 1: Figure S1. Scatter plot of VBPI estimates vs MrBayes posterior means onthe branch lengths for DS1, DS3, DS4, DS5, DS6, DS7, and DS8 for sDAG edges that appear in at least 10 posterior samples. **Figure S2.** Scatter plot of VBPI estimates vs MrBayes posterior means on the branch lengths with outlier estimates removed for DS1, DS3, DS4, DS5, DS6, DS7, and DS8.

Acknowledgements
We thank Ognian Milanov for fixing a performance issue in the generalized pruning code, and Joe Felsenstein for discussions. This research was supported by NIH R01 AI162611 and AI153044. The research of Frederick Matsen was supported in part by a Faculty Scholar grant from the Howard Hughes Medical Institute and the Simons Foundation; FAM is an investigator of the Howard Hughes Medical Institute. Scientific Computing Infrastructure at Fred Hutch funded by ORIP grant S10OD028685. This article is subject to HHMI's Open Access to Publications policy. HHMI lab heads have previously granted a nonexclusive CC BY 4.0 license to the public and a sublicensable license to HHMI in their research articles. Pursuant to those licenses, the author-accepted manuscript of this article can be made freely available under a CC BY 4.0 license immediately upon publication.

Author contributions
SJ, HN, AK, DHR, and FAM developed the algorithm and implementation. SJ, HN, CJ-S, DHR, MF, CZ, MAS, and FA.M devised and implemented benchmarks. SJ, HN, CJ-S, MAS, and FAM wrote the main manuscript text. All authors reviewed the manuscript.

Funding
This work was supported through US National Institutes of Health grants R01 AI162611 and R01 AI153044. Scientific Computing Infrastructure at Fred Hutch was funded by ORIP grant S10OD028685. Dr. Matsen is an Investigator of the Howard Hughes Medical Institute.

Availability of data and materials
The implementation of the generalized pruning algorithm is available in our Python-interface C++ library `bito` (https://github.com/phylovi/bito). All data, scripts, and instructions for reproducing the results presented are available at https://github.com/matsengrp/gp-benchmark-1.

Declarations

Ethics approval and consent to participate
Not applicable.

Competing interests
The authors declare no competing interests.

