# Phylogenetic Inference with Generative Flow Networks

Ming Yang Zhou, School of Computer Science

McGill University, Montreal

December, 2023

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of

Master of Computer Science

# Abstract

Phylogenetic inference is the task to infer evolution history and relationship among biological entities. As an essential tool in modern biology, phylogenetic inference finds crucial applications in various domains, including medical research, conservation biology and forensic investigations. However, due to the intricate nature of phylogenetic tree space, phylogenetic inference remains challenging for the current combinatorial and probablistic techniques.

Generative flow networks (GFlowNets) are algorithms for learning generative models of complex distributions given by unnormalized density functions over structured spaces. In this thesis, we present a novel phylogenetic inference algorithm that is based on GFlowNets. To provide context, we first review the three main phylogenetic inference approaches. Subsequently, we delve into the methodology and applications of GFlowNets. Lastly, we introduce PhyloGFN, an innovative phylogenetic inference algorithm built upon GFlowNets.

PhyloGFN is introduced to solve two types of phylogenetic inference problems: parsimony phylogenetic inference and Bayesian phylogenetic inference. We test the performance of PhyloGFN using eight real benchmark datasets. Our results demonstrate that for parsimony analysis, PhyloGFN can successfully retrieve all optimal parsimonious trees. Furthermore, for Bayesian inference, PhyloGFN is competitive with prior works in marginal likelihood estimation and achieves a closer fit to target distribution than state-of-the-art variational inference methods.

# Abrégé

L'inférence phylogénétique est la tâche consistant à déduire l'histoire de l'évolution et les relations entre les entités biologiques. En tant qu'outil essentiel en biologie moderne, l'inférence phylogénétique trouve des applications cruciales dans divers domaines. Cependant, en raison de la nature complexe de l'espace des arbres phylogénétiques, l'inférence phylogénétique reste un défi pour les techniques combinatoires et probabilistes actuelles.

Les réseaux de flot génératifs (GFlowNets) sont des algorithmes permettant d'apprendre des modèles génératifs de distributions complexes définies par des fonctions de densité non normalisées sur des espaces structurés. Dans cette thèse, nous présentons un nouvel algorithme d'inférence phylogénétique basé sur les GFlowNets. Pour situer le contexte, nous passons en revue les trois principales approches en matière d'inférence phylogénétique. Ensuite, nous nous penchons sur la méthodologie et les applications des GFlowNets. Enfin, nous introduisons PhyloGFN, un algorithme innovant d'inférence phylogénétique construit sur la base des GFlowNets.

PhyloGFN est utilisé pour résoudre deux types de problèmes d'inférence phylogénétique : l'inférence phylogénétique de la parcimonie et l'inférence phylogénétique bayésienne. Nous avons testé les performances de PhyloGFN en utilisant huit ensembles de données de référence réels. Nos résultats démontrent que, dans le cadre de l'analyse de la parcimonie, PhyloGFN est capable de récupérer avec succès tous les arbres parcimonieux optimaux. De plus, dans le cas de l'inférence bayésienne, PhyloGFN est compétitif avec les travaux antérieurs en ce qui concerne l'estimation de la vraisemblance marginale et

parvient à s'ajuster plus étroitement à la distribution cible que les méthodes d'inférence variationnelle de pointe.

# Acknowledgements

I first would like to express my sincere appreciation to my supervisor Prof. Mathieu Blanchette for the unwavering support and invaluable guidance I received during my graduate study.

I also want to thank the examiner of my master thesis Prof. Siamak Ravanbakhsh.

I express my heartfelt gratitude to my friends in the Blanchette Lab—Zichao Yan, Elliot Layne, Audrey Baguette, Yanlin Zhang, Dongjoon Lim, Atia Amin, Bowei Xiao, and Cesar Miguel Valdez Cordova. I am also deeply thankful to my collaborators at MILA—Nikolay Malkin, Dinghuai Zhang, Moksh Jain, and Prof. Yoshua Bengio. Their generous sharing of knowledge not only enriched my understanding but also played a pivotal role in making my graduate study both enjoyable and immensely worthwhile

# Table of Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Phylogenetic inference is the process of reconstructing and depicting the evolutionary relationships and branching patterns among different species or groups based on shared genetic, morphological, or behavioral characteristics. The complex nature of phylogenetic tree space poses persistent challenges for current combinatorial and probabilistic techniques. In this thesis, we introduce an innovative phylogenetic inference algorithm that utilizes Generative Flow Networks (GFlowNets), a generative modeling approach proposed in (Bengio et al., 2021), capable of learning complex distributions from unnormalized density functions across structured spaces.

The following chapters are organized as follows: In chapter 2, we first review the three main phylogenetic inference approaches: parimony-based phylogenetic inference, probablistic phylogenetic inference and distance-based phylogenetic inference. In chapter 3, we describe GFlowNets and review their applications. In chapter 4, we introduce PhyloGFN, a GFlowNets based phylogenetic inference algorithm, to solve two types of phylogenetic inference problems: parsimony phylogenetic inference and Bayesian phylogenetic inference.

We apply PhyloGFN on 8 real benchmark datasets. We show that for parsimony analysis, PhyloGFN can retrieve all optimal parsimonious tree. For Bayesian inference, Phy-

loGFN is competitive with prior works in marginal likelihood estimation and achieves a closer fit to target distribution than the state-of-the-art variational inference methods.

## 1.1 Author contribution

This thesis includes the full text and figure of the scientific article *PhyloGFN: Phylogenetic inference with generative flow networks*. The work is accepted in ICLR 2024. I am the first author of the work.

Zhou, M., Yan, Z., Layne, E., Malkin, N., Zhang, D., Jain, M., Blanchette, M., Bengio, Y. (2023) PhyloGFN: Phylogenetic inference with generative flow networks. In ICLR 2024.

*Z.M., Y.Z., L.E., M.N., Z.D., J.M., B.M., B.Y. conceived the study. Z.M. and Y.Z. implemented PhyloGFN and performed data analysis. M.N., Z.D. suggested methods to evaluate PhyloGFN-Bayesian. M. N. derived the formula to estimate the lower bound of log marginal likelihood. Z.M., Y.Z. L.E. Z.D. J.M. and M.N. wrote the manuscript. M.B. and B.Y. supervised the project. M.N., M.B and B.Y. edited the manuscript. All authors read and approved the final article.*

# Chapter 2

# Phylogenetic Inference

Phylogenetics is the study of evolution history and relationship among biological entities. Phylogenetic inference is the task of constructing evolutionary relationships among a set of species, typically represented as phylogenetic trees.

From a historical perspective, in 1804, Charles Darwin's groundbreaking work, *On the Origin of Species*, formalized the theory of evolution by natural selection. In 1866, the German biologist Ernst Haeckel introduced the theory of recapitulation. Although the theory was later disproved, it was Haeckel who coined the term "phylogeny" (originally "Phylogenie" in German) and introduced the concept of the phylogenetic tree: a graphical representation of the evolutionary relationships among species. In 1950, the German entomologist Willi Hennig's *Phylogenetic Systematics* provided a systematic framework for constructing phylogenetic trees. Before the advent of molecular biology, phylogenetic inference relied primarily on the morphological characteristics (form and structure) of species. Starting in the latter half of the 20th century, scientists began to use data from protein, DNA, and RNA sequences to perform phylogenetic inferences.

Today, phylogenetic inference stands as an essential tool in modern biology with significant applications across various domains. For example, in medical science, accurate phylogenetic inference can help understanding the development of antibiotic resistance (Aminov & Mackie, 2007; Ranjbar et al., 2020; Layne et al., 2020) and characterizing tumor

progression (Lähnemann et al., 2020). In forensic investigations, phylogenetic inference is performed to identify HIV transmissions between the suspect and victims (Scaduto et al., 2010). In the realm of conservation biology, phylogenetic inference is applied to assess the risk posed by invasive species (Hamelin et al., 2022).

There are three main types of phylogenetic inferences approaches: 1. probabilistic phylogenetic inference, 2. parsimony phylogenetic inference, and 3. distance-based phylogenetic inference. This chapter provides a detailed explanation of each of these methods.

The inputs for phylogenetic inference comprise heritable traits of the studied species, which may include genetic sequence data such as DNA, RNA, or protein sequences, morphological characteristics like the size or shape of organisms, or genomic markers such as single nucleotide polymorphisms. In this thesis, our focus is primarily on phylogenetic methods that utilize biological sequence data. Additionally, we assume that the sequences have been aligned using a multiple sequence alignment algorithm prior to their use in the phylogenetic inference algorithms.

The results of phylogenetic inference take the form of phylogenetic trees, which are graphical representations of the evolutionary relationships and history among species. The leaves of a phylogenetic trees represents the investigated species. Distances between leaves in a tree indicate the degree of relatedness between the corresponding species. Branch splits at internal nodes correspond to speciation events, where one species diverges into multiple species. A phylogenetic tree can be either weighted or unweighted. An unweighted phylogenetic tree only consists of the tree topology, while a weighted phylogenetic tree includes both the tree topology and branches lengths. Branch lengths denote the extent of evolutionary changes between two species situated at the ends of a branch. Certain phylogenetic inference methods, such as parsimony analysis, focus solely on tree topologies, while others, such as distance-based inference and probabilistic inference, construct weighted phylogenetic trees. Phylogenetic trees can be rooted or unrooted. Rooted trees reveal the directional history of evolutionary relationships, with

internal nodes signifying common ancestors and the root node representing the ancestor of all observed species. Unrooted trees, on the other hand, illustrate relatedness among species at the leaves without specifying an evolutionary direction. Let $n!!$ denote the double factorials of $n$. For $n$ observed species, there are a total of $(2n-3)!!$ possible rooted tree topologies and $(2n-5)!!$ unrooted tree topologies. A rooted tree can be converted to an unrooted tree by removing the root, and vice versa. Phylogenetic trees have two main branching patterns: bifurcating trees and multifurcating trees. Rooted bifurcating trees are rooted binary trees where each parent node has exactly two children nodes, while unrooted bifurcating trees are unrooted binary trees where each internal node has precisely three neighbors. Multifurcating trees do not impose limits on the number of neighbors for internal nodes. In this project, we restrict our discussion on phylogenetic inference methods for bifurcating trees.

