Title

Finding high posterior density phylogenies by systematically extending a directed acyclic graph.

Permalink

https://escholarship.org/uc/item/358464jc

Journal

Algorithms for Molecular Biology, 20(1)

ISSN

1748-7188

Authors

Jennings-Shaffer, Chris

Rich, David

Macaulay, Matthew

et al.

Publication Date

2025-02-28

DOI

10.1186/s13015-025-00273-x

Peer reviewed

Algorithms for
Molecular Biology

Finding high posterior density phylogenies by systematically extending a directed acyclic graph

Chris Jennings-Shaffer[1], David H. Rich[1], Matthew Macaulay[2], Michael D. Karcher[3], Tanvi Ganapathy[1], Shosuke Kiami[1], Anna Kooperberg[1], Cheng Zhang[4], Marc A. Suchard[5,6,7] and Frederick A. Matsen IV[1,8,9,10*]

Abstract

Bayesian phylogenetics typically estimates a posterior distribution, or aspects thereof, using Markov chain Monte Carlo methods. These methods integrate over tree space by applying local rearrangements to move a tree through its space as a random walk. Previous work explored the possibility of replacing this random walk with a systematic search, but was quickly overwhelmed by the large number of probable trees in the posterior distribution. In this paper we develop methods to sidestep this problem using a recently introduced structure called the subsplit directed acyclic graph (sDAG). This structure can represent many trees at once, and local rearrangements of trees translate to methods of enlarging the sDAG. Here we propose two methods of introducing, ranking, and selecting local rearrangements on sDAGs to produce a collection of trees with high posterior density. One of these methods successfully recovers the set of high posterior density trees across a range of data sets. However, we find that a simpler strategy of aggregating trees into an sDAG in fact is computationally faster and returns a higher fraction of probable trees.

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

Introduction

Despite decades of work, Bayesian phylogenetics remains a computationally challenging problem. Existing methods are based on the Markov chain Monte Carlo (MCMC) algorithm. These methods begin with a tree, which may be random or generated by another method (e.g., parsimony), and propose random modifications to the tree. A random modification is accepted with probability proportional to the Metropolis-Hastings ratio of the new tree and current tree, which biases the chain towards trees with higher likelihood and prior probability. We use the term *topology* for the graph theoretic tree structure and *tree* for a topology with branch lengths. Because the high-confidence region of topologies is a tiny subspace of a space of super-exponential size, most of these random modifications will result in a significantly worse tree, and thus substantial modifications are overwhelmingly discarded. This leads to low acceptance rates, which may reduce efficiency. Thus, although MCMC is a robust and

flexible algorithm, it is inherently limited in its ability to efficiently infer phylogenetic posterior distributions.

Maximum likelihood methods take a different approach, and primarily work to find the maximum likelihood tree systematically rather than randomly. These methods are typically iterative and apply local rearrangements at each step to improve the likelihood of the current highest likelihood tree. These methods are substantially faster than MCMC methods, but do not attempt to characterize the entire credible set of possible trees or topologies.

Is it possible to combine these two approaches, in which one systematically infers an approximate Bayesian posterior distribution on trees? Although the phylogenetic likelihood can only be evaluated on a tree with branch lengths, one can define the likelihood of a topology to be the likelihood of the tree given by the topology along with optimal branch lengths. Previous work [14] performed a systematic and parallel exploration of tree space by local rearrangements of the visited topologies and collecting only the resulting topologies above some likelihood threshold. Normalizing the likelihoods of all visited topologies gives an approximation of the Bayesian posterior distribution. Although this formed an interesting proof of concept, it was ultimately defeated by there being too many high quality trees to do likelihood-based branch length optimization on each one.

More recent work [2, 3, 6, 16–18] has developed computational structures that are capable of storing and manipulating many trees or topologies at once. We use terminology from [6] and call this structure a *subsplit*

Directed Acyclic Graph or *sDAG* for short. Different bifurcating tree-graphs in the sDAG correspond to different tree topologies (Fig. 1).

In this paper, we develop systematic search strategies with sDAGs instead of topologies, with the goal of finding the smallest sDAG that contains a credible set of topologies. Because sDAGs can represent many topologies at once, we avoid the problem of having too many topologies to consider individually. Our approach in this work is to extend the nearest neighbor interchange (NNI) operation on topologies to an operation on sDAGs. As detailed in the Methods section, we develop NNIs as an operation to enlarge an sDAG rather than a move from topology to topology.

This approach requires a means of deciding whether an NNI is worth applying to the sDAG. One cannot directly apply classical phylogenetic criteria that judge a single topology at a time, because a single NNI operation can add many topologies to the sDAG at once (see Methods section). We apply two approaches. Both approaches associate branch lengths with each edge of the sDAG, which means that there is a one-to-one correspondence between topologies in the sDAG and trees in the sDAG.

The first approach, *top pruning*, implements the idea that one would like to apply NNIs that generate at least one good topology; this corresponds to the idea of collecting a credible set of topologies and merging them into a sDAG. In slightly more detail, top pruning maintains *choice maps* that can be used recursively to get a "best" tree containing the central edge of any given NNI, and branch lengths of new additions to the sDAG
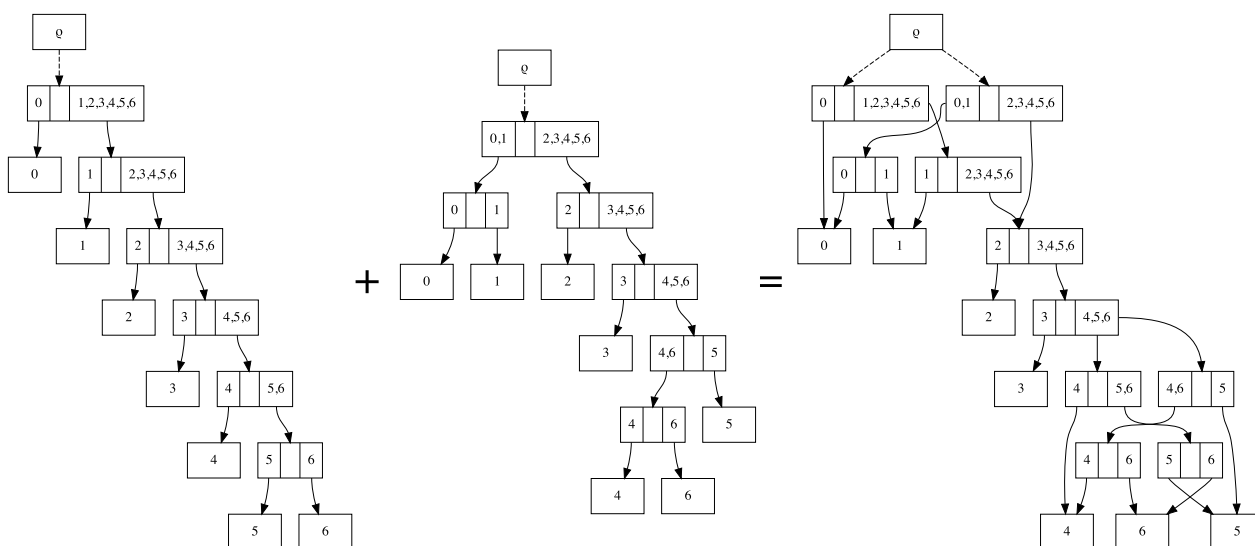
Fig. 1 The combination of two topologies τ_1 and τ_2 into a single sDAG. The sDAG (right) contains the union of the nodes and edges of the individual sDAGs from each of the topologies. It also contains additional topologies, such as the topology containing both $\{\{0\}, \{1, 2, 3, 4, 5, 6\}\}$ and $\{\{4, 6\}, \{5\}\}$, that are not present in the original set of two topologies

Jennings-Shaffer *et al. Algorithms for Molecular Biology*   (2025) 20:2

Page 3 of 18

are optimized to maximize the likelihood of the "best" tree containing the central edge of that NNI. Thus, the likelihood of NNIs in the top pruning case is the classical Felsenstein phylogenetic likelihood of that "best" tree associated to the NNI. (Here "best" is put in quotes because the "best" tree is an approximation to the maximum likelihood tree.)

The second, *generalized pruning*, or *GP*, implements the idea that one would like to apply NNIs that generate many probable topologies, but with a composite-like approximation to the marginal likelihood. One can use this marginal likelihood to optimize branch lengths for newly added sDAG edges, as well as to decide if an NNI is worth applying to the sDAG under the GP criterion. Specifically, the likelihood for an NNI is this GP marginal likelihood with optimized branch lengths. All of this is made possible by a recently-developed algorithm to calculate the marginal composite likelihood across topologies [6] for which computation time scales linearly in the number of edges of the sDAG.

When applied to benchmark data sets, we find that top pruning performs significantly better than generalized pruning in terms of discovering a credible set of topologies. However, neither method delivers a major advance in terms of finding a small subsplit DAG containing a credible set when compared to aggregating an sDAG from a short run of MrBayes [11]. Although this aggregation approach was taken to generate sDAGs in past work (e.g., [6, 16, 17]), here we show, using a variety of data sets, that this aggregation strategy gives good representation of the posterior distribution without being over-diffuse.

Methods

We begin by introducing the sDAG. Although it was described briefly as part of previous work [6], which assumed that such a structure was given, here our goal is to infer such a structure, so we will spend more time on developing and motivating the idea. Other recent independent work [2] has developed a related but different structure.

Introduction to the subsplit DAG

We first describe how the subsplit DAG is a generalization of a single, rooted, bifurcating topology, and how the general case can be considered the union of a collection of topologies. Take, for example, a single caterpillar topology on the taxon set $\{0, \ldots, 6\}$ (Fig. 1 left).

In this representation, we label each internal node with the two taxon sets that are leafward of each of the two edges coming from that internal node. So, for example, the second node below the root ρ has the taxon set $\{1\}$ on

one side and the taxon set $\{2, 3, 4, 5, 6\}$ on the other, so the internal node is labeled with $\{\{1\}, \{2, 3, 4, 5, 6\}\}$.