Received: 30 December 2022 Accepted: 3 July 2023
Published online: 31 July 2023

References
1. Lakner C, van der Mark P, Huelsenbeck JP, Larget B, Ronquist F. Efficiency of Markov chain Monte Carlo tree proposals in Bayesian phylogenetics. Syst Biol. 2008;57(1):86–103. https://doi.org/10.1080/10635150801886156.
2. Felsenstein J. Evolutionary trees from DNA sequences: a maximum likelihood approach. J Mol Evol. 1981;17(6):368–76.
3. Baldauf SL. Phylogeny for the faint of heart: a tutorial. Trends Genet. 2003;19(6):345–51. https://doi.org/10.1016/S0168-9525(03)00112-4.
4. Salemi M, Lemey P, Vandamme AM. The phylogenetic handbook: a practical approach to phylogenetic analysis and hypothesis testing. Cambridge: Cambridge University Press; 2009.
5. Kschischang FR, Frey BJ, Loeliger H-A. Factor graphs and the sum-product algorithm. IEEE Trans Inf Theory. 2001;47(2):498–519.
6. Höhna S, Drummond AJ. Guided tree topology proposals for Bayesian phylogenetic inference. Syst Biol. 2012;61(1):1–11. https://doi.org/10.1093/sysbio/syr074.
7. Larget B. The estimation of tree posterior probabilities using conditional clade probability distributions. Syst Biol. 2013;62(4):501–11. https://doi.org/10.1093/sysbio/syt014.
8. Zhang C, Matsen FA IV. Generalizing tree probability estimation via bayesian networks. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems 31, pp. 1449–1458. Curran Associates, Inc., 2018. http://papers.nips.cc/paper/7418-generalizing-tree-probability-estimation-via-bayesian-networks.pdf
9. Zhang C, Matsen FA IV. Variational bayesian phylogenetic inference. In: international conference on learning representations (ICLR) 2019. https://openreview.net/pdf?id=SJVmjjR9FX.
10. Suchard MA, Weiss RE, Dorman KS, Sinsheimer JS. Inferring spatial phylogenetic variation along nucleotide sequences: a multiple changepoint model. J Am Stat Assoc. 2003;98(462):427–37.
11. Anisimova M, Gil M, Dufayard J-F, Dessimoz C, Gascuel O. Survey of branch support methods demonstrates accuracy, power, and robustness of fast likelihood-based approximation schemes. Syst Biol. 2011;60(5):685–99. https://doi.org/10.1093/sysbio/syr041.
12. Fourment M, Magee AF, Whidden C, Bilge A, Matsen FA IV, Minin VN. 19 dubious ways to compute the marginal likelihood of a phylogenetic tree topology. Syst Biol. 2020;69(2):209–20. https://doi.org/10.1093/sysbio/syz046.
13. Zhang C, Matsen FA IV A variational approach to bayesian phylogenetic inference 2022. arXiv:2204.07747
14. Ji X, Zhang Z, Holbrook A, Nishimura A, Baele G, Rambaut A, Lemey P, Suchard MA. Gradients do grow on trees: a linear-time o(n)-dimensional gradient for statistical phylogenetics. Mol Biol Evol. 2020. https://doi.org/10.1093/molbev/msaa130.
15. Redelings BD, Suchard MA. Joint Bayesian estimation of alignment and phylogeny. Syst Biol. 2005;54(3):401–18. https://doi.org/10.1080/10635150590947041.
16. Schadt EE, Sinsheimer JS, Lange K. Computational advances in maximum likelihood methods for molecular phylogeny. Genome Res. 1998;8(3):222–33.
17. Boussau B, Gouy M. Efficient likelihood computations with nonreversible models of evolution. Syst Biol. 2006;55(5):756–68. https://doi.org/10.1080/10635150600975218.
18. Kenney T, Gu H. Hessian calculation for phylogenetic likelihood based on the pruning algorithm and its applications. Stat Appl Genet Mol Biol. 2012;11(4):14. https://doi.org/10.1515/1544-6115.1779.
19. Wikipedia contributors: Newick format. https://en.wikipedia.org/w/index.php?title=Newick_format. Accessed: 2021-08-25 (2021). https://en.wikipedia.org/w/index.php?title=Newick_format
20. Brent RP. Algorithms for Minimization Without Derivatives. Courier Corporation, 2013.
21. Sudderth EB, Freeman WT. Signal and image processing with belief propagation [DSP applications]. IEEE Signal Process Mag. 2008;25(2):114–41.
22. Murphy K, Weiss Y, Jordan MI. Loopy belief propagation for approximate inference: An empirical study. 2013. arXiv:1301.6725
23. Whidden C, Matsen FA IV. Quantifying MCMC exploration of phylogenetic tree space. Syst Biol. 2015;64(3):472–91. https://doi.org/10.1093/sysbio/syv006.
24. Ronquist F, Huelsenbeck JP. MrBayes 3: Bayesian phylogenetic inference under mixed models. Bioinformatics. 2003;19(12):1572–4.
25. Fourment M, Swanepoel CJ, Galloway JG, Ji X, Gangavarapu K, Suchard MA, Matsen FA IV. Automatic differentiation is no panacea for phylogenetic gradient computation 2022. arXiv:2211.02168
26. Ayres DL, Cummings MP, Baele G, Darling AE, Lewis PO, Swofford DL, Huelsenbeck JP, Lemey P, Rambaut A, Suchard MA. BEAGLE 3: improved performance, scaling, and usability for a high-performance computing library for statistical phylogenetics. Syst Biol. 2019. https://doi.org/10.1093/sysbio/syz020.
27. Dudas G, Carvalho LM, Bedford T, Tatem AJ, Baele G, Faria NR, Park DJ, Ladner JT, Arias A, Asogun D, Bielejec F, Caddy SL, Cotten M, D'Ambrozio J, Dellicour S, Caro AD, Diclaro JW, Duraffour S, Elmore MJ, Fakoli LS, Faye O, Gilbert ML, Gevao SM, Gire S, Gladden-Young A, Gnirke A, Goba A, Grant DS, Haagmans BL, Hiscox JA, Jah U, Kugelman JR, Liu D, Lu J, Malboeuf CM, Mate S, Matthews DA, Matranga CB, Meredith LW, Qu J, Quick J, Pas SD, Phan MVT, Pollakis G, Reusken CB, Sanchez-Lockhart M, Schaffner SF, Schieffelin JS, Sealfon RS, Simon-Loriere E, Smits SL, Stoecker K, Thorne L, Tobin EA, Vandi MA, Watson SJ, West K, Whitmer S, Wiley MR, Winnicki SM, Wohl S, Wölfel R, Yozwiak NL, Andersen KG, Blyden SO, Bolay F, Carroll MW, Dahn B, Diallo B, Formenty P, Fraser C, Gao GF, Garry RF, Goodfellow I, Günther S, Happi CT, Holmes EC, Kargbo B, Keïta S, Kellam P, Koopmans MPG, Kuhn JH, Loman NJ, Magassouba N, Naidoo D, Nichol ST, Nyenswah T, Palacios G, Pybus OG, Sabeti PC, Sall A, Ströher U, Wurie I, Suchard MA, Lemey P, Rambaut A. Virus genomes reveal factors that spread and sustained the ebola epidemic. Nature. 2017. https://doi.org/10.1038/nature22040.
28. Minh BQ, Nguyen MAT, von Haeseler A. Ultrafast approximation for phylogenetic bootstrap. Mol Biol Evol. 2013;30(5):1188–95.
29. Friedman N, Ninio M, Pe'er I, Pupko T. A structural EM algorithm for phylogenetic inference. J Comput Biol. 2002;9(2):331–53. https://doi.org/10.1089/10665270252935494.
30. Chow C, Liu C. Approximating discrete probability distributions with dependence trees. IEEE Trans Inf Theory. 1968;14(3):462–7. https://doi.org/10.1109/TIT.1968.1054142.