To introduce some common notations: let $\boldsymbol{Y} = \{\boldsymbol{y}_1, \boldsymbol{y}_2 \dots \boldsymbol{y}_n\} \in \Sigma^{n\times m}$ denote the set of biological sequences of the $n$ investigated species. Each sequence has $m$ characters from alphabet $\Sigma$, for example, $\{A, C, G, T\}$ for DNA sequences. We denote the $i^{\text{th}}$ site of all sequences by $\boldsymbol{Y}^i = \{\boldsymbol{y}_1[i], \boldsymbol{y}_2[i] \dots \boldsymbol{y}_n[i]\}$. A weighted phylogenetic tree is denoted by $(z, b)$, where $z$ represents the tree topology with its leaves labeled by observed sequences, and $b$ represents the branch lengths. For a tree topology $z$, let $E(z)$ denote the set of edges. For any edge $e \in E(z)$, let $b(e)$ denote its length.

## 2.1 Probabilistic phylogenetic inference

Probabilistic phylogenetic inference constructs phylogenetic trees using a predefined probabilistic evolution model. Given a weighted phylogenetic tree $(z, b)$ with leaves sequences $\boldsymbol{Y}$, the evolution model defines the likelihood, denoted as $P(\boldsymbol{Y}|z, b)$ and the prior, denoted as $P(z, b)$. Utilizing Bayes' rule, the posterior likelihood is formulated as follows:

$$P(z, b|\boldsymbol{Y}) = \frac{P(\boldsymbol{Y}|z, b)P(z, b)}{P(\boldsymbol{Y})} \tag{2.1}$$

where $P(\boldsymbol{Y})$ is the intractable marginal. There are two main types of probabilistic analysis approaches: 1. maximum likelihood analysis (Neyman, 1971) seeks phylogenetic trees with maximum likelihood $P(\boldsymbol{Y}|z,b)$. 2. Bayesian inference (Mau et al., 1999) models the posterior distribution of weighted phylogenetic trees based on the equation 2.1 and generates phylogenetic trees such that the sampling probability is proportional to the unnormalized posterior density $P(\boldsymbol{Y}|z,b)P(z,b)$.

Probabilistic evolution models view evolution as a stochastic process in which sequence mutations are treated as probabilistic events occurring over time. It is a common assumption that each site within a biological sequence evolves independently. Therefore, the occurrence of sequence mutations can be seen as a collection of independent events involving character mutations at different sites of the sequence. Substitution models are mathematical model that describe character mutations. For every pair of characters $(c_1, c_2) \in \Sigma^2$, let $P(c_1|c_2,t)$ denote the probability of mutation $c_2 \to c_1$ after a time period $t$. The probability of substitution for different pairs of characters can vary. A substitution matrix, denoted as $S(t)$, records the substitution rates for all pairs of substitutions after a time period $t$: $S(t)^{ij} = P(c_j|c_i,t)$. The substitution process is a *continuous-time markov chain* (CTMC). A CTMC is a mathematical model that describe a Markovian stochastic process where transitions between different states occur continuously over time. Similar to discrete-time Markov chains, the states at a future timestamp depend only on the current states and events that occur between the current time and the future timestamp. In the context of the substitution process, this means that the transition from the current character $c_1$ to the character $c_2$ at a future timestamp $t$ depends solely on $c_1$ and the events that occur between the current time and $t$. Given two time periods $t_1$ and $t_2$, the substitution probability of $c_1 \to c_2$ for the period $t_1 + t_2$ is formulated as $P(c_2|c_1, t_1 + t_2) = \sum_{b\in\Sigma} P(c_2|b,t_2)P(b|c_1,t_1)$. Such substitution models have *multiplicative* substitution matrices, meaning $S(t)S(s) = S(t+s)$. For RNA/DNA sequences, common CTMC substitution models include the Jukes-Cantor models (Jukes et al., 1969) and the Kimura models (Kimura, 1980). The Jukes-Cantor model assumes equal equilibrium fre-

quencies for all bases and equal substitution rates of all base-pairs. In contrast, the Kiruma models distinguish between substitution rates for transitions ($A \leftrightarrow G$ and $C \leftrightarrow T$) and transversions (other base pairs), acknowledging that transitions are generally more frequent than transversions (Durbin et al., 1998). Probabilistic evolution models also defines the priors distribution of tree topologies and branch lengths. In this thesis, we follow the same prior models used in Zhang & Matsen IV (2018b); Koptagel et al. (2022); Mimori & Hamada (2023): the priors over tree topologies and branch lengths are modelled independently: $P(z, b) = P(z)P(b)$. The tree topologies prior $P(z)$ is modeled as uniform distribution over tree topologies space: $P(z) = \frac{1}{(2n-5)!!}$ for unrooted trees inference and $P(z) = \frac{1}{(2n-3)!!}$ for rooted trees inference. The branch lengths $P(b)$ is modeled jointly, with each branch length being independently modeled by an exponential distribution: $P(b) = \prod_e P(b_e)$, $P(b_e) \sim EXP(\lambda)$ with $\lambda = 10$. We also follow the common assumption that evolution at each site occurs independently (There also exist other evolution models where the variation of evolution rate is correlated among neighboring sites. For example: covarion models introduced in Fitch & Markowitz (1970)). Hence the total likelihood is the product of likelihood at each site: $P(\boldsymbol{Y}|z, b) = \prod_i P(\boldsymbol{Y}^i|z, b)$. Likelihood at each site can be calculated by marginalizing over all internal nodes and their possible character assignment. Given a rooted phylogenetic tree $(z, b)$ with leaves sequences $\boldsymbol{Y}$, the site likelihood $P(\boldsymbol{Y}^i \mid z, b)$ is calculated as following:

$$P(\boldsymbol{y}_1[i] \dots \boldsymbol{y}_n[i] \mid z, b) = \sum_{a^i_{n+1}, \dots a^i_{2n-1}} P(a_{2n-1}) \prod_{j=n+1}^{2n-2} P(a^i_j | a^i_{\alpha(j)}, b(e_j)) \prod_{k=1}^{n} P(\boldsymbol{y}_k[i] | a^i_{\alpha(k)}, b_z(e_k))$$

where $a^i_{n+1}, \dots a^i_{2n-2}$ represent the internal node characters assigned to site $i$ and $\alpha(i)$ represent the parent of node $i$. $P(a_{2n-1})$ is the equilibrium base frequency. $P(a^i_j | a^i_{\alpha(j)}, b(e_j))$ is the substitution probability of $a^i_{\alpha(j)} \to a^i_j$ with evolution time $b(e_j)$. The substitution probability is obtained from the substitution model. Likelihood can be efficiently calculated with the Felsenstein's pruning algorithm (Felsenstein, 1973) in a bottom-up fashion through dynamic programming. Defining $L^i_u$ as the leaf sequence characters at site $i$ be-

low the internal node $u$, and given its two child nodes $v$ and $w$, the conditional probability $P(L_u^i|a_u^i)$ can be obtained from $P(L_v^i|a_v^i)$ and $P(L_w^i|a_w^i)$:

$$P(L_u^i \mid a_u^i) = \sum_{a_v^i, a_w^i \in \Sigma} P(a_v^i \mid a_u^i, b(e_v)) P(L_v^i \mid a_v^i) P(a_w^i \mid a_u^i, b(e_w)) P(L_w^i \mid a_w^i). \tag{2.2}$$

This equation can be used to recursively compute $P(L_u^i|a_u^i)$ at all nodes of the tree and sites $i$. For each node $u$ at site $i$, a real vector $\boldsymbol{f}_u^i \in [0,1]^{|\Sigma|}$ is used to store conditional probability $\boldsymbol{f}_u^i[c] = P(L_u^i|a_u^i = c)$. For leaves nodes, $f_u^i$ is the one hot vector of the $i^{th}$ sequence character. The algorithm performs a post-order traversal of the tree to compute conditional probability at each nodes. The likelihood $P(\boldsymbol{Y}^i|z,b)$ is calculated using the root-level conditional probability: $P(\boldsymbol{Y}^i|z,b) = \sum_{a_{2n-1}^i \in \Sigma} P(a_{2n-1}^i) P(\boldsymbol{Y}^i|a_{2n-1}^i)$, where $P(a_{2n-1}^i)$ is the equilibrium base frequency.

If the substitution model is multiplicative and time-reversible (the substitution matrices are symmetric), likelihood of an unrooted tree is the same to any of its rooted version. Hence, its likelihood can be computed by firstly converting the unrooted tree to a rooted tree, then applying the Felsenstein's algorithm to the rooted tree.