We call such a bipartition of a subset of the taxon set a *subsplit* (following [6, 16–18]; subsplits were called *partial splits* in [10]). Each component of the subsplit is a *clade* of the subsplit, and each of these clade components is referred to as a *subsplit-clade*. One can represent any rooted phylogenetic topology as an sDAG: each node is labeled with a subsplit describing the bipartition of the taxa leafward of that node, and each edge is directed in a leafward direction from the root. More generally, an sDAG is a directed acyclic graph with subsplits as nodes and edges connecting parent subsplits to child subsplits partitioning individual clades in the parent subsplit. To distinguish which subsplit-clade is partitioned along an edge, we write $(t, \underline{X}) \to s$, where X is a subsplit-clade of the subsplit t, s is a child subsplit of t, and $\bigcup(s) = X$. We underline the subsplit-clade X so that there is no ambiguity in which variables are subsplits and which are subsplit-clades. For convenience, we assume there is an order on the taxa, which extends to an order on clades. We call the lesser subsplit-clade the left subsplit clade and the greater subsplit-clade the right subsplit-clade.

We extend some common terminology for clades to sDAG edges. Suppose t_1, t_2, s_1, s_2 are subsplits, e_1 is the edge from t_1 to s_1, and e_2 is the edge from t_2 to s_2. When $s_1 = t_2$, we say e_1 is a parent edge of e_2 and say e_2 is a child edge of e_1; we may refine this by saying left child edge or right child edge, depending on which subsplit clade of s_1 is partitioned by s_2. When $t_1 = t_2$, but s_1 and s_2 partition distinct subsplit clades of t_1, we say e_1 and e_2 are sibling edges.

To build an sDAG encoding multiple topologies, we take the sDAG for each topology, then take the union of the nodes and edges in the two individual sDAG representations (Fig. 1). Thus an sDAG may contain a collection of topologies: any graph-theoretic-tree-structured subset of the nodes and edges in an sDAG that contains all of the leaves represents a tree topology. Each subsplit-clade of outdegree greater than one requires a choice between the descending arrows. For example, consider the edges leaving the subsplit-clade $\{4, 5, 6\}$ in Fig. 1. If we pick the edge leading to $\{\{4\}, \{5, 6\}\}$, we will have a tree with $\{4\}$ branching off first, and if we pick the edge leading to $\{\{4, 6\}, \{5\}\}$, we will have a tree with $\{5\}$ branching off first.

Note that if we build the sDAG from a collection of topologies, the sDAG may contain additional topologies beyond those used to build the sDAG (Fig. 1). In many respects this is a feature, not a bug: it allows us to expand the support of the sDAG combinatorially beyond the set of topologies used to build it. On the other hand this can add topologies outside the credible set. The balance

between the advantage of additional topologies and disadvantage of "false positive" topologies will be a key consideration in our systematic search strategies. However, as described next, we have finer control of the topology distributions than if we were to use the conditional clade distribution of [5] and [8].

Phylogenetic tree distributions

A key application of the sDAG is to represent a probability distribution on phylogenetic topologies and trees. If we assign such a probability distribution to the edges originating in each of the subsplit-clades, then we obtain a probability distribution on phylogenetic tree topologies. For example, in Fig. 1 we can assign probabilities to the two options for the root split and to the two options for resolving the subsplit-clade $\{4, 5, 6\}$: $\{\{4\}, \{5, 6\}\}$ and $\{\{4, 6\}, \{5\}\}$. Suppose the probability of $\{\{0, 1\}, \{2, 3, 4, 5, 6\}\}$ is 0.4 and the probability of $\{\{4\}, \{5, 6\}\}$ is 0.3, then the probability of the topology $((0, 1), (2, (3, (4, (5, 6)))))$ containing both of these subsplits is $0.4 \times 0.3 = 0.12$. It is easy to see that assigning probability distributions to each set of edges leaving each subsplit-clade in the sDAG yields a (normalized) probability distribution on the topologies represented in the sDAG.

Such a distribution is a generalization of previous work [5, 8] that led to topology distributions called *conditional clade distributions* (CCDs); CCDs consist of a distribution of subsplits conditioned on a clade, i.e. a set of taxa. Recently, these CCDs were attached to a directed acyclic graph structure similar to our sDAG [2]. The difference between this and the sDAG formulation is that the sDAG enables the expression of additional conditional dependencies on the sister clade (Fig. 2). We have shown such flexibility greatly improves fit [16]. However, if those are not important, one can allow them to be independent of the sister clade, recovering conditional clade distributions in a graph structure.

In terms of probabilities of edges, the sDAG formulation takes a probability distribution on edges leaving each parent subsplit-clade, whereas the conditional clade distribution requires these distributions to be identical for parent subsplit-clades on the same clade.

If one has a sample of topologies, such as that from an MCMC algorithm, one can use it to fit the probabilities labeling the edges of the sDAG. For an sDAG on rooted topologies, the probabilities for edges from a parent subsplit to its child subsplits are simply normalized frequency counts of the child subsplits in the sampled topologies that contain the parent subsplit. If we desire a distribution on unrooted topologies, we can consider the sDAG containing all possible rootings of the topologies
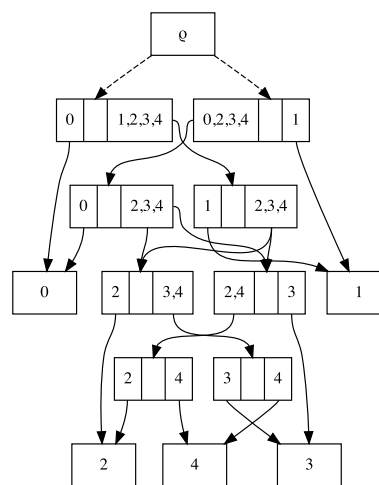
Fig. 2 An example showing how the sDAG is more flexible than the conditional clade distribution of [2, 5, 8]. Specifically, we may have different splitting probabilities for the clade $\{2, 3, 4\}$ depending on which subsplit it is contained in (either $\{\{0\}, \{2, 3, 4\}\}$ or $\{\{1\}, \{2, 3, 4\}\}$), showing that our approach is a strict generalization of clade-conditional approaches

in the sample, and an expectation-maximization algorithm can be used to infer probabilities [16].

To extend such a distribution on topologies to a distribution on trees, we attach parameterized distributions for branch lengths to the sDAG edges. Taking such an sDAG and inferring the branch length and subsplit distributions is a case of variational Bayesian phylogenetic inference. This approach was introduced and studied in [17, 18], taking the sDAG as a fundamental object, although it was described using different terminology: "subsplit Bayesian networks." However, in this paper, as described below, we will be assigning a single fixed branch length to each edge of an sDAG, so that each topology in an sDAG corresponds to a phylogenetic tree.

Performing NNIs to the subsplit DAG

The goal of this paper is to develop a systematic inference of the sDAG. In order to do so, we describe modifications of the sDAG that are analogous to the sort of modifications typically done to phylogenetic topologies. *Nearest-neighbor interchange* (NNI) is a common method to make minor topological modifications and grow the topological support. An NNI swaps two subtopologies of adjacent subsplits to create a new topology (see left-hand arrow of Fig. 3).

We can form an analogous operation on the sDAG, however, the NNI operation enlarges the sDAG rather than modifies it in place. That is, on an sDAG, we can perform NNI on two subsplit-clades, then combine the pre-NNI sDAG with the post-NNI sDAG into a single
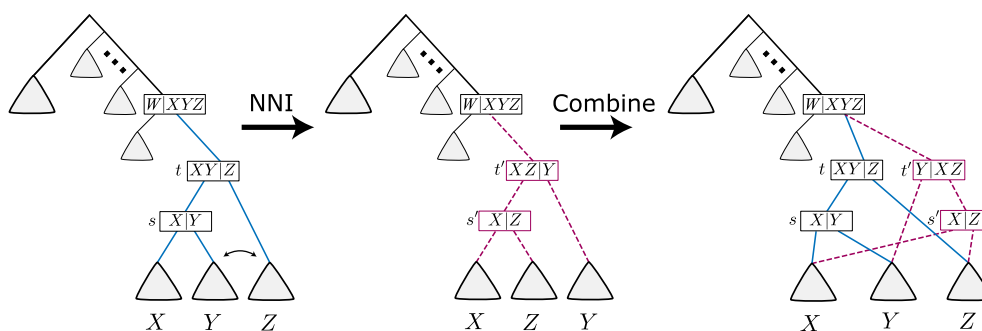
Fig. 3 Applying an NNI to a sDAG at subsplits t and s, then combining the pre-NNI sDAG with the post-NNI sDAG into a single sDAG. Here t' and s' are the new nodes, and the red dashed edges are the new edges. Triangles are substructures in the sDAG, and X, Y, and Z are clades

new sDAG. Consider Fig. 3, where t and s are subsplits on the clade set $X \cup Y \cup Z$, with $t = \{X \cup Y, Z\}$ and $s = \{X, Y\}$. Performing an NNI on clades Y and Z produces a new sDAG with subsplits t' and s', where $t' = \{X \cup Z, Y\}$ and $s' = \{X, Z\}$.

To construct the combined sDAG from the pre-NNI sDAG, we need only add subsplits t', s' (if not already in the sDAG) and edges (a) between the new nodes: $t' \to s'$, (b) parents of t': $u \to t'$ for all u where $u \to t$, and (c) descendants of the new subsplit-clades: $(s', \underline{X}) \to u$ for all u where $(s, \underline{X}) \to u$, $(t', \underline{Y}) \to u$ for all u where $(s, \underline{Y}) \to u$, and $(s', \underline{Z}) \to u$ for all u where $(t, \underline{Z}) \to u$. The new edges described above will preserve all previously existing incoming and outgoing edges from each clade. The addition of t' and s' creates a locally different splitting order between the X, Y, and Z clades, leaving all other parts of the sDAG unmodified. We refer to the edge $t' \to s'$ as the *central edge* of the NNI.