The maximum likelihood phylogenetic inference is an NP-Hard optimization task (Roch, 2006). Common approaches for maximum likelihood inference typically involve two iterative steps: 1. given a fixed tree topology, optimize branch lengths to maximize likelihood. 2. Fixing branch lengths, identify the tree topology with maximum likelihood (Dhar & Minin, 2015). Expectation maximization algorithms and gradient-based methods such as Newton-Raphson are used to optimize branch lengths for a fixed tree topology. To optimize tree topology while keeping branch lengths fixed, greedy search heuristics explore the tree shape space by performing local modifications to previously visited tree topologies. One widely used software for maximum likelihood inference is IQ-Tree (Minh et al., 2020). It implements the UFBoot algorithm (Hoang et al., 2018) which has been employed in various recent studies to generate maximum likelihood trees (Koptagel et al., 2022; Zhang, 2023).

Bayesian phylogenetic inference is a more challenging problem as it requires to model the posterior distribution over weighted tree space. Markov chain Monte Carlo (MCMC)-based algorithms are commonly employed for Bayesian phylogenetics. Notable MCMC-based software include MrBayes and RevBayes (Ronquist et al., 2012; Höhna et al., 2016). Estimated lower bound of marginal log likelihood (MLL) is a a common metric to assess the performance of bayesian phylogenetic algorithms. With Stepping-Stone method, a variant of MCMC sampling, Mr.Bayes achieves the state-of-the-art performance on MLL estimation on real datasets (Zhang, 2023). A known limitation of MCMC is its lack of scalability to high-dimensional distributions with multiple separated modes (Tjelmeland & Hegstad, 2001), which arises in larger phylogenetic datasets. Amortized variational inference (VI) (Wainwright & Jordan, 2008) is an alternative approach that parametrically estimates the posterior distribution. VBPI is the first variational inference algorithm for Bayesian phylogenetics (Zhang & Matsen IV, 2018b). A key challenge to model explicitly the distribution of trees is to efficiently explore the tree topology space due to the combinatorially vast numbers of discrete tree shapes. VBPI employs subsplit Bayesian networks (SBN) (Zhang & Matsen IV, 2018a) to model tree topologies distribution. SBN construct tree topology with an intuitive top-down approach: given a root node and a set of sequences $Y$, split $Y$ into two sets and assign them to the two children nodes, and continue the splitting process until each leave node is labeled by one sequence of $Y$. One interesting property of this method is that every tree topology is constructed by exactly one unique sequence of construction steps. The disadvantage of the method is that there are $\mathcal{O}(2^N)$ ways to split $N$ sequences. A successful VI algorithm needs to model correctly the sampling probability distribution of all $\mathcal{O}(2^N)$ splitting patterns at every step such that the joint sampling probability of a tree is proportional to the un-normalized posterior, which is a difficult task for large $N$. To simplify the problem, SBN first run a separate algorithm, for example: an MCMC-based method, to sample high quality tree topologies. It then limits the set of split patterns at each step of SBN by discarding all split patterns that do not appear in the pre-generated tree set. By restricting split patterns

in SBN, majority of phylogenetic trees cannot be modeled with VBPI. Following from this work, Zhang (2023) proposes VBPI-GNN to improve branch length modeling process. Instead of using hand engineered features, VBPI-GNN models tree structure using Graph Neural Network. The algorithm achieves state-of-the-rat performance in term of MLL estimation. However, without changing the method to model tree topologies, the method is still only able to model a small portion of tree space. There exist also VI algorithms that model the entire tree space. VaiPhy (Koptagel et al., 2022) approximates the posterior distribution in the augmented space of tree topologies, edge lengths, and ancestral sequences. Combined with combinatorial sequential Monte Carlo, the proposed method enables faster estimation of marginal likelihood. GeoPhy (Mimori & Hamada, 2023) models the tree topology distribution in continuous space by mapping continuous-valued coordinates to tree topologies, using the same technique as VBPI-GNN to model tree topological embeddings. While both methods model the entire tree topology space, their performance on marginal likelihood estimation underperforms the state of the art.

Probabilistic methods offer a significant advantage in their robustness and flexibility, allowing researchers to apply complex evolutionary models that more accurately represent the true evolutionary processes. Bayesian inference, in particular, has the capability to learn parameter distributions, thereby naturally quantifying the uncertainties associated with the estimates. The drawback of these methods lies in their high computational complexity. For large-scale problems, they often require considerably more time to execute, and the optimization process can be challenging. Comparing with Bayesian inference, maximum likelihood inference methods are notably faster. However, since maximum likelihood inference does not produce distribution over inferred trees, it requires to use other methods such as bootstrapping to infer the variability.

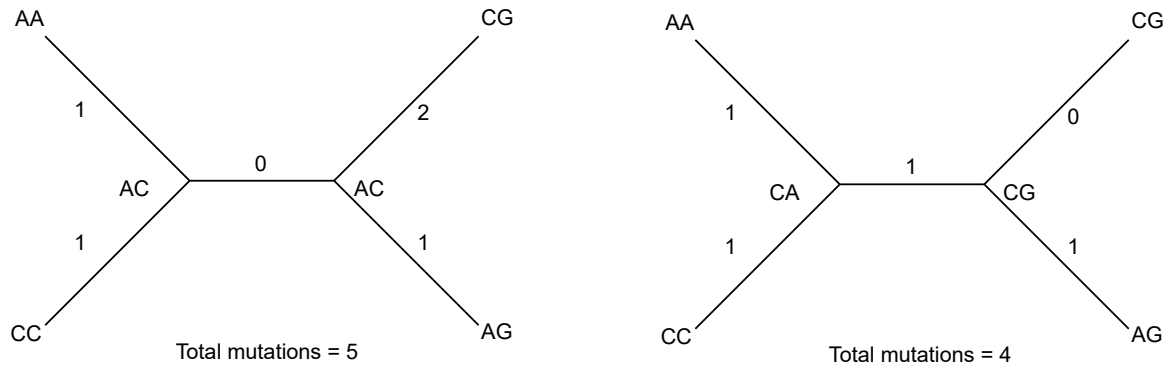



**Figure 2.1:** Example of two sets of sequence assignments to internal nodes. The assignment on the right is a parsimonious assignment resulting minimum total number of mutations

## 2.2 Parsimony Analysis

Parsimony analysis explains the observed sequences by phylogenetic trees that involve the minimum net amount of evolution. Given a tree topology, for any sequences assignment on internal nodes, we can count the total number of substitutions by summing up all character changes to pass from one sequence to another over all branches of the tree. The parsimony score for a tree topology is the minimum possible character changes when assigning ancestral sequences optimally. Figure 2.1 shows an example of 2 different internal sequences assignments on a phylogenetic trees with 4 leaves. The assignments on the right plot is an optimal assignment that produce the minimum total of 4 character changes in the tree. Hence the parsimony score of this phylogenetic tree is 4. Weighted parsimony is an extension of parsimony such that instead of counting number of substitution, for every substitution of $c_1$ to $c_2$, we add a cost $S(c_1, c_2)$ based on a substitution weight matrix $S$. The weighted parsimony score is thus the minimum substitution score with optimal internal sequences assignment. The original parsimony problem can be seen as a special case of the weighted parsimony problem where $S(c_1, c_2) = 1$ for any $c_1 \neq c_2$.

In parsimony analysis, we also assume that each site of the sequences evolves independently. Therefore, optimal sequences assignment and parsimony score is computed

independently for each site. Let $M(z|Y_i)$ denote the parsimony score for site $i$ and $M(z|Y)$ denote the total parsimony score of the tree topology $z$: $M(z|Y) = \sum_i M(z|Y_i)$. For both weighted and original parsimony problem, the substitution weight matrix is symmetric, hence the parsimony score for unrooted tree equals that of any of the equivalent rooted tree's score. Fixing the tree topology, finding the optimal internal sequences assignment and parsimony score is referred as the Small Parsimony Problem. The problem of finding the optimal tree topology to obtain minimum parsimony score across the entire tree space is referred to as the Large Parsimony problem (Felsenstein, 2003).

The Small Parsimony Problem can be solved efficiently with dynamic programming. Sankoff algorithm (Sankoff, 1975) and Fitch algorithm (Fitch, 1971) are notable methods to solve the small parsimony problem for the weighted version and unweighted version (see sections 2.2.1 and 2.2.2 for computation details) . There is no efficient and exact methods for the Large Parsimony problem as this discrete optimization problem is NP-hard (Durbin et al., 1998) . When number of sequences is small, exact methods such as Branch and Bound (Little et al., 1963) can be used to sweep the entire data space and retrieve all parsimonious trees. For large-scale problems, heuristic algorithms are commonly used to perform greedy search with different branch swapping mechanisms including nearest neighbor exchange, subtree pruning regrafting and bisection reconnection. PAUP* (Swofford, 1998) is one of the commonly used software for parsimony analysis with heuristics algorithms. The main advantage of parsimony-based phylogenetic inference is its simplicity. The parsimony principle is intuitive and the resulting phylogenetic trees are easy to interpret. Parsimony methods are generally faster than probabilistic based methods. The disadvantage of parsimony based method is that they leverage a simple model that only considers the events of character changes themselves while ignoring other important factors in the evolution process such as evolution time. Also, parsimony methods do not consider the probabilistic nature of evolution. The solution for large parsimony problem is deterministic. Similar to the maximum likelihood inference, in order to as-

sess the variability of the results, one has to use other methods such as non-parametric bootstrapping.