In our systematic inference methods, we maintain an sDAG with edges between all compatible subsplits, i.e. subsplits that can co-exist in a tree. Such sDAGs exhibit favorable properties. For example, when the pre-NNI sDAG has edges between all compatible subsplits, every new topology in the post-NNI sDAG is an NNI of a topology in the pre-NNI sDAG. Additionally, any new topology must contain the central edge of the NNI. Both statements can fail when the original sDAG is missing compatible edges. Details are given in the appendix. However, after applying an NNI to an sDAG, the resulting sDAG may not have the maximum number of edges even when the original sDAG does. In particular, the sDAG may not have edges from t' to all compatible child subsplits and edges from all compatible parent subsplits to s'. Thus, when enlarging an sDAG via an NNI in our systematic search algorithms, we will include these additional edges.

Evaluating new additions to the sDAG

Our inference algorithms proceed in a manner analogous to that for hill-climbing search of a single phylogenetic tree: evaluate all possible local modifications, and accept a modification according to an optimality criterion. Thus, we require a means of evaluating an NNI of the sDAG to decide if we should apply it to the sDAG. Specifically, we describe two ways that generalize the calculation of a likelihood for a phylogenetic tree. This is enabled by assigning a single fixed branch length to each edge of the sDAG, as described above, so that each topology in an sDAG corresponds to a phylogenetic tree.

Before describing these approaches in detail, we introduce some further notation. We write $\mathbf{Y}$ for the given multiple sequence alignment written as a rectangular array, each sequence is one row. The ith column of $\mathbf{Y}$ is $\mathbf{Y}^i$, which is the column vector containing the ith site of each sequence. As we deal only with fixed branch lengths, we write $p_\psi(\mathbf{Y} \mid \tau)$ for the phylogenetic likelihood for the data $\mathbf{Y}$, branch lengths ψ, and topology τ: that is, this is the classical phylogenetic likelihood of a tree that has τ as the topology and branch lengths assigned according to ψ, assuming data $\mathbf{Y}$. As commonly done, we assume site independence so that $p_\psi(\mathbf{Y} \mid \tau) = \prod_{i=1}^{K} p_\psi(\mathbf{Y}^i \mid \tau)$, where K is the number of sites. In this case, $p_\psi(\mathbf{Y}^i \mid \tau)$ is efficiently calculated by Felsenstein's pruning algorithm. In our implementation and benchmarking of our NNI-search algorithms, we use the Jukes-Cantor substitution model for simplicity. We could use a general time reversible model, as long as we take fixed model parameters (equilibrium frequencies and substitution rates) for the sDAG. One approach to infering these model parameters is to fit them on a single tree in the sDAG. We also do not consider across-site rate variation, although this could be added to both of the algorithms here if desired. Also, we optimize branch lengths rather than marginalize over them, in the interest of efficiency. In Bayesian

phylogenetics one typically considers a distribution of branch lengths, however this style of approximation has shown surprisingly good performance [1, 4, 13].

The two approaches, described in detail in the following two sections, are called "top pruning" and "generalized pruning." Each approach is based on its own definition of likelihood, described roughly above and in more detail in the following sections. However, for now, we note that generalized pruning is based on an across-tree marginalization so NNI operations are evaluated based on the sDAG as a whole, while top pruning is based on the (classical) likelihood of the single best additional tree enabled by the NNI.

Top Pruning

The idea behind top pruning is to add the highest likelihood tree, obtained by an NNI, not previously in the sDAG. Recall an NNI on an sDAG may introduce more than one new tree. Ideally one would select the NNI based on a computation like

$$\max_{\tau \in T_e} \prod_{j=1}^{K} p_\psi(\mathbf{Y}^j \mid \tau), \tag{1}$$

where T_e is the set of trees in the post-NNI sDAG with central edge e introduced by the NNI. However, this criterion does not yield an efficient algorithm, as we would need to enumerate all trees in T_e and compute their likelihoods.

Our approach is to instead store local choices of subtrees, which may yield a sufficient approximation to the tree maximizing the likelihood above. These local choices determine the topology and branch lengths, and so one can evaluate the likelihood of the tree in a classical way. In this section we focus on giving an intuitive understanding of how these local choices

work, and full details are found in the supplementary material.

The local choices of subtrees are implemented by a data structure we call a "choice map", which simply records a specific choice of neighboring edges for each edge of the sDAG (Fig. 4). We have a "rootward choice map", which when given an edge returns a parent edge and sibling edge (assuming the given edge does not begin with the universal ancestor). We have a "leafward choice map", which when given an edge returns a left child edge and right child edge (assuming the given edge does not end with a leaf).

These choice maps can be recursively applied, defining tree structures as follows. The leafward choice map defines a tree in the direction of the leaves from any given edge of the sDAG by recursively applying it until reaching the leaves. To get a tree in the direction of the root from any given edge of the sDAG, one uses the rootward choice map to pick edges toward the root, and the leafward choice map to choose edges descending from these edges which haven't already been determined. Combining these, we get a tree we call the "best known tree" for the edge. Although it may not be the maximum likelihood tree in the sDAG containing the edge, by the way we construct the choice map (described below) we believe it should be a good approximation.

The following is a more precise description of the deterministic process to find the best known tree for a given edge.

1. Initialize a graph G consisting of the single given edge.
2. While G is not a tree on the taxon set, we examine the edges of G.
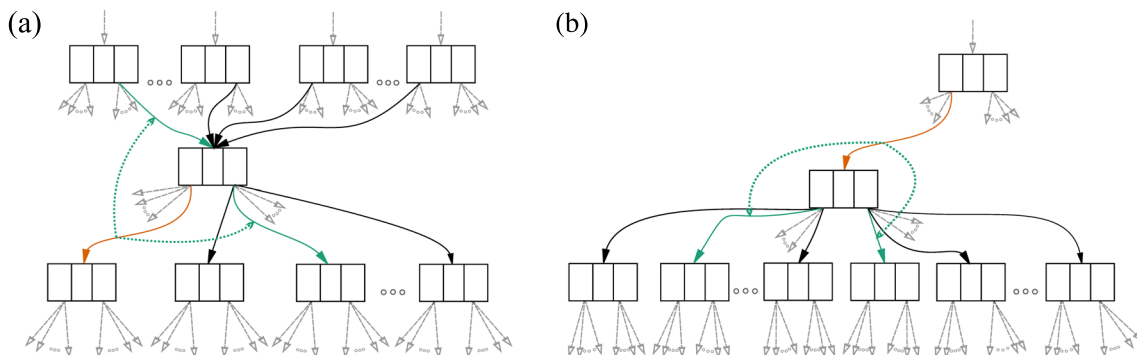
Fig. 4 An example of choice maps. In **a** the green solid edges are the parent and sibling edges returned by the rootward choice map (schematized by dashed line) for the given orange edge. In **b** the green solid edges are the left and right child edges returned by leafward choice map for the given orange edge. The "best known tree" for the given orange edge begins with the five highlighted edges: the orange edge common to both (**a**) and (**b**), the two solid green edges from (**a**), and the two solid green edges from (**b**)

(a) For each edge e not ending in a leaf: if G does not contain a left and right child edge of e, then we add the two children provided by the leafward choice map at e.

(b) For each edge e not originating at the universal ancestor: if G does not contain a parent and sibling edge of e, then we add the parent and sibling edges provided by the rootward choice map at e.

Furthermore, we can perform this construction after performing an NNI on the sDAG. If we cache partial likelihood vectors (PLVs) at the nodes of the sDAG, we can evaluate the likelihood of a best known tree in time that is constant in the number of leaves of the sDAG.

Next we explain which edges are selected for the choice maps. Suppose we generate an sDAG from a list of trees ordered by likelihood (the highest likelihood tree is first). To each edge of the sDAG, we assign the branch length from the first tree of the list containing the edge. Consider Fig. 4, where the given edge is highlighted in orange. In panel (a), we select the parent and sister edges highlighted in green; the two child edges are selected and highlighted in panel (b). The orange edge may appear in more than one tree of the original list, but since the trees are ordered by likelihood, we focus on the first (highest likelihood) tree with this edge. The four neighboring edges are taken from this tree and the choices are recorded in the choice maps. In particular, note that the choice maps depend on the likelihood-ordered list of trees used to construct the sDAG.

Next we will write out a complete example for an sDAG constructed from two input trees (Fig. 5). In this toy example we have trees τ_0 and τ_1 on seven taxa. We assume the classical setup for phylogenetic inference, with a sequence alignment and model, as well as branch lengths along the edges, so we can calculate the likelihood of a tree topology. Assume that τ_0 is of higher likelihood than τ_1. Figure 5a depicts τ_0, τ_1, and the sDAG spanned by the two. This sDAG contains two additional trees (Fig. 5b). The edges of the sDAG are labeled with the maximum likelihood input tree (τ_0 or τ_1) containing the edge. To build the best known tree for an edge, which need not be one of the input trees, we begin with the given edge and attach the immediately neighboring edges from the τ_i of the label. For the attached neighboring edges that touch neither root nor leaf, we must choose two additional edges (either children or parent and sibling) to flesh out the best known tree. Such additional edges are taken from the input tree (again τ_0 or τ_1) given by the label of the attached edge. We continue this process until a tree is fully constructed. The best known tree for the edges of the sDAG labeled τ_0 is τ_0. The best known tree for edges

labeled τ_1 is τ_2 for edges above the subsplit $\{\{1\}, \{2, 3, 4, 5\}\}$ and τ_3 for edges below.

For most edges in the sDAG of Fig. 5, there is only one choice for parent, sibling, and child edges. The non-trivial choices can be phrased as:

- Does $\{\{1\}, \{2, 3, 4, 5\}\} \rightarrow \{1\}$ use the parent edge from $\{\{1, 2, 3, 4, 5\}, \{6\}\}$ or $\{\{0\}, \{1, 2, 3, 4, 5\}\}$?
- Does $\{\{1\}, \{2, 3, 4, 5\}\} \rightarrow \{\{2\}, \{3, 4, 5\}\}$ use the parent edge from $\{\{1, 2, 3, 4, 5\}, \{6\}\}$ or $\{\{0\}, \{1, 2, 3, 4, 5\}\}$?
- Does $\{\{1\}, \{2, 3, 4, 5\}\} \rightarrow \{\{2\}, \{3, 4, 5\}\}$ use the right child edge to $\{\{3\}, \{4, 5\}\}$ or $\{\{3, 4\}, \{5\}\}$?
- Does $\{\{2\}, \{3, 4, 5\}\} \rightarrow \{2\}$ use the sibling edge to $\{\{3\}, \{4, 5\}\}$ or $\{\{3, 4\}, \{5\}\}$?

In Fig. 5c we highlight the edge $\{\{1\}, \{2, 3, 4, 5\}\} \rightarrow \{1\}$ in orange along with its chosen neighbor edges. Additionally, we highlight $\{\{1\}, \{2, 3, 4, 5\}\} \rightarrow \{\{2\}, \{3, 4, 5\}\}$ in green and $\{\{2\}, \{3, 4, 5\}\} \rightarrow \{2\}$ in purple, as well as their chosen neighbor edges.

The key point is each edge of the sDAG is assigned a tree systematically. Importantly, we can extend this to assign trees to the central edges of NNIs of the sDAG. The details of extending the choice maps to such edges are given in the supplementary materials. Furthermore, this assignment of trees allows for efficient phylogenetic likelihood calculations. We define the "top pruning likelihood" of an NNI to be the likelihood of the best known tree for the central edge of the NNI.

The top pruning algorithm proceeds in the following manner. Suppose we are given a list of trees, ordered by likelihood.

1. Initialize $\mathcal{D}$ to the sDAG spanned by the topologies from the list of trees.
2. Assign branch lengths to the edges of $\mathcal{D}$ by taking the branch length from the first tree in the list containing a given edge.
3. Initialize choice maps for the edges of $\mathcal{D}$.
4. Add edges between all compatible parent and child subsplits. Assign choice maps with an argmax strategy (see Supplement, equation (4)). Assign branch lengths optimizing best known trees for these edges.
5. Create a list of NNIs, ordering by top pruning likelihood, for each edge in the sDAG (two NNIs per sDAG edge). We do not record NNIs in the list if they do not enlarge $\mathcal{D}$.
6. Enlarge $\mathcal{D}$ to $\mathcal{D}'$ with the highest top pruning likelihood NNI of the list. Assign choice maps and branch lengths as in (3).

(a)

(b)

(c)
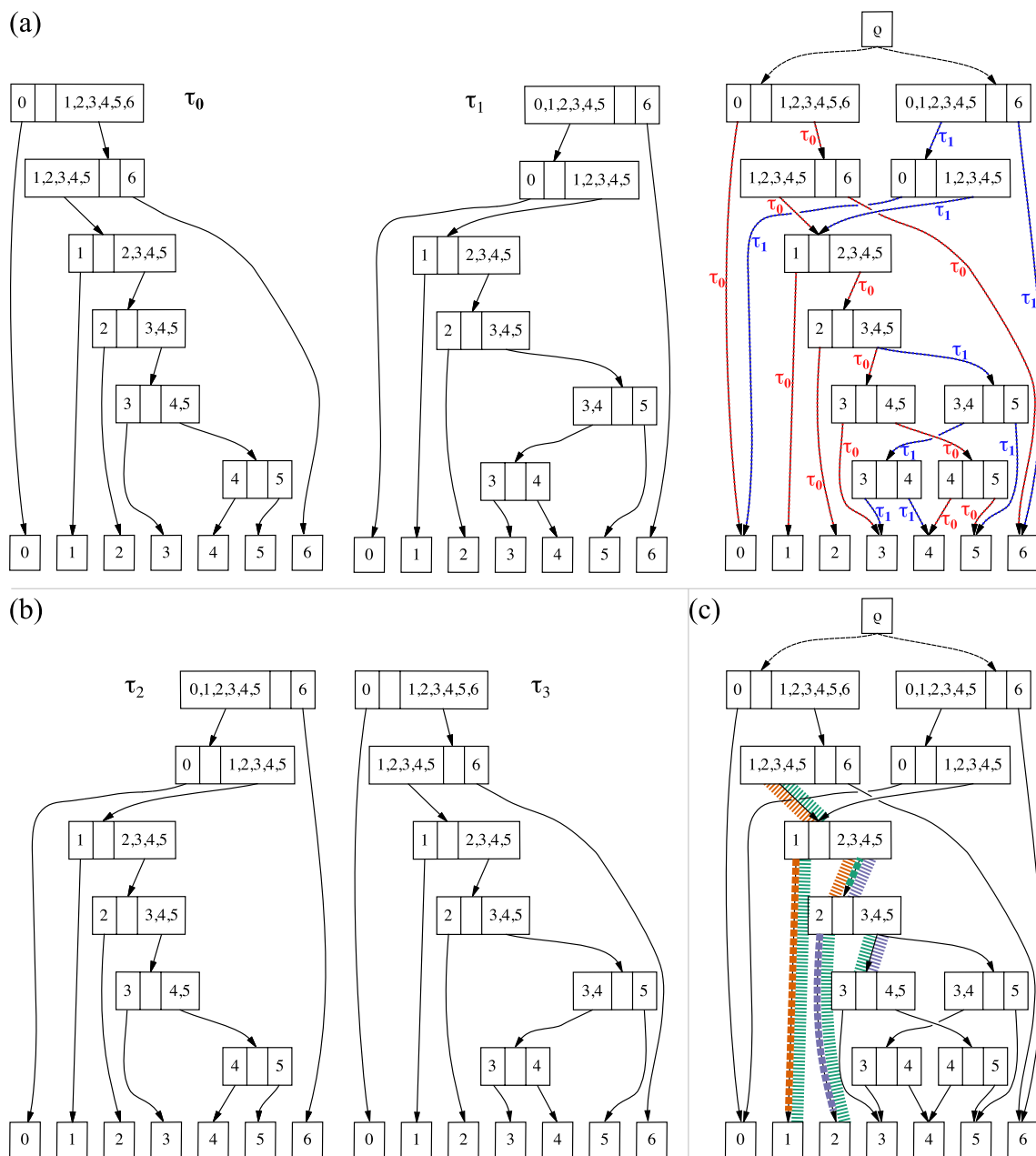
Fig. 5 Choice map initialization for an sDAG, assuming τ_0 has higher likelihood than τ_1. In **a** are two trees on seven taxa and the sDAG built from the trees. The edges of the sDAG are labeled with the topology, τ_0 in red or τ_1 in blue, where the edges first appeared. In **b** are the two additional topologies in the sDAG. In **c** we highlight the edges that have an option in the choice maps. In orange is an edge (bold dashed line) and its chosen neighbors (lightweight dashed line), with a choice of parent edge; in purple is an edge and its chosen neighbors, with a choice of right child edge; and in green is an edge and its chosen neighbors, with a choice of parent edge and right child edge

7. Enlarge $\mathcal{D}'$ to $\mathcal{D}''$ by adding all additional edges between compatible subsplits. Assign choice maps and branch lengths as in (4).

8. Remove the NNI of step 6 from the list.

9. Insert into the list, while maintaining the order by top pruning likelihood, the new NNIs that enlarge

$\mathcal{D}''$. The top pruning likelihood is not updated for NNIs already present in the list.

10. Return to step 6 with $\mathcal{D}''$ in place of $\mathcal{D}$.

The looping step of the algorithm repeats either for a fixed number of iterations or until the likelihood of all NNIs in the list are below a given threshold.

Generalized pruning

The generalized pruning (GP) objective applies the NNI to an sDAG that most increases a topology-marginal composite likelihood called the generalized pruning likelihood [6]. Specifically, let $\mathcal{D}_e$ be the sDAG from applying an NNI with central edge e. Let T_e be the set of trees of $\mathcal{D}_e$ that contain e. The GP algorithm applies the NNI that maximizes

$$\prod_{j=1}^{K} \sum_{\tau \in T_e} p_\psi(\mathbf{Y}^j \mid \tau)\, p(\tau \mid e), \tag{2}$$

across edges e. The term $p(\tau \mid e)$ is the relative prior probability of τ among the topologies of T_e. Explicitly,

$$p(\tau \mid e) = \frac{p(\tau)}{\sum_{\tau' \in T_e} p(\tau')},$$

where $p(\tau)$ is a prior distribution on topologies. Our implementation uses the uniform distribution, so that $p(\tau \mid e) = \frac{1}{|T_e|}$. The likelihood in (2) is the *per-edge marginal likelihood* introduced in [6], the details of which are given in the section of the same name. Specifically, this is the generalized pruning per-edge composite marginal likelihood of the edge e in the sDAG after applying the NNI. We call this likelihood the generalized pruning likelihood of the NNI.

Note that we call (2) a "composite likelihood" because it is taking a product of per-site marginal likelihoods rather than marginalizing over the complete likelihood. This correct marginal likelihood of the edge would be

$$\sum_{\tau \in T_e} p(\tau \mid e) \prod_{j=1}^{K} p_\psi(\mathbf{Y}^j \mid \tau).$$

However, there are no efficient means known of directly calculating this correct marginal likelihood, whereas the generalized pruning version scales linearly in the size of the sDAG. The generalized pruning marginal likelihood is computed with a traversal of the sDAG that populates partial likelihood vectors at each node, where each node has a PLV for each site of the sequence alignment. In contrast, to compute the true marginal likelihood, each node would require a PLV not just for each site, but each site and distinct topology with that node. That is to say, the GP likelihood is computed using PLVs that are shared by topologies, while the true marginal likelihood would require separate PLVs for topologies. Although it is not the same as the true marginal likelihood, we have found

that the GP likelihood is a sufficient approximation of the true likelihood for the purpose of optimizing branch lengths [6].

Suppose we are given a list of starting trees ordered by likelihood, the generalized pruning systematic inference algorithm is as follows.

1. Initialize $\mathcal{D}$ to the sDAG spanned by the topologies from the list of trees.
2. Assign branch lengths to the edges of $\mathcal{D}$ by taking the branch length from the first tree in the list containing a given edge. Optionally, we further optimize the branch lengths to maximize the overall generalized pruning likelihood of the sDAG.
3. Add edges between all compatible parent and child subsplits; assign GP per-edge composite marginal likelihood optimized branch lengths to these edges.
4. Create a list of NNIs, ordering by generalized pruning likelihood, for each edge in the sDAG (two NNIs per sDAG edge). We do not record NNIs in the list if they do not enlarge $\mathcal{D}$.
5. Enlarge $\mathcal{D}$ to $\mathcal{D}'$ with the highest GP likelihood NNI of the list. Add any additional edges between either of the new subsplits and compatible existing subsplits. Assign GP per-edge composite marginal likelihood optimized branch lengths to the new edges.
6. Remove the NNI of step 5 from the list.
7. Insert into the list, while maintaining the order by GP likelihood, the new NNIs that enlarge $\mathcal{D}''$. The GP likelihood is not updated for NNIs already present in the list.
8. Return to step 5 with $\mathcal{D}'$ in place of $\mathcal{D}$.