### 2.2.1 Fitch's Algorithm

Given a rooted tree topology $z$, the Fitch algorithm assigns optimal sequences to internal nodes and computes the parsimony score in $\mathcal{O}(MNC)$ where $M, N, C$ represent site number per sequence, sequences number and characters number. At each node $u$, the algorithm tracks the set of possible characters labeling for node $u$ that can yield a most parsimonious solution for the subtree rooted at $u$. This character set can be represented by a binary vector $\boldsymbol{f}_u^i \in \{0,1\}^{|\Sigma|}$ for site $i$. As in Felsenstein's algorithm, this vector is a one-hot encoding of the sequences at the leaves and is computed recursively for non-leaves. Specifically, given a rooted tree with root $u$ and two child trees with roots $v$ and $w$, the Fitch character set $\boldsymbol{f}_u^i$ is calculated as:

$$\boldsymbol{f}_u^i = \begin{cases} \boldsymbol{f}_v^i \wedge \boldsymbol{f}_w^i & \text{if } \boldsymbol{f}_v^i \cdot \boldsymbol{f}_w^i \neq 0 \\ \boldsymbol{f}_v^i \vee \boldsymbol{f}_w^i & \text{otherwise} \end{cases},$$

where $\wedge$ and $\vee$ are element-wise conjunctions and disjunctions. The algorithm first traverses the tree in post-order (bottom-up) to calculate the characters set at each node. It then traverses in pre-order (top-down) to assign optimal sequences. For the root node $u$ of the tree at the site $i$, the character assignment $\boldsymbol{y}_u^i$ can be any character in $\boldsymbol{f}_u^i$. To calculate the children internal node character assignments, given a node $u$'s assignment $\boldsymbol{y}_u^i$ and its child node $v$'s characters set $\boldsymbol{f}_v^i$:

$$\boldsymbol{y}_v^i = \begin{cases} \boldsymbol{y}_u^i & \text{if } \boldsymbol{y}_u^i \in \boldsymbol{f}_v^i \\ \text{any character of } \boldsymbol{f}_v^i & \text{otherwise} \end{cases},$$

The total number of character changes between these optimal sequences along the tree's edges is counted as the parsimony score.

### 2.2.2 Sankoff's Algorithm

Sankoff's algorithm computes the optimal sequence assignment and weighted parsimony score for rooted tree topologies. The algorithm works in similar fashion as Fitch algorithm. Since the substitution weight is no longer constant, at every node, we need to track minimum parsimony score for every character assignment. Let $\boldsymbol{f}_u^i \in N^{|\Sigma|}$ denote the partial parsimony scores at node $u$ for site $i$. Let $S(i,j)$ denote the parsimony weight for the substitution pair $(i,j)$. Given a rooted tree with root $u$ and two child trees with roots $v$ and $w$, the partial scores $\boldsymbol{f}_u^i$ can be calculated using the partial scores of its children $\boldsymbol{f}_v^i$ and $\boldsymbol{f}_w^i$:

$$\boldsymbol{f}_u^i[j] = min_{k\in\Sigma}\{\boldsymbol{f}_v^i[k] + S(j,k)\} + min_{k\in\Sigma}\{\boldsymbol{f}_w^i[k] + S(j,k)\}$$

For a leaf node $u$, $f_u$ is initialized as following:

$$f_u^i[k] = \begin{cases} 0 & \text{if } k \text{ is the assigned character at site } i \\ \infty & \text{otherwise} \end{cases}$$

The algorithm traverses the tree two times, first in post-order (bottom-up) to compute partial parsimony scores at each node, then in pre-order (top-down) to assign optimal sequences. The minimum score stored at the root node is the weighted parsimony score.

## 2.3 Distance-Based phylogenetic inference

Distance-based phylogenetic inferences are methods to construct phylogenetic trees based on a pre-defined distance matrix $D$, where $D_{i,j}$ represents the expected distance between

to two species $(i, j)$ in a phylogenetic tree. For any particular weighted tree, the distance $d_{i,j}$ between the species $i$ and $j$ in the tree can be computed by adding up branch lengths on the path between two species. The goal of distance-based analysis to construct phylogenetic trees such that species distances $d_{i,j}$ in the tree align with the expected distance $D_{i,j}$.

The distance matrix can be defined with various distance measures. For sequences data, the simplest distance measure is the edit distance between two sequences. The edit distance is the minimum number of operations (insertions, deletions and substitutions) required to transform one sequence into the other. The distance measure can also be obtained from the substitution model. For example, using the Jukes-Cantor model, let $p_{ij}$ be the portion of the sites with different nucleotides for sequences $i$ and $j$, the Jukes-Cantor distance $D_{ij}$ is defined as $D_{ij} = -\frac{3}{4}ln(1 - \frac{4}{3}p_{ij})$.

Given a phylogenetic tree $(z, b)$, we define the squared distance $Q$ in the following form:

$$Q = \sum_{i,j} w_{ij}(D_{ij} - d_{ij})^2$$

where $w_{ij}$ are predefined weights. Methods to find phylogenetic trees with minimum squared distance $Q$ are called least squared methods. Given a tree topology $z$, one can compute optimal branch lengths by taking gradient with respect to each pair of species then solving the system of linear equations (Felsenstein, 2003). Constructing the optimal phylogenetic tree $(z, b)$ is more challenging as finding the optimal tree topologies faces the same type of difficulties as described in parsimony analysis. There are also approximation methods that do not guarantee to find the minimal square distance phylogenetic trees, but run much faster. One notable algorithm is called UPGMA (Sokal et al., 1958) which restrict the search space to rooted trees that are *ultrametric*: total branch length from the root to any leave equal. The Neighbor-Joining algorithm is another well-known approximation method (Saitou & Nei, 1987). The algorithm generates unrooted tree. And it

is guaranteed to find the optimal tree if the provided distance matrix exactly records the leaves' distances of the optimal tree (Felsenstein, 2003).

The primary advantage of distance-based inference is its simplicity. Much like parsimony-based inference, this method is intuitive, and the results are straightforward to interpret. However, the effectiveness of distance-based inference is heavily reliant on selecting an appropriate distance measure. It's important to note that the choice of a distance measure can lead to different computed phylogenetic trees, and there is no established standard procedure for selecting the optimal distance measure. Additionally, distance-based phylogenetic inference operates based on a distance matrix as input, which restricts researchers from incorporating complex evolutionary models that might better represent the true evolutionary processes.

## 2.4 Estimating uncertainties with Bootstrapping methods

Non-parametric bootstrapping methods are commonly employed to estimate uncertainties for parsimony-based phylogenetic inferences, distance-based phylogenetic inferences and maximum likelihood phylogenetic inferences (Felsenstein, 2003). Given the sequences set of $m$ sites $\boldsymbol{Y} = \{\boldsymbol{Y}^1, \boldsymbol{Y}^2 \dots \boldsymbol{Y}^m\}$, a bootstrap sequences set is obtained by sampling with replacement $m$ times from sites entries $\{\boldsymbol{Y}^1, \boldsymbol{Y}^2 \dots \boldsymbol{Y}^m\}$. For each re-sampled sequences set, we apply the phylogenetic inference method to obtain a phylogenetic tree. By repeating the procedure of re-sampling sequences set and generating trees, we obtain a collection of phylogenetic trees. The variability of the phylogenetic inference method is quantified by the variability among this collection of trees. One common method to capture summary statistics for a large set of trees is to construct a consensus tree. A consensus tree is a tree structure that shows relationship of taxa that commonly appear in a set of phylogenetic trees. Figure 2.2 shows an example of consensus tree for 100 bootstrapped phylogenetic trees. The sequences set is sampled from the first 10 sequences of the data set collected in Hedges et al. (1990). The method in evaluation is parsimony

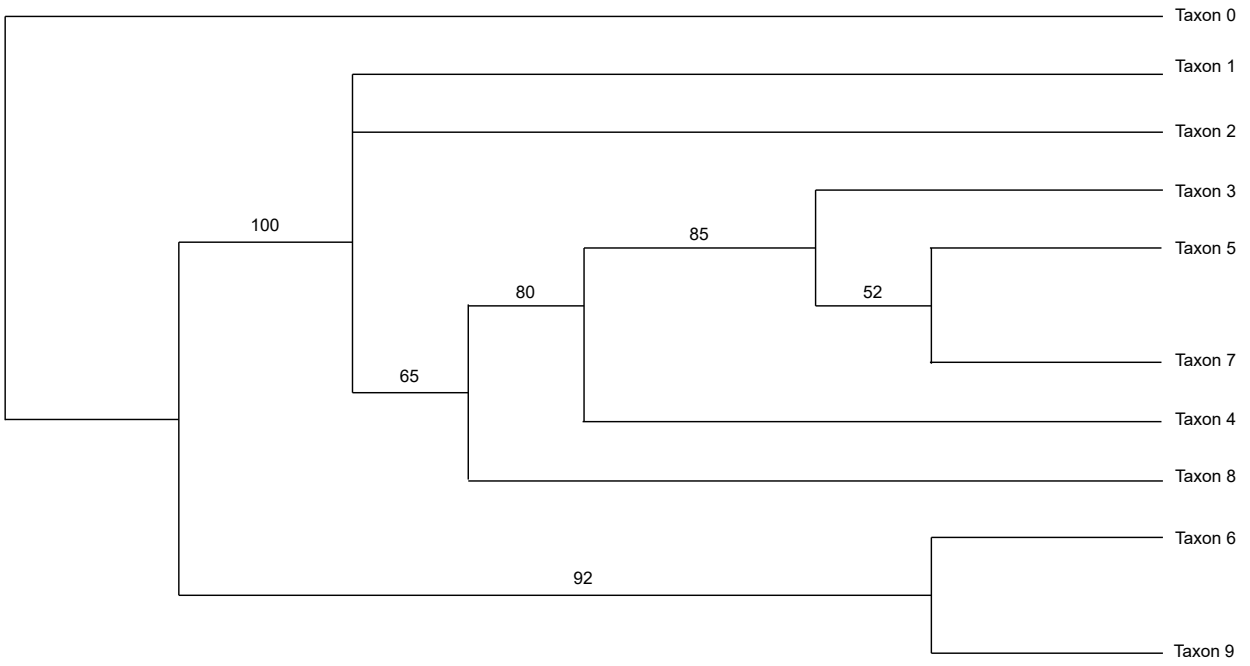



**Figure 2.2:** Concensus tree of 100 phylogenetic trees calculated by PAUP* on first 10 sequences of dataset DS1 with parsimony method.