The looping step of the algorithm repeats either for a fixed number of iterations or until the GP likelihood of all NNIs in the list are below a given threshold. The GP likelihoods in items 4 and 7 are calculated after optimizing branch lengths.

Implementation of systematic inference

The necessary functionality for both NNI-searches are implemented in the Python-interface C++ library `bito`(https://github.com/phylovi/bito) and an interface to perform a search is further implemented in Python (https://github.com/matsengrp/sdag-nni-experiments). Both top pruning and generalized pruning use PLVs for fast and efficient likelihood calculations. The PLVs for top pruning are defined as usual for a two-pass version of Felsenstein's pruning algorithm and are propagated along the choice maps. The PLVs for generalized pruning follow a different pattern and are discussed in detail in [6]. It is these likelihoods and associated PLVs that dominate the computational expense of our algorithms, while

maintaining the the graph structure of the current sDAG and potential additions is secondary.

When new edges are introduced for top or generalized pruning, the optimization of associated branch lengths is done as follows. We take one branch length, hold the others fixed, apply Brent optimization (maximizing the likelihood of the edge in terms of best known tree likelihood or generalized pruning), repeat with another branch length, and continue until values have approximately converged. We leave the branch lengths of old edges unaltered. This approach showed good performance for branch length optimization in previous work [6].

Benchmarking setup

As in previous work (e.g., [4, 5, 7, 8, 15]), we compare our inferences to very long MrBayes "golden runs". We use common benchmark data sets, which we call the DS-datasets, as well as a 100 influenza A (flu) sequence data set from [18]. Each very long run of MrBayes yields numerous topologies and posterior density estimates of these topologies, which form the empirical posterior. We take the 95% credible set to be the minimal size set of topologies whose cumulative density is at least 95%. Basic information for the data sets and their empirical posteriors is in Table 1. We say a subsplit is credible if it appears in at least one topology of the 95% credible set. This is different from taking a credible set of subsplits from a topology marginalized probability on subsplits.

We will use the term "diffuse" to qualitatively describe the size of the posterior credible set. Taking the numbers in Table 1 (either for credible topologies or posterior topologies) the order of data sets from least to most diffuse is DS3, DS1, DS4, DS7, DS8, DS6, and DS5. The data sets DS1, DS3, DS4, and DS7 are not diffuse, while DS5 and DS6 are very diffuse, and DS8 is somewhere between. While there is another data set, commonly called DS2, the posterior is only a few topologies and so we ignore it here.

Table 1 Description of the DS-datasets

Data set	Taxa	Sites	95% Credible set Topology count	Empirical posterior Topology count
1	27	1949	42	1245
3	36	1812	16	240
4	41	1137	219	4539
5	50	378	260894	298768
6	50	1133	157942	195816
7	59	1824	756	6000
8	64	1008	4329	26442
flu	100	1681	16475	20262

For each data set, we calculate how much of the posterior density is captured per NNI-search iteration and by run time. We use short runs of MrBayes for a direct comparison of run time. These short runs of MrBayes use similar specifications to the golden runs, except we record all topologies (i.e., sample frequency 1 and no burn-in). With these short runs, we have two comparisons with an NNI-search. First, we compare the cumulative posterior density of the topologies from an NNI-search to that of the topologies from the short run. While this is a fair comparison in terms of run time, comparing the density from topologies of an sDAG to topologies from a simple list puts the short runs at a disadvantage. Our second comparison addresses this issue by comparing the density of topologies from an NNI-search to the density of the topologies in the sDAG spanned by the topologies from the short run of MrBayes. This gives a more accurate comparison in terms of cumulative density, as we compare an sDAG to an sDAG. We emphasize that while our systematic inference algorithms produce sDAGs that represent trees, we ignore branch lengths and work with topologies for these comparisons. While we can efficiently build an sDAG from a large list of topologies, there is required processing of topologies from MrBayes before building the sDAG (rerooting, ordering of splits of subtopologies in the newick string, etc.). When memory is not a constraint, this processing of topologies is also quick and efficient. However, for a very large number of topologies (as is the case with DS5), we use a less efficient method. The run-time we report for the sDAG spanned by topologies of MrBayes includes the run-time for MrBayes, the estimated run-time of processing topologies with the efficient implementation, and the run-time for constructing the sDAG from the processed topologies. As discussed in the following section, generalized pruning performs poorly and so we do not show it in every benchmark.

We start all of the searches (top pruning, generalized pruning, and short runs of MrBayes) with the maximum posterior density topology of the golden run with branch lengths optimized by iqtree for likelihood. Starting the searches at the maximum posterior density topology is a best-case scenario for performance, as the search starts at the highest peak of the distribution.

We also compare top pruning and the short runs of MrBayes by the quality of subsplits. We calculate both how many of the credible subsplits are found by a search and how many of the subsplits found by a search are credible. The comparison with top pruning and short runs of MrBayes matches the two on the number of subsplits encountered. In the ideal setting, the search methods would only add subsplits present in the posterior.

Our search methods can take multiple trees as input, so we investigate two additional starting points. The first is a "best case" minor variation: we begin the search with the top 10 posterior density topologies with likelihood optimized branch lengths. The second is a more realistic setting and randomized: we take the unique topologies produced by 200 runs of RAxML [12] with likelihood optimized branch lengths. We find that RAxML produces more distinct topologies than iqtree when given random starting trees (data not shown). We compare the performance with different starting trees by cumulative density per iteration. There is not a comparison with short runs of MrBayes. It is worth noting that we may use the trees from RAxML without any knowledge of the credible set or an empirical posterior distribution.

For consistency, all timing scripts were run with the SLURM job scheduler with exclusive access to a single node. The experiments of this section can be recreated by following the instructions in the experiments GitHub repository   (https://github.com/matsengrp/sdag-nni-experiments).

Results

We first describe the disappointing performance of generalized pruning, and then the somewhat better performance of top-pruning. For top-pruning we also provide a more detailed analysis, discussing run time, performance with multiple starting points, and performance on the larger flu data set. For this first set of analyses, we will be interested in the total posterior density of the topologies present in the sDAG generated by the various methods through time.

The performance of generalized pruning is lackluster. Regardless of data set, generalized pruning fails to capture a reasonable amount of posterior density in a reasonable amount of time (Fig. 6 for DS1 and DS3-6, Figure S1 for DS7-8). It is not competitive with the list of topologies from the short runs of MrBayes. Even on DS3, the least diffusive data set, generalized pruning spends many iterations adding subsplits and edges not contributing to the density.

Top pruning fares better than generalized pruning on the DS-datasets. It consistently outperforms the topologies of a short MCMC run and is competitive with the sDAG spanned by these topologies. While top pruning experiences long lengths of time with little to no increase in coverage of the posterior density, it usually finds its way back and captures much of the posterior density (Fig. 7 for DS1 and DS3-6, Figure S2 for DS7-8).

The credibility of subsplits found by top-pruning tends to be worse than the short runs of MrBayes (recall that we say that a subsplit is credible if it appears in at least one topology of the 95% credible set). Top pruning and MrBayes are very similar on DS3 and DS7, but on the other datasets top pruning adds many non-credible subsplits along with credible subsplits (Fig. 8). The orange (magenta) lines are essentially the true-positive rates of top pruning (short MCMC, respectively) in terms of subsplits. While not included here, we performed the same analysis with sDAG edges rather than subsplits and the results were similar.

Multiple starting trees are beneficial, but the benefit depends on the trees. Using the 10 highest probability trees from the credible set, or high likelihood trees generated by RAxML, of course, provides the gain in cumulative density of the additional topologies (Fig. 9). Past that, performance improvements are more subtle. Suppose we take the sDAG formed by the multiple initial trees and compare with the sDAG from running top pruning, without the additional trees, until we have approximately the same cumulative density of topologies in the sDAGs. After applying a few iterations of top pruning to both of these sDAGs, one may have a higher cumulative density of topologies than the other. Using the 10 highest probability trees usually does not give this kind of improvement, while using the high likelihood RAxML trees often does. That is, using the trees from RAxML improves the selection of NNIs toward the beginning of the search, while using the 10 highest probability trees does not. One possible explanation is the top 10 trees may be too close to each other in the posterior distribution and top pruning would visit these topologies fairly quickly and approximately in order, while the trees from RAxML may be near different peaks of the distribution.

Our implementation of top pruning does not exhibit linear run time with the number of iterations (Figure S3). This is important to consider, since we compare top pruning against MCMC by run-time. Later iterations of top pruning may add more edges than in previous iterations. This means we expect more likelihood calculations on average in later iterations than in early iterations. With this, we do not expect linear run time is possible, but it is at worst quadratic. There is also the issue of scaling in terms of the number of taxa (the number of leaves in the sDAG, the number of sequences in the alignment, etc.). The performance of top pruning on the flu data set is consistent with that of the DS-datasets (Fig. 10). This suggests top pruning may scale well with the number of taxa.

Lastly, we compare the empirical posterior probability of an edge to the likelihood of the best known tree for the edge (the top pruning likelihood). By posterior probability of an edge, we mean the topology marginalized probability, which is the sum of posterior probabilities of topologies containing the edge. Across data sets, high posterior probability corresponds well to high
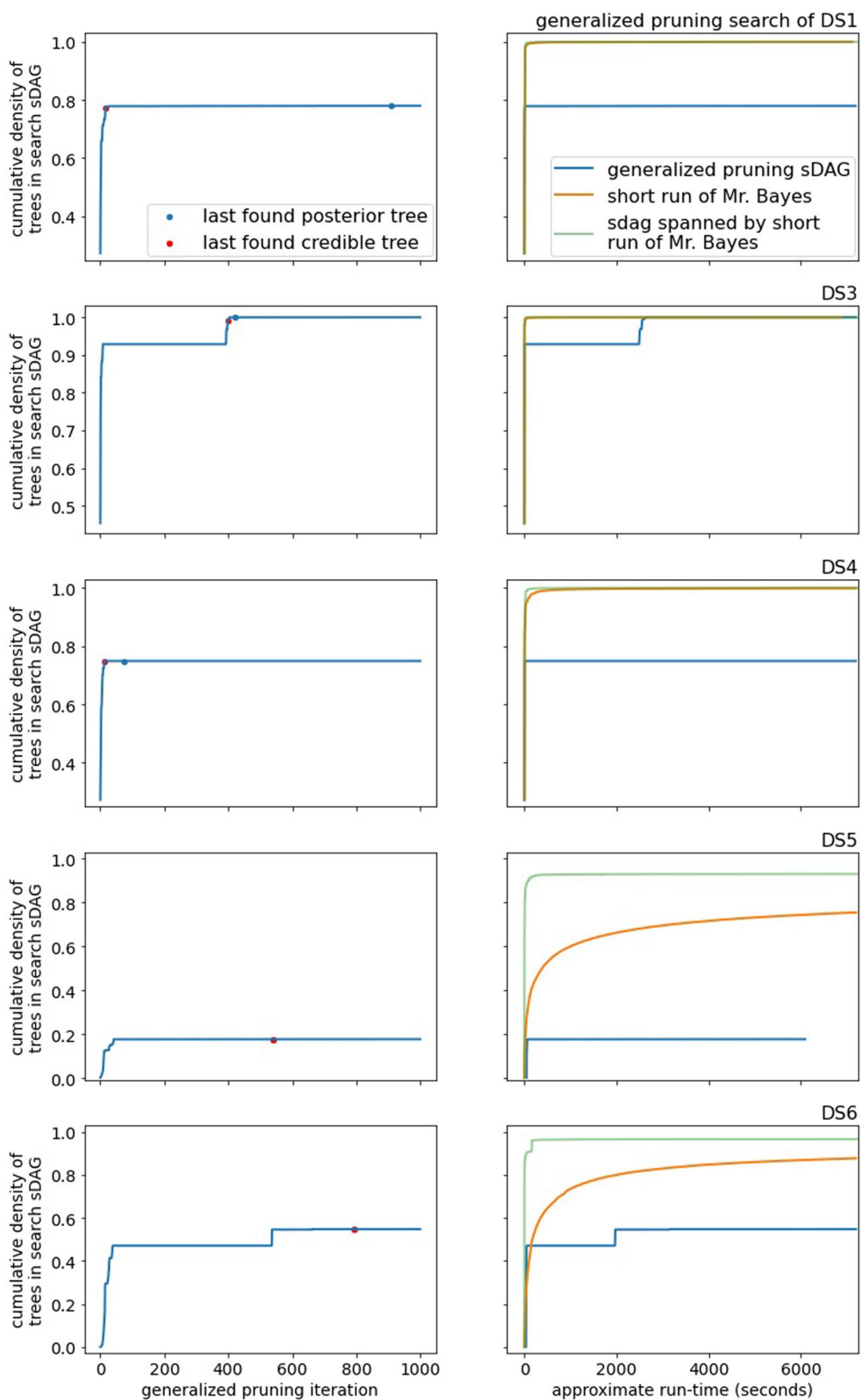
Fig. 6 The empirical posterior density found by generalized pruning on a subset of the DS-datasets and a comparison with MCMC. The red dots in the plots of the left indicate the last iteration finding a topology of the credible set and the blue dots indicate the last iteration finding a topology of the empirical posterior
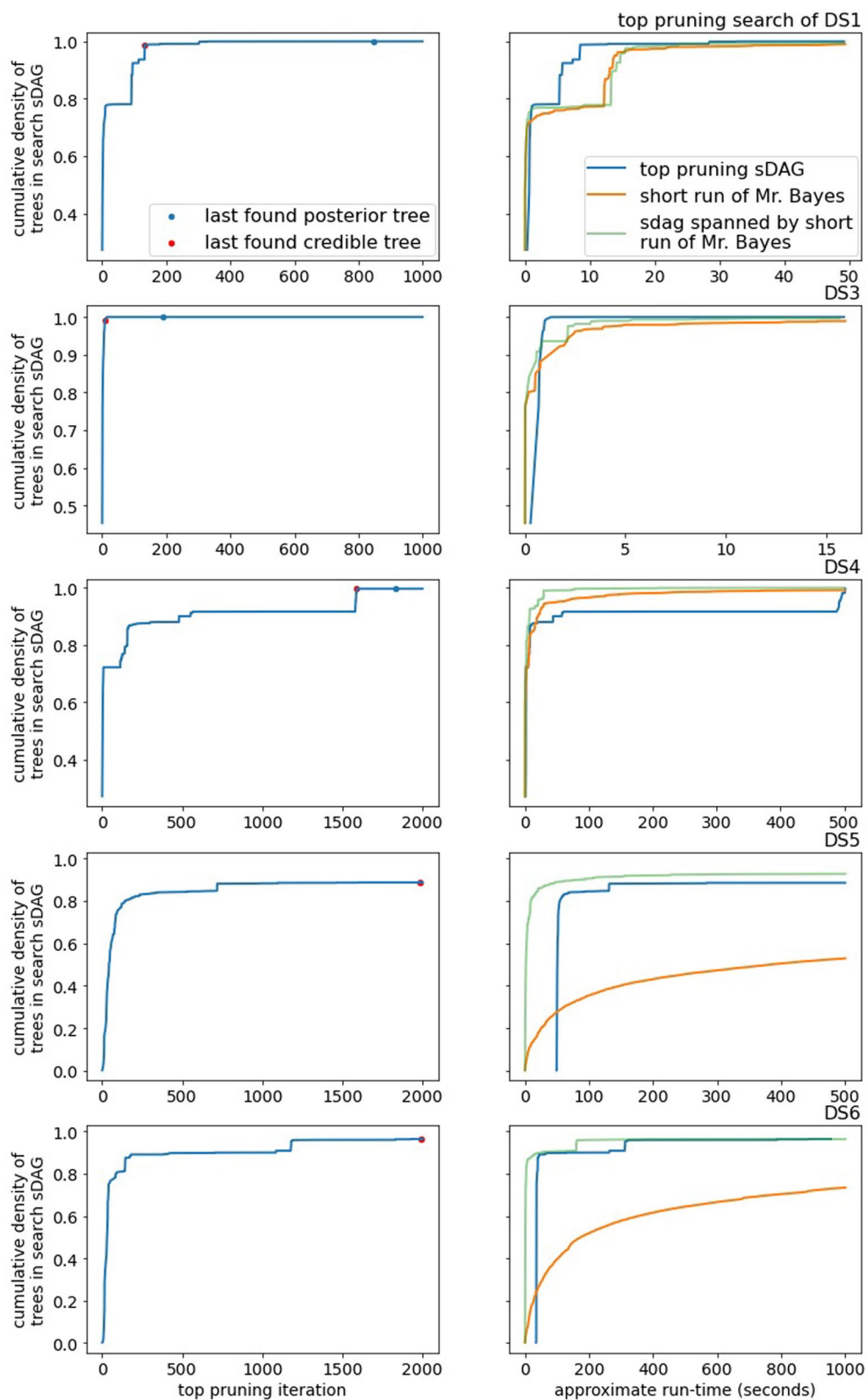
Fig. 7 The empirical posterior density found by top pruning on a subset of the DS-datasets and a comparison with MCMC. Note the *x*-axis scale varies between data sets
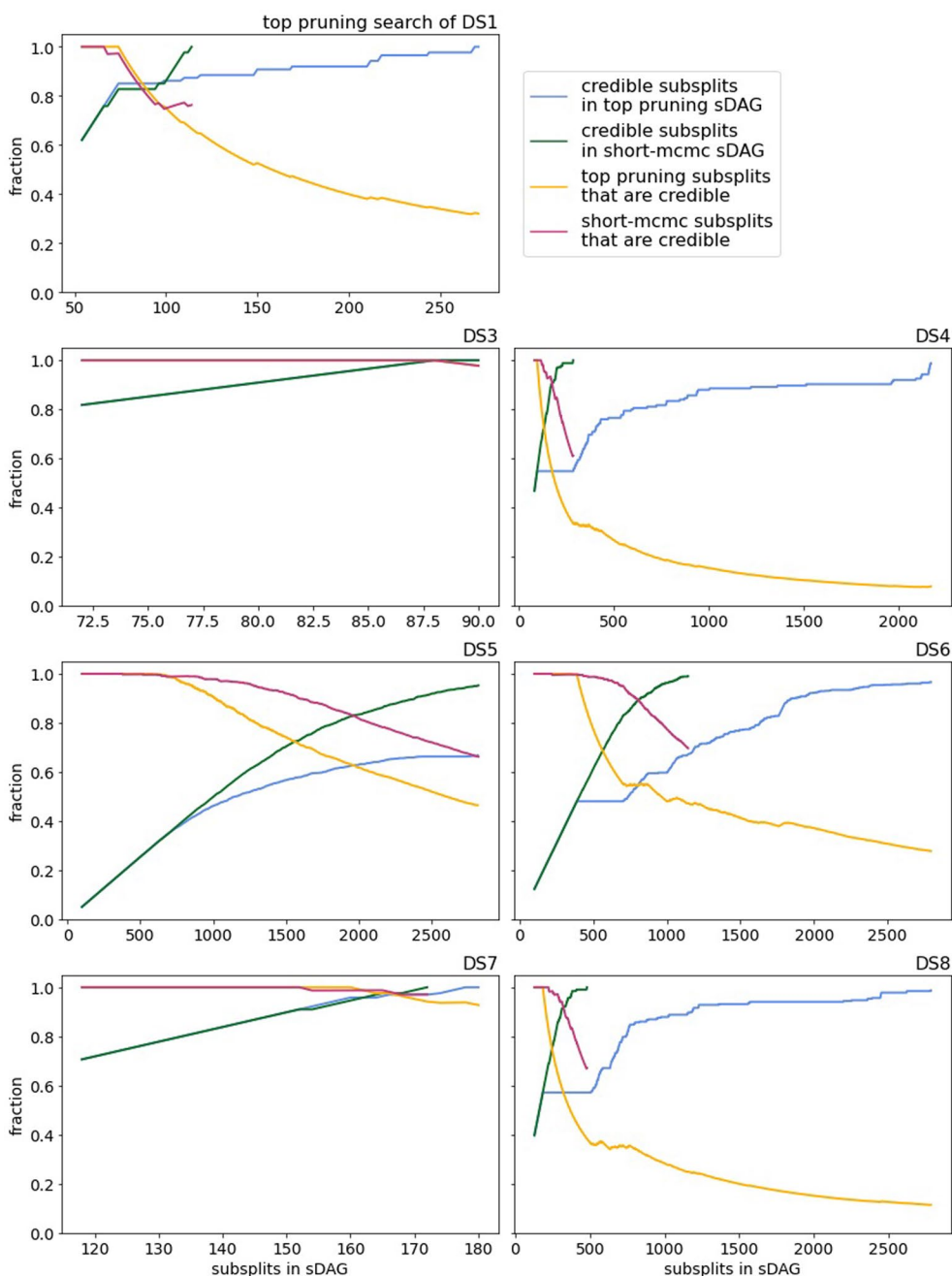
Fig. 8 Quality of sDAGs generated by top-pruning and short-MCMC in terms of subsplits. For each data set, two sDAGs were iteratively enlarged (by top pruning or aggregating trees from MrBayes). On the *x*-axis we have the number of subsplits in the sDAGs and on the *y*-axis we have the fraction of credible subsplits in the sDAGs and the fraction of subsplits in the sDAGs that are credible. In all cases, the initial sDAGs are given by the maximum posterior density topology, and so all subsplits of the initial sDAGs are credible