analysis and phylogenetic trees are generated by the software PAUP*. The consensus tree shows grouping relations of taxa that appear in at least $50\%$ of the 100 trees. For example, for 85 trees out of 100, taxa 3, 5, 7 share a common ancestor. For all 100 bootstrapped trees, taxa 1, 2, 3, 5, 7, 4, 8 share a common ancestor.

In conclusion, this chapter reviews phylogenetic inference and the three main types of computation approaches: distance-based phylogenetic inference, parsimony-based phylogenetic inference and probabilistic phylogenetic inference. Efficient and accurate phylogenetic inference remains an open problem with new approaches proposed every year.

# Chapter 3

# Generative Flow Networks

Generative flow networks (GFlowNets) are algorithms proposed by Bengio et al. (2021) that learn generative models of complex distributions given by unnormalized density functions over structured spaces. As generative models, GFlowNets have shown great success in scientific discovery tasks such as biological sequences design (Jain et al., 2022) and molecule design (Bengio et al., 2021). They also show good performance on producing diverse optimal solutions for NP-hard optimization problems such as scheduling (Zhang et al., 2023a) and graph combinatorics optimization problems (Zhang et al., 2023b). The flexibility afforded by GFlowNets to learn sequential samplers for distributions over compositional objects makes them a promising candidate for performing inference over the posterior space of phylogenetic tree topologies and evolutionary distances. This chapter introduces discrete GFlowNet, the method we leverage to perform phylogenetic inference in PhyloGFN.

Given a data space $\mathcal{X}$ with a reward function $R : x \in \mathcal{X} \to \mathbb{R}^+$ and a sequential procedure to construct objects in $\mathcal{X}$, GFlowNet learns a generative model that samples from $\mathcal{X}$ through a sequence of actions. The sequential procedure to construct objects in $\mathcal{X}$ is used to design a Markov Decision Process (MDP) to sample data. The reward function $R(x)$ is predefined by the user, usually in form of $\exp(-\epsilon(x))$ for some energy function $\epsilon(x)$. Rather than finding the object $x^*$ with optimal $R(x^*)$ score, GFlowNet learns a sam-

pler such that $p(x) \propto R(x)$. In the following sections, we first introduce Markov Decision Process, then we introduce the GFlowNets setup and learning. Finally we discuss some of its applications.

## 3.1 Markov Decision Process

Markov Decision Process (MDP) is a discrete-time mathematical framework that is used to model sequential decision making. An MDP can be defined as a tuple of 4 components $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$:

- $\mathcal{S}$ denotes the state space, the set of states capturing all possible situations in which an agent can find itself. Each state contains relevant information on the environment for an agent to make decision.
- $\mathcal{A}$ denotes the action space, the set of actions an agent can take to transition from one state to another. Note that the action pool for an agent can be different for different states. Denote $\mathcal{A}_s \subseteq \mathcal{A}$ the set of available action for state $s$.
- $\mathcal{P}(s'|s, a) : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ denotes the transition probability from state $s$ to $s'$ when applying action $a$
- $\mathcal{R}(s'|s, a) : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ denotes the expected reward by applying action $a$ on state $s$ to transit to state $s'$

Let $S_t$, $A_t$, $R_t$ defines the random variable of state, action and reward at timestamp $t$. We define the *dynamic* of MDP to describe the transition probability of both state and reward:

$$p(s', r|s, a). = Pr[S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a],$$

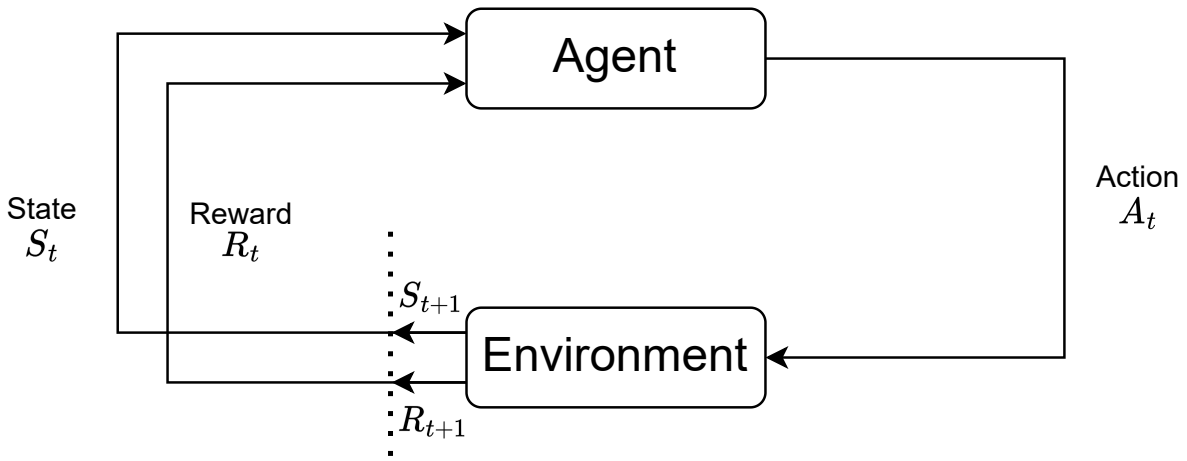



**Figure 3.1:** Agent and environment interaction in MDP

Expected reward and transition probability of states can be formulated using $p(s', r|s, a)$

$$\mathcal{P}(s'|s, a) = \sum_r P(s', r|s, a)$$
$$\mathcal{R}(s'|s, a) = \sum_r r \frac{P(s', r|s, a)}{\mathcal{P}(s'|s, a)}$$

The process is **markovian** in the sense that when applying an action $a$ over a state $s$, the next transition state $s'$ and collected reward $r$ are only determined by current state and action and disregard all previous operations. The diagram 3.1 shows how an agent (the decision maker) interacts with the environment in an MDP. Given the agent at state $S_t$, it performs an action $A_t$ to the environment. Based on the MDP dynamic defined above, the environment transits the agent to state $S_{t+1}$ and give the agent reward $R_{t+1}$. And the agent performs a new action $A_{t+1}$ and the process continuous until some pre-defined termination condition is met

A policy $\pi : \mathcal{S} \to (a \in \mathcal{A}_s \to [0, 1])$ determine how the agent navigates in the environment. A policy can be deterministic, i.e. for each state the policy chooses one action to operate, or stochastic, i.e. the policy outputs a probability distribution over the actions set. MDP is commonly used to model the environment for Reinforcement Learning (RL)

algorithms. For GFlowNets, given a sequential construction process for data space $\mathcal{X}$, a MDP is designed to model the sampling process. A policy on this MDP determines a sampling distribution for $\mathcal{X}$. The goal of GFlowNets is to learn an optimal policy such that sampling probability is proportional to the object reward.

## 3.2 GFlowNet

We first describe the sampling MDP. Figure 3.2 shows an example of the MDP states graph. The states graph is a Directed Acylcic Graph (DAG) with an intial state $s_0$ as the source and a terminal state $s_f$ as the sink. The initial state represents the initialization of sampling. The terminal state represents the termination of sampling. Any states lying between the source and the sink represent partial constructs generated based on the construction procedure. States that are one step away from the sink represent data objects in $\mathcal{X}$. A trajectory $\tau : s_0 \to \cdots \to x \to s_f$ represent a sampling path to generate object $x$. The GFlowNet MDP is deterministic: applying an action to a state result in transition to an unique subsequent state. Therefore, a policy can be defined directly by the forward transition probabilities between adjacent states: $P_F(s_{i+1}|s_i)$. The probability of sampling a trajectory $P_F(\tau)$ is then the product of forward transition probability between pair of states along the path:

$$P_F(\tau) = \prod_{(s_i \to s_{i+1}) \in \tau} P_F(s_{i+1}|s_i)$$

Mulitple trajectories can lead to the same object $x$, hence sampling probability $P(x)$ is the sum of sampling probabilities of all trajectories ending at $x \to s_f$:

$$P(x) = \sum_{\tau:(x \to s_f) \in \tau} P_F(\tau) \propto R(x)$$

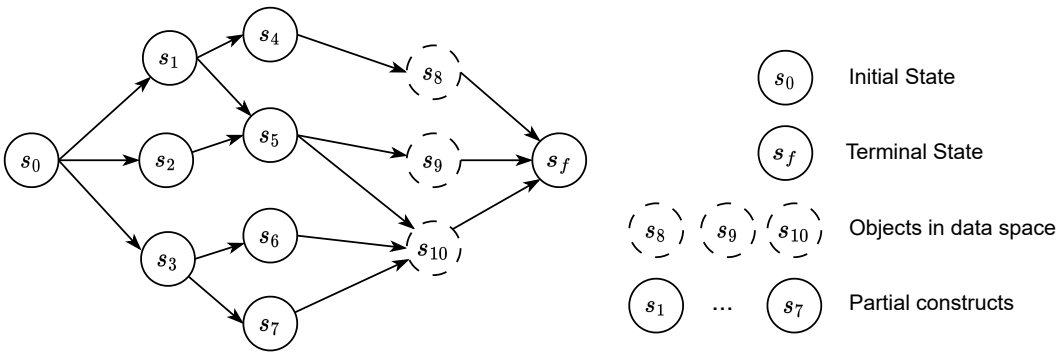



**Figure 3.2:** Example GFlowNet states graph

For large scale problems, it is unrealistic to calculate this term explicitly because the calculation involves finding all trajectories between $s_0$ and $x$, and there is no efficient method to do so. Zhang et al. (2022) proposes to estimate $P(x)$ with importance sampling. While we cannot enumerate efficiently all trajectories ending at $x \to s_f$, one can sample backwards trajectories starting at $x$ with uniform sampling policy: at current state, uniformly choose a parent state and transition backward until reaching $s_0$. The uniform backward transition probabilities are then defined as following:

$$P_B(\tau|x) = \prod_{s_i \to s_{i+1} \in \tau} P_B(s_i|s_{i+1}) \quad P_B(s_i|s_{i+1}) = \frac{1}{|\alpha(s_{i+1})|}$$

where $\alpha(s_{i+1})$ denote the set of parents of $s_{i+1}$. With uniform backward sampling policy, we can estimate the $P(x)$ by first sampling $K$ trajectories with backward policy, then aggregating as following:

$$P(x) \approx \frac{1}{K} \sum_{\tau_i : s_0 \to \cdots \to x} \frac{P_F(\tau_i)}{P_B(\tau_i|x)}$$

Given the reward $R(x)$, the goal of GFlowNet is to estimate a policy $P_F(\cdot|\cdot;,\theta)$ such that $P(x) \propto R(x)$. Multiple Learning objectives are designed specifically for GFlowNet (Bengio et al., 2021; Malkin et al., 2022; Pan et al., 2023; Madan et al., 2023), we review two commonly used learning objective

**Detailed-balance (DB)** The DB objective is proposed in (Bengio et al., 2021). The loss function learns the following three functions in parametric form:

- A forward policy $P_F(s'|s;,\theta)$ representing the sampling policy of the GFlowNet
- A backward policy $P_B(s|s';,\theta)$ representing a probability distribution over the parent states $s$ for $s' \in \mathcal{S} - \{s_f\}$.
- A state flow function $F(s,\theta) : \mathcal{S} \to \mathbb{R}^+$ representing the un-normalized probability mass of all trajectories that pass through state $s$

For any pair of transition $s \to s'$, the detailed balance loss is defined as:

$$\mathcal{L}_{\text{DB}}(s, s', \theta) = \begin{cases} \left(\log \frac{F(s,\theta)P_F(s'|s;\theta)}{F(s',\theta)P_B(s|s';\theta)}\right)^2 & \text{if } s' \notin \mathcal{X} \\ \left(\log \frac{F(s,\theta)P_F(s'|s;\theta)}{R(s')P_B(s|s';\theta)}\right)^2 & \text{otherwise} \end{cases}$$

**Trajectory-balance (TB)** The TB objective is proposed in Malkin et al. (2022). The loss function learns the three following objectives:

- A forward policy $P_F(s'|s;,\theta)$ representing the sampling policy of the GFlowNet
- A backward policy $P_B(s|s';,\theta)$ representing a probability distribution over the parent states $s$ for $s' \in \mathcal{S} - \{s_f\}$.
- A partition function $Z_\theta$ representing the total reward of all objects in $\mathcal{X}$

For any trajectory $\tau : s_0 \to \cdots \to x \to s_f$, the TB loss is defined as:

$$\mathcal{L}_{\text{TB}}(\tau) = \left(\log \frac{Z_\theta \prod_{i=0}^{n-1} P_F(s_{i+1} \mid s_i;\theta)}{R(x)\prod_{i=1}^{n} P_B(s_{i-1}|s_i);\theta}\right)^2$$

For both loss objectives, a backward policy $P_B(s|s';,\theta)$ is learned to faciliate training for the forward policy $P_F(s'|s;,\theta)$. Actually, Bengio et al. (2023) shows that $P_B(s|s')$ can be chosen freely as long as $\sum_{s\in\alpha(s')} P_B(s|s') = 1$. Therefore, an alternative choice is to use the uniform backward sampling policy: $P_B(s|s') = \frac{1}{|\alpha(s')|}$.

Training a GFlowNet follows a similar procedure as training an RL model. The training data consists of trajectories. These can be online trajectories sampled directly using the current GFlowNet. To encourage exploration, one can introduce random perturbations when generating training trajectories. For example: with $\epsilon$-greedy training, $\epsilon$ percent of actions to build trajectories are drawn from uniform random. Trajectories can also be offline, we can store high reward / high loss trajectories in a replay buffer, and feed the model with trajectories sampled from the replay buffer. Also, if we know ahead of time some important objects in data space, we can generate trajectories that lead to these objects with backward sampling.

### 3.2.1 Advantage and disadvantage of GFlowNets

The main advantage of GFlowNets lies in its representation power. With a sequential construction procedure, complex objects can be sampled with GFlowNets through a sequence of simple actions. For large scale problems, while construction trajectories are intractable, they are well encapsulated within the GFlowNet MDP. For any reward function that may lead to arbitrarily complex and multiple modal target distributions, GFlowNet provides the theoretical guarantee that there exist a GFlowNet sampler that samples from the target distribution (Bengio et al., 2023). Moreover, GFlowNet is a machine learning algorithm, thus it can generalize on unseen data. While GFlowNet cannot visit and learn every object/ trajectories for a large scale problem, it can approximate their probabilities as long as there are underlying learnable patterns. GFlowNet has great similarities with RL algorithms as most of RL algorithms also operates within the MDP environment. Therefore, research works in RL algorithms can be transferred to improve GFlowNet algorithms. For example: inspired by Distributional Reinforcement Learning (Bellemare et al., 2017), Zhang et al. (2023c) propose Distributional GFlowNets that can handle environment where rewards are stochastic. Comparing with MCMC, an alternative sampling algorithms, GFlowNet produces an **amortized sampler**: past sampling objects are used to train GFlowNet. Once GFlowNet is trained, at anytime we can use the trained model

to sample directly from the target distribution. In contrast, past samples do not impact the decision making of MCMC, so every time to sample based on target distribution, one has to execute a long run until convergence. The disadvantage of GFlowNet also lies in its complexity. To train a neural-network based GFlowNet requires more hardware resources. Also, if the target distribution does not have a learnable statistical structure, GFlowNet will have a hard to time generalize (Bengio et al., 2023).

## 3.3 Applications of GFlowNet

GFlowNets have been applied to generate diverse optimal solutions for discrete optimization problem. Let $\epsilon(x)$ be the objective score function to be minimized for a given optimization problem. One can design a corresponding Boltzmann distribution $P(x) = \exp\left(\frac{-\epsilon(x)}{T}\right)$ where $T$ is the statistical temperature. For sufficiently small $T$, the distribution will be dominated by the set of optimal solutions. GFlowNet learns to sample solutions from the distribution $P(x)$. One advantage to use GFlowNet is that if there exist multiple optimal solutions, their sampling probability are the same. Therefore with an appropriate $T$, an optimally learned GFlowNet can output multiple diverse optimal solutions instead of one. Zhang et al. (2023b) use GFlowNets to solve graph combinatorial optimization problems such as maximum independent set and maximum clique. Zhang et al. (2023a) use GFlowNet to generate efficient plans to execute computation graphs. In section 4.5.2, we set $\epsilon(x)$ to be the parsimony score of phylogenetic tree $x$, and solve the parsimony problem with GFlowNets.

GFlowNets have also been applied in AI driven design and discovery tasks. For instance, Bengio et al. (2021) use GFlowNets to design small molecules for drug discovery. Cipcigan et al. (2023) utilize GFlowNets to design reticular materials for $CO_2$ capture. Jain et al. (2022) leverage GFlowNet to design biological sequences. Zhu et al. (2023) use GFlowNets to incorporate multiple optimization objectives for the creation of diverse molecules aligning with Pareto optimality. While ML algorithms have succesfully pro-

duce AI-agent that outperforms human in various complex tasks, such as playing GO (Silver et al., 2016), these tasks often come with clear reward metrics for optimization. Moreover, these algorithms require extensive training on vast datasets. These conditions can be challenging to satisfy for real-world design and discovery tasks. Jain et al. (2023) outlines the main challenges in AI-driven scientific design tasks, which are also relevant to design tasks in general: in many real-life problems, unlike video games or simulated environments, the true optimization goal cannot be directly modeled as a training objective for machine learning algorithms. Consider designing drug molecules to treat a specific disease, for example; it is not feasible to have a numerical reward that measures the effectiveness of "treating a disease" for every type of molecule in the dataset. If we hypothesize that the goal can be achieved by optimizing certain measurable metrics, we can design algorithm to generate objects to maximize this auxiliary metrics. However, there's no guarantee that the objects with the maximum metrics score are optimal for our true goal. Moreover, collecting a labeled dataset for training can be both expensive and time-consuming. Labels may be noisy, especially when dealing with data generated from biological experiments. This results in training data that is not only scarce but also noisy, making it challenging to train a reliable machine learning model. In such scenarios, it is ideal for the machine learning algorithm not just to output the object with the maximum score, but also to generate diverse high-scoring objects that can be investigated by human experts in a subsequent stage. Additionally, learning the posterior distribution allows us to estimate uncertainties. GFlowNet learns a sampler in a way that the sampling distribution is proportional to the target reward. Consequently, it serves as a suitable algorithm for AI-driven design tasks in real-life scenarios where the optimization goal cannot be directly modeled or when obtaining quality training data is challenging.

# Chapter 4

# PhyloGFN: Phylogenetic inference with generative flow networks

Mingyang Zhou [1] Zichao Yan [1,2] Elliot Layne [1,2] Nikolay Malkin [2,3] Dinghuai Zhang [2,3]
Moksh Jain [2,3] Mathieu Blanchette [1] & Yoshua Bengio [2,3,4]

[1] McGill University, [2] MILA, Quebec AI Institute, [3] Université de Montréal, [4] CIFAR

## 4.1 Abstract

Phylogenetics is a branch of computational biology that studies the evolutionary relationships among biological entities. Its long history and numerous applications notwithstanding, inference of phylogenetic trees from sequence data remains challenging: the extremely large tree space poses a significant obstacle for the current combinatorial and probabilistic techniques. In this paper, we adopt the framework of generative flow networks (GFlowNets) to tackle two core problems in phylogenetics: parsimony-based and Bayesian phylogenetic inference. Because GFlowNets are well-suited for sampling complex combinatorial structures, they are a natural choice for exploring and sampling from the multimodal posterior distribution over tree topologies and evolutionary distances. We demonstrate that our amortized posterior sampler, PhyloGFN, produces diverse and high-quality evolutionary hypotheses on real benchmark datasets. PhyloGFN is competitive with prior works in marginal likelihood estimation and achieves a closer fit to the target distribution than state-of-the-art variational inference methods. Our code is available at `https://github.com/zmy1116/phylogfn`.

## 4.2 Introduction

Phylogenetic inference has long been a central problem in the field of computational biology. Accurate phylogenetic inference is critical for a number of important biological analyses, such as understanding the development of antibiotic resistance (Aminov & Mackie, 2007; Ranjbar et al., 2020; Layne et al., 2020), assessing the risk of invasive species (Hamelin et al., 2022; Dort et al., 2023), and many others. Accurate phylogenetic trees can also be used to improve downstream computational analyses, such as multiple genome alignment (Blanchette et al., 2004), ancestral sequence reconstruction (Ma et al., 2006), protein structure and function annotation (Celniker et al., 2013).

Despite its strong medical relevance and wide applications in life science, phylogenetic inference has remained a standing challenge, in part due to the high complexity

of tree space — for $n$ species, $(2n-5)!!$ unique unrooted bifurcating tree topologies exist. This poses a common obstacle to all branches of phylogenetic inference; both maximum-likelihood and maximum-parsimony tree reconstruction are NP-hard problems (Day, 1987; Chor & Tuller, 2005). Under the Bayesian formulation of phylogenetics, the inference problem is further compounded by the inclusion of continuous variables that capture the level of sequence divergence along each branch of the tree.

One line of prior work considers Markov chain Monte Carlo (MCMC)-based approaches, such as MrBayes (Ronquist et al., 2012). These approaches have been successfully applied to Bayesian phylogenetic inference. However, a known limitation of MCMC is scalability to high-dimensional distributions with multiple separated modes (Tjelmeland & Hegstad, 2001), which arise in larger phylogenetic datasets. Recently, variational inference (VI)-based approaches have emerged. Among these methods, some model only a limited portion of the space of tree topologies, while others are weaker in marginal likelihood estimation due to simplifying assumptions. In parsimony analysis, state-of-the-art methods such as PAUP* (Swofford, 1998) have extensively relied on heuristic search algorithms that are efficient but lack theoretical foundations and guarantees.

Coming from the intersection of variational inference and reinforcement learning is the class of models known as generative flow networks (GFlowNets; Bengio et al., 2021). The flexibility afforded by GFlowNets to learn sequential samplers for distributions over compositional objects makes them a promising candidate for performing inference over the posterior space of phylogenetic tree topologies and evolutionary distances.

In this work, we propose **PhyloGFN**, the first adaptation of GFlowNets to the task of Bayesian and parsimony-based phylogenetic inference. Our contributions are as follows:

1. We design an acyclic Markov decision process (MDP) with fully customizable reward functions, by which our PhyloGFN can be trained to construct phylogenetic trees in a bottom-up fashion.

2. PhyloGFN leverages a novel tree representation inspired by Fitch and Felsenstein's algorithms to represent rooted trees without introducing additional learnable pa-

rameters to the model. PhyloGFN is also coupled with simple yet effective training techniques such as using mixed on-policy and dithered-policy rollouts, replay buffers and cascading temperature-annealing.

3. PhyloGFN has the capacity to explore and sample from the entire phylogenetic tree space, achieving a balance between exploration in this vast space and high-fidelity modeling of the modes. While PhyloGFN performs on par with the state-of-the-art MCMC- and VI-based methods in the summary metric of marginal log-likelihood, it substantially outperforms these approaches in terms of its ability to estimate the posterior probability of suboptimal trees.

## 4.3 Related work

Markov chain Monte Carlo (MCMC)-based algorithms are commonly employed for Bayesian phylogenetics, with notable examples including MrBayes and RevBayes (Ronquist et al., 2012; Höhna et al., 2016), which are considered state-of-the-art in the field. Amortized variational inference (VI) is an alternative approach that parametrically estimates the posterior distribution. VBPI-GNN (Zhang, 2023) employs subsplit Bayesian networks (SBN) (Zhang & Matsen IV, 2018a) to model tree topology distributions and uses graph neural networks to learn tree topological embeddings (Zhang, 2023). While VBPI-GNN has obtained marginal log likelihood competitive with MrBayes in real datasets, it requires a pre-generated set of high-quality tree topologies to constrain its action space for tree construction, which ultimately limits its ability to model the entire tree space.

There exist other VI approaches that do not limit the space of trees. VaiPhy (Koptagel et al., 2022) approximates the posterior distribution in the augmented space of tree topologies, edge lengths, and ancestral sequences. Combined with combinatorial sequential Monte Carlo (CSMC; Moretti et al., 2021), the proposed method enables faster estimation of marginal likelihood. GeoPhy (Mimori & Hamada, 2023) models the tree topology distribution in continuous space by mapping continuous-valued coordinates to tree

topologies, using the same technique as VBPI-GNN to model tree topological embeddings. While both methods model the entire tree topology space, their performance on marginal likelihood estimation underperforms the state of the art.

For the optimization problem underpinning maximum parsimony inference, PAUP* is one of the most commonly used programs (Swofford, 1998); it features several fast, greedy, and heuristic algorithms based on local branch-swapping operations such as tree bisection and reconnection.

GFlowNets are a family of methods for sampling discrete objects from multimodal distributions, such as molecules (Bengio et al., 2021) and biological sequences (Jain et al., 2022), and are used to solve discrete optimization tasks (Zhang et al., 2023a,b). With their theoretical foundations laid out in Bengio et al. (2023); Lahlou et al. (2023), and connections to variational inference established in Malkin et al. (2023), GFlowNets have been successfully applied to tackle complex Bayesian inference problems, such as inferring latent causal structures in gene regulatory networks (Deleu et al., 2022, 2023), and parse trees in hierarchical grammars (Hu et al., 2023).

## 4.4 Background

### 4.4.1 Phylogenetic inference

Here we introduce the problems of Bayesian and parsimony-based phylogenetic inference. A weighted phylogenetic tree is denoted by $(z, b)$, where $z$ represents the tree topology with its leaves labeled by observed sequences, and $b$ represents the branch lengths. The tree topology can be either a rooted binary tree or a bifurcating unrooted tree. For a tree topology $z$, let $L(z)$ denote the labeled sequence set and $E(z)$ the set of edges. For an edge $e \in E(z)$, let $b(e)$ denote its length. Let $\boldsymbol{Y} = \{\boldsymbol{y}_1, \boldsymbol{y}_2 \dots \boldsymbol{y}_n\} \in \Sigma^{n \times m}$ be a set of $n$ observed sequences, each having $m$ characters from alphabet $\Sigma$, *e.g.*, $\{A, C, G, T\}$ for DNA sequences. We denote the $i^{\text{th}}$ site of all sequences by $\boldsymbol{Y}^i = \{\boldsymbol{y}_1[i], \boldsymbol{y}_2[i] \dots \boldsymbol{y}_n[i]\}$. In this work, we make two assumptions that are common in the phylogenetic inference

literature: (i) sites evolve independently; (ii) evolution follows a time-reversible substitution model. The latter implies that an unrooted tree has the same parsimony score or likelihood as its rooted versions, and thus the algorithms we introduce below (Fitch and Felsenstein) apply to unrooted trees as well.

**Bayesian inference**

In Bayesian phylogenetic inference, we are interested in sampling from the posterior distribution over weighted phylogenetic trees $(z, b)$, formulated as:

$$P(z, b \mid \boldsymbol{Y}) = \frac{P(z, b)P(\boldsymbol{Y} \mid z, b)}{P(\boldsymbol{Y})}$$

where $P(\boldsymbol{Y} \mid z, b)$ is the likelihood, $P(\boldsymbol{Y})$ is the intractable marginal, and $P(z, b)$ is the prior density over tree topology and branch lengths. Under the site independence assumption, the likelihood can be factorized as: $P(\boldsymbol{Y} \mid z, b) = \prod_i P(\boldsymbol{Y}^i \mid z, b)$, and each factor is obtained by marginalizing over all internal nodes $a_j$ and their possible character assignment:

$$P(\boldsymbol{y}_1[i] \dots \boldsymbol{y}_n[i] \mid z, b) = \sum_{a^i_{n+1}, \dots a^i_{2n-1}} P(a_{2n-1}) \prod_{j=n+1}^{2n-2} P(a^i_j | a^i_{\alpha(j)}, b(e_j)) \prod_{k=1}^{n} P(\boldsymbol{y}_k[i] | a^i_{\alpha(k)}, b_z(e_k))$$

where $a^i_{n+1}, \dots a^i_{2n-2}$ represent the internal node characters assigned to site $i$ and $\alpha(i)$ represent the parent of node $i$. $P(a_{2n-1})$ is a distribution at the root node, which is usually assumed to be uniform over the vocabulary, while the conditional probability $P(a^i_j | a^i_{\alpha(j)}, b(e_j))$ is defined by the substitution model (where $e_j$ is the edge linking $a_j$ to $\alpha(a_j)$).

**Felsenstein's algorithm** The likelihood of a given weighted phylogenetic tree can be calculated efficiently using Felsenstein's pruning algorithm (Felsenstein, 1973) in a bottom-up fashion through dynamic programming. Defining $L^i_u$ as the leaf sequence characters

at site $i$ below the internal node $u$, and given its two child nodes $v$ and $w$, the conditional probability $P(L_u^i|a_u^i)$ can be obtained from $P(L_v^i|a_v^i)$ and $P(L_w^i|a_w^i)$:

$$P(L_u^i \mid a_u^i) = \sum_{a_v^i, a_w^i \in \Sigma} P(a_v^i \mid a_u^i, b(e_v)) P(L_v^i \mid a_v^i) P(a_w^i \mid a_u^i, b(e_w)) P(L_w^i \mid a_w^i). \tag{4.1}$$

The dynamic programming, or recursion, is essentially a post-order traversal of the tree and $P(L_u^i \mid a_u^i)$ is calculated at every internal node $u$, and we use one-hot encoding of the sequences to represent the conditional probabilities at the leaves. Finally, the conditional probability for each node $u$ at site $i$ is stored in a data structure $\boldsymbol{f}_u^i \in [0,1]^{|\Sigma|}$: $\boldsymbol{f}_u^i[c] = P(L_u^i|a_u^i = c)$, and we call it the **Felsenstein feature** for node $u$. Note that the conditional probability at the root $P(\boldsymbol{Y}^i|a_{2n-1}^i)$ is used to calculate the likelihood of the tree: $P(\boldsymbol{Y}^i|z,b) = \sum_{a_{2n-1}^i \in \Sigma} P(a_{2n-1}^i) P(\boldsymbol{Y}^i|a_{2n-1}^i)$.

**Parsimony analysis**

The problem of finding the optimal tree topology under the maximum parsimony principle is commonly referred as the Large Parsimony problem, which is NP-hard. For a given tree topology $z$, the parsimony score is the minimum number of character changes between sequences across branches obtained by optimally assigning sequences to internal nodes. Let $M(z|\boldsymbol{Y})$ be the parsimony score of tree topology $z$ with leaf labels $\boldsymbol{Y}$. Due to site independence, $M(z|\boldsymbol{Y}) = \sum_i M(z|\boldsymbol{Y}^i)$. The trees with the lowest parsimony score, or most parsimonious trees, are solutions to the Large Parsimony problem. Note that the Large Parsimony problem is a limiting case of the maximum likelihood tree inference problem, where branch lengths are constrained to be equal and infinitesimally short.

**Fitch algorithm** Given a rooted tree topology $z$, the Fitch algorithm assigns optimal sequences to internal nodes and computes the parsimony score in linear time. At each node $u$, the algorithm tracks the set of possible characters labeling for node $u$ that can yield a most parsimonious solution for the subtree rooted at $u$. This character set can

be represented by a binary vector $\boldsymbol{f}_u^i \in \{0,1\}^{|\Sigma|}$ for site $i$. We label this vector the **Fitch feature**. As in Felsenstein's algorithm, this vector is a one-hot encoding of the sequences at the leaves and is computed recursively for non-leaves. Specifically, given a rooted tree with root $u$ and two child trees with roots $v$ and $w$, $\boldsymbol{f}_u^i$ is calculated as:

$$\boldsymbol{f}_u^i = \begin{cases} \boldsymbol{f}_v^i \wedge \boldsymbol{f}_w^i & \text{if } \boldsymbol{f}_v^i \cdot \boldsymbol{f}_w^i \neq 0 \\ \boldsymbol{f}_v^i \vee \boldsymbol{f}_w^i & \text{otherwise} \end{cases},$$

where $\wedge$ and $\vee$ are element-wise conjunctions and disjunctions. The algorithm traverses the tree two times, first in post-order (bottom-up) to calculate the character set at each node, then in pre-order (top-down) to assign optimal sequences. The total number of character changes between these optimal sequences along the tree's edges is counted as the parsimony score.

### 4.4.2 GFlowNets

Generative flow networks (GFlowNets) are algorithms for learning generative models of complex distributions given by unnormalized density functions over structured spaces. Here, we give a concise summary of the the GFlowNet framework.

**Setting** A GFlowNet treats generation of objects $x$ lying in a sample space $\mathcal{X}$ as a sequential decision-making problem on an acyclic deterministic MDP with set of states $\mathcal{S} \supset \mathcal{X}$ and set of actions $\mathcal{A} \subseteq \mathcal{S} \times \mathcal{S}$. The MDP has a designated *initial state* $s_0$, which has no incoming actions, and a set of *terminal states* (those with no outgoing actions) that coincides with $\mathcal{X}$. Any $x \in \mathcal{X}$ can be reached from $s_0$ by a sequence of actions $s_0 \to s_1 \to \cdots \to s_n = x$ (with each $(s_i, s_{i+1}) \in \mathcal{A}$). Such sequences are called *complete trajectories*, and the set of all complete trajectories is denoted $\mathcal{T}$.

A *(forward) policy* $P_F$ is a collection of distributions $P_F(s' \mid s)$ over the children of each nonterminal state $s \in \mathcal{S} \setminus \mathcal{X}$. A policy induces a distribution over $\mathcal{T}$:

$$P_F(\tau = (s_0 \to s_1 \to \cdots \to s_n)) = \prod_{i=0}^{n-1} P_F(s_{i+1} \mid s_i).$$

A policy gives a way to sample objects in $\mathcal{X}$, by sampling a complete trajectory $\tau \sim P_F$ and returning its final state, inducing a marginal distribution $P_F^\top$ over $\mathcal{X}$; $P_F^\top(x)$ is the sum of $P_F(\tau)$ over all complete trajectories $\tau$ that end in $x$ (a possibly intractable sum).

The goal of GFlowNet training is to estimate a parametric policy $P_F(\cdot \mid \cdot; \theta)$ such that the induced $P_F^\top$ is proportional to a given *reward function* $R : \mathcal{X} \to \mathbb{R}_{>0}$, *i.e.*,

$$P_F^\top(x) = \frac{1}{Z} R(x) \quad \forall x \in \mathcal{X}, \tag{4.2}$$

where $Z = \sum_{x \in \mathcal{X}} R(x)$ is the unknown normalization constant (partition function).

**Trajectory balance objective** The direct optimization of $P_F$'s parameters $\theta$ is impossible since it involves an intractable sum over all complete trajectories. Instead, we leverage the trajectory balance (TB) training objective (Malkin et al., 2022), which introduces two auxiliary objects: an estimate $Z_\theta$ of the partition function and a *backward policy*. In our experiments, we fix the backward policy to uniform, which results in a simplified objective:

$$\mathcal{L}_{\mathrm{TB}}(\tau) = \left( \log \frac{Z_\theta \prod_{i=0}^{n-1} P_F(s_{i+1} \mid s_i; \theta)}{R(x) P_B(\tau \mid x)} \right)^2, \quad P_B(\tau \mid x) := \prod_{i=1}^{n} \frac{1}{|\mathrm{Pa}(s_i)|}, \tag{4.3}$$

where $\mathrm{Pa}(s)$ denotes the set of parents of $s$. By the results of Malkin et al. (2022), there exists a unique policy $P_F$ and scalar $Z_\theta$ that simultaneously make $\mathcal{L}_{\mathrm{TB}}(\tau) = 0$ for all $\tau \in \mathcal{T}$, and at this optimum, the policy $P_F$ satisfies (4.2) and $Z_\theta$ equals the true partition function $Z$. In practice, the policy $P_F(\cdot \mid s; \theta)$ is parameterized as a neural network that outputs the logits of actions $s \to s'$ given a representation of the state $s$ as input, while $Z_\theta$ is parameterized in the log domain.

$\mathcal{L}_{\text{TB}}(\tau)$ is minimized by gradient descent on trajectories $\tau$ chosen from a behaviour