likelihood of the best known tree of an edge (Fig. 11). This suggests that the challenge with the top pruning algorithm is its tendency to find and include edges outside of the credible set of topologies, but top pruning does a fair job of ranking edges within the credible set.

Discussion

In this paper we develop strategies to identify potentially high posterior density regions of tree space. This fits as part of a larger program to infer Bayesian phylogenetic posterior distributions via optimization [6, 16–18]. We
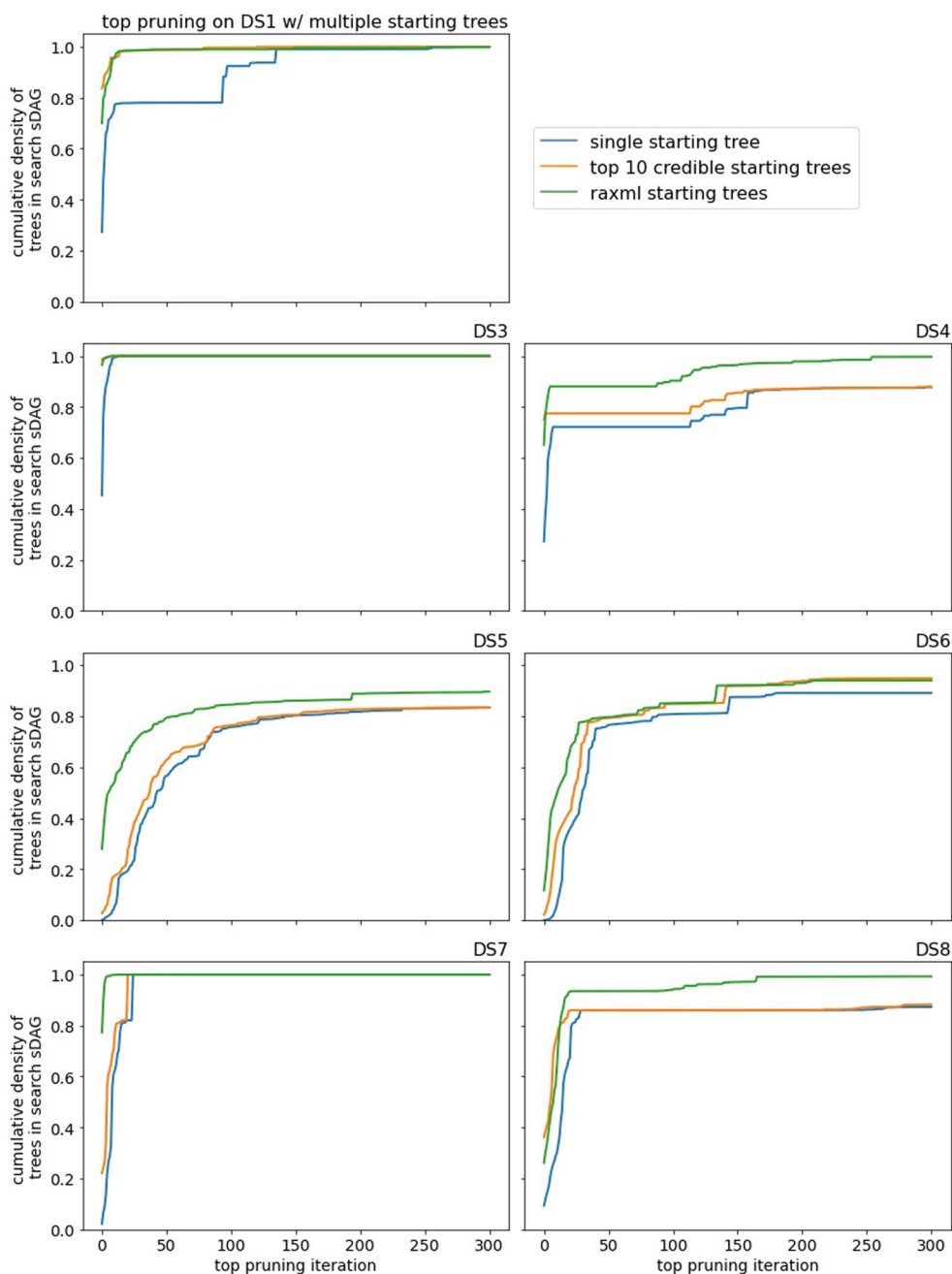
Fig. 9 Performance of top pruning with different starting points

emphasize that our goal for this paper is to find high-posterior regions of topologies, rather than trying to determine posterior densities of these topologies. Once a region is located by a search method, be it a systematic inference or aggregation of trees, another method such as variational inference may be used to infer posterior densities in this region. Such methods may additionally infer distributions for branch lengths and evolutionary

process parameters, allowing for joint posterior density estimates. One could also imagine using sDAG inference to build a means of estimating split frequency as a recent development of adaptive MCMC [9].

While top pruning performs well on our test data sets, the number of sequences is relatively small, with 100 being the largest. We do not yet know how well top pruning scales to larger data sets, where we have
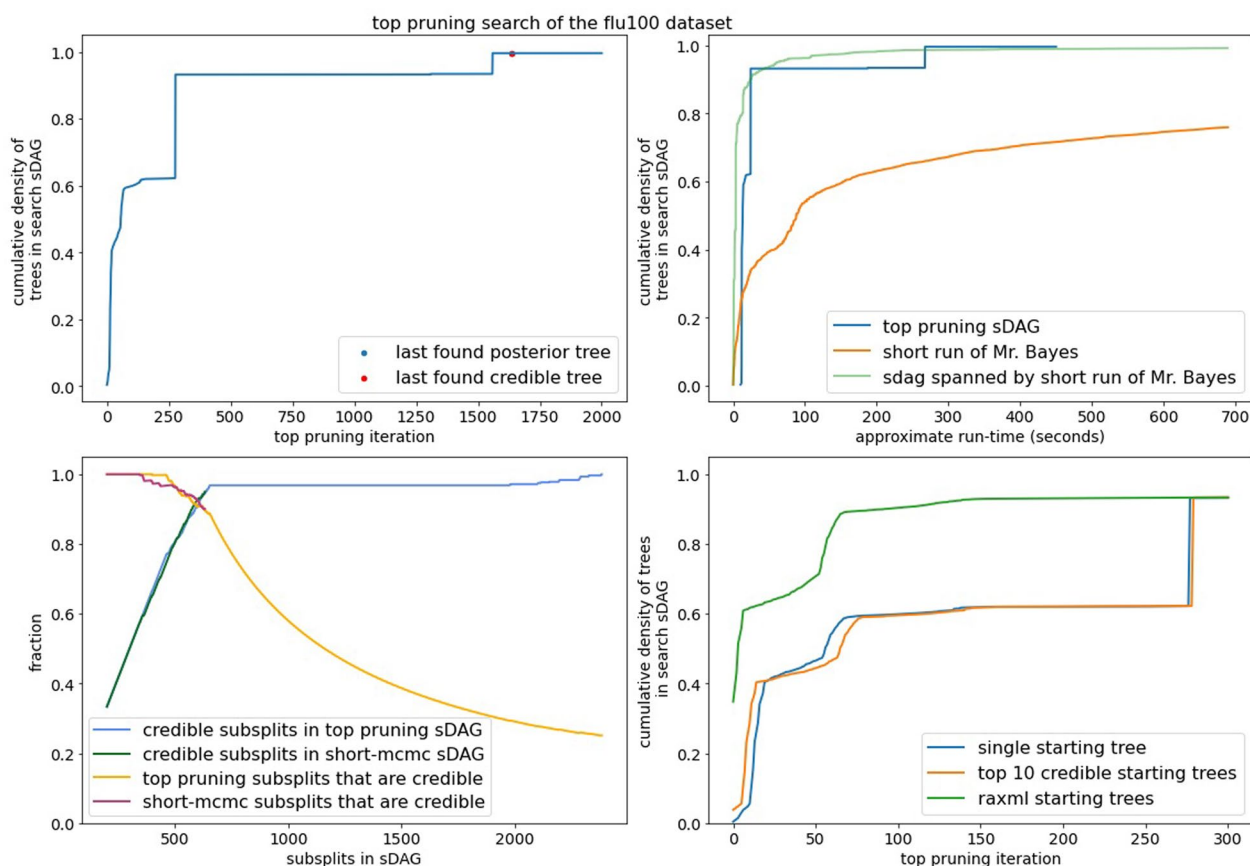
**Fig. 10** Performance of top pruning on the flu100 data set

thousands or tens of thousands of sequences. Our current method of benchmarking is not possible for such data sets, as we require an empirical estimate of the true posterior. However, since top pruning lags behind the performance of aggregating MCMC-sampled trees into an sDAG on these small datasets, that seems like a more promising approach for future work and benchmarking.

Generalized pruning demonstrates poor performance with an NNI search, yet it has been shown useful when determining branch lengths [6]. We have not found any way to improve the performance here. Perhaps the generalized pruning likelihood of an edge does not sufficiently capture the quality of the newly introduced topologies.

One could imagine extensions allowing more general additions to the sDAG in our search algorithms. When proposing and ranking NNIs for top pruning, we could try edges other than those present in the choice maps. For example, rather than using the parent of t specified in the choice map, we could try the other edges to t as candidate parents for t' (such as the dotted blue edge in Figure S5). This requires only a little extra evaluation and branch length optimization.

Furthermore, when we apply the best NNI to the sDAG and remove it from the ranked list, the best known trees for some of the NNIs in the list may now have an edge present in the current sDAG that was not present before. Such an edge has a branch length and choice maps that may yield a different likelihood from before. In such cases, we could trigger re-optimization of branch lengths for that NNI and replace the corresponding likelihoods, PLVs, and choice maps.

An alternative version of top pruning adds exactly five new edges of the best known tree associated to an NNI. That is, in each iteration, there are at most five edges of the best known tree for an NNI that are new to the sDAG. We can add only these edges to the sDAG, rather than edges between all compatible subsplits or the five types of edges detailed in the subsection on Performing NNIs to the subsplit DAG. This algorithm would enjoy linear run time, which the current implementation does not. However, this approach has a fundamental flaw: there are unobtainable subsplits and edges. One can begin with a single topology on a set of taxa, iteratively build an sDAG by adding the five edges from the best known tree for an NNI, continue until all NNIs are exhausted, and

Jennings-Shaffer *et al. Algorithms for Molecular Biology*   (2025) 20:2

Page 17 of 18
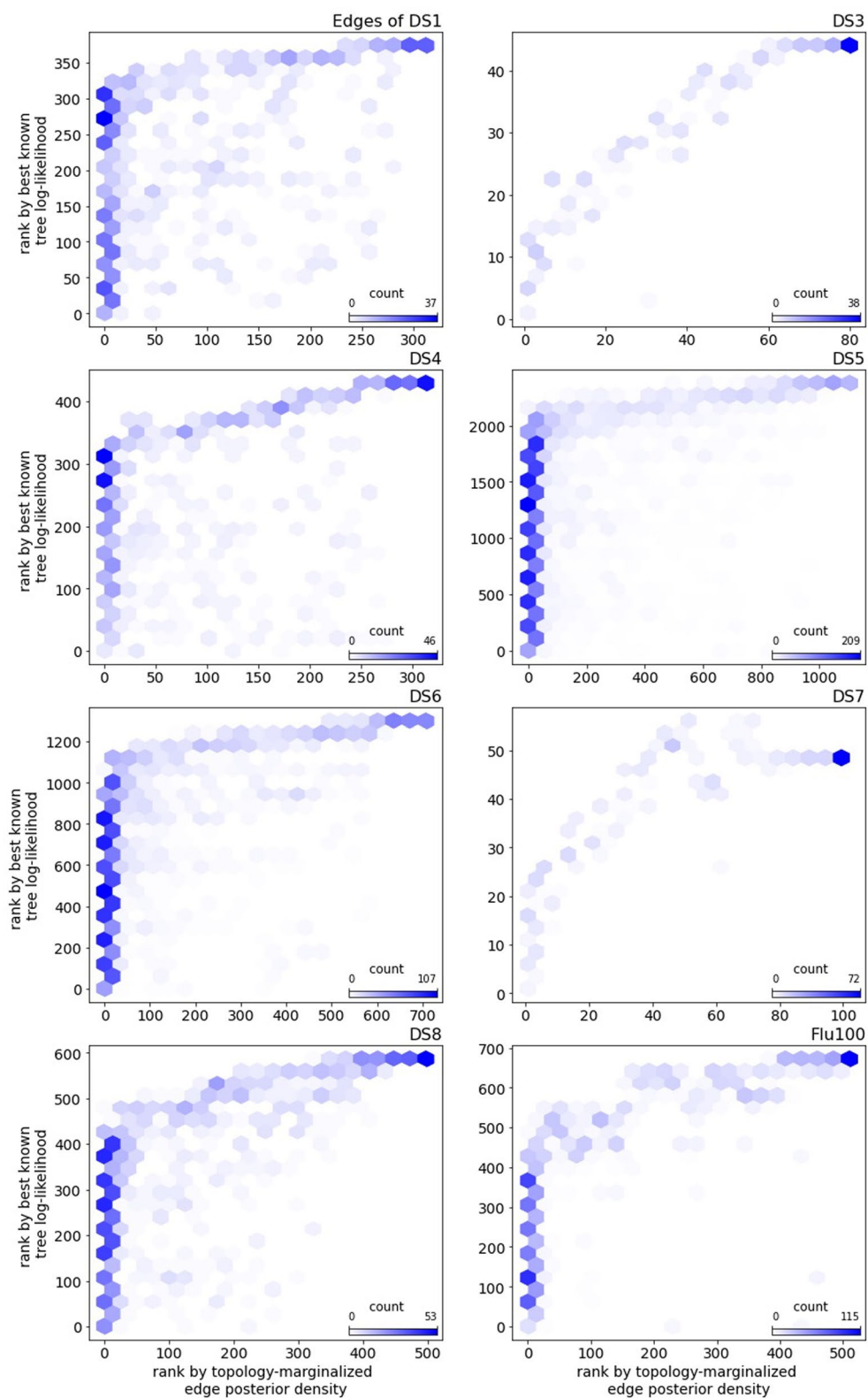
Fig. 11 A comparison of edges ranked by posterior probability and top pruning likelihood. The edges are ordered by their posterior probability along the *x*-axis (with the lowest probability edge at 0) and by their top pruning likelihood along the *y*-axis (with the lowest likelihood edge at 0). Edges are binned for counts, with a darker color meaning a higher count

Jennings-Shaffer *et al. Algorithms for Molecular Biology*   (2025) 20:2

Page 18 of 18

obtain an sDAG missing valid subsplits and edges. This is in strict contrast to NNIs on topologies, where every topology can be reached from any other topology by a sequence of NNIs.

Overall, our goal in this work is to do systematic inference to find high posterior density regions of tree space. We introduce the first structures and operations on those structures to make this possible without considering each tree individually. In order to do systematic inference, one needs a means of evaluating these structures, and here we introduce top pruning and generalized pruning. Although top pruning shows reasonable performance, further work will be needed to improve performance over aggregating trees from an MCMC sampler into an sDAG.

Supplementary Information

The online version contains supplementary material available at https://doi.org/10.1186/s13015-025-00273-x.

> Supplementary file 1.

Author contributions
CJ-S, DHR, and FAM developed the algorithms, implementations, and benchmarks. CJ-S and FAM wrote the main manuscript text. All authors reviewed the manuscript.

Data availability
Data is provided within the manuscript.

Declarations

Competing interests
The authors declare no Competing interests.

References
1. 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.
2. Berling L, Klawitter J, Bouckaert R, Xie D, Gavryushkin A, Drummond AJ. A tractable tree distribution parameterized by clade probabilities and its application to Bayesian phylogenetic point estimation. bioRxiv, 2024.
3. Dumm W, Barker M, Howard-Snyder W, DeWitt WS III, Matsen FA IV. Representing and extending ensembles of parsimonious evolutionary histories with a directed acyclic graph. J Mathe Biol. 2023;87(5):75.
4. 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.
5. Höhna S, Drummond AJ. Guided tree topology proposals for Bayesian phylogenetic inference. Syst Biol. 2012;61(1):1–11.
6. Jun S-H, Nasif H, Jennings-Shaffer C, Rich DH, Kooperberg A, Fourment M, Zhang C, Suchard MA, Matsen FA IV. A topology-marginal composite likelihood via a generalized phylogenetic pruning algorithm. Algorithms Mol Biol. 2023;18(1):10.
7. Lakner C, van der Mark P, Huelsenbeck JP, Largent B, Ronquist F. Efficiency of Markov chain Monte Carlo tree proposals in Bayesian phylogenetics. Syst Biol. 2008;57(1):86–103.
8. Largent B. The estimation of tree posterior probabilities using conditional clade probability distributions. Syst Biol. 2013;62(4):501–11.
9. Meyer X. Adaptive tree proposals for Bayesian phylogenetic inference. Syst Biol. 2021;70(5):1015–32.
10. Redelings B. Bayesian phylogenies unplugged: majority consensus trees with wandering taxa. work in progress, 2011.
11. Ronquist F, Teslenko M, van der Mark P, Ayres DL, Darling A, Höhna S, Largent B, Liu L, Suchard MA, Huelsenbeck JP. MrBayes 3.2: efficient Bayesian phylogenetic inference and model choice across a large model space. Syst Biol. 2012;61(3):539–42.
12. Stamatakis A. RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. Bioinformatics. 2014;30(9):1312–3.
13. 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.
14. Whidden C, Claywell BC, Fisher T, Magee AF, Fourment M, Matsen FA IV. Systematic exploration of the high likelihood set of phylogenetic tree topologies. Syst Biol. 2020;69(2):280–93.
15. Whidden C, Matsen FA IV. Quantifying MCMC exploration of phylogenetic tree space. Syst Biol. 2015;64(3):472–91.
16. Zhang C, Matsen FA IV. Generalizing tree probability estimation via Bayesian networks. Adv Neural Inf Proc Syst. 2018;31:1449–58.
17. Zhang C, Matsen FA IV. Variational Bayesian phylogenetic inference. In International Conference on Learning Representations (ICLR), (2019).
18. Zhang C, Matsen FA IV. A variational approach to Bayesian phylogenetic inference. arXiv preprint, 2022.

Publisher's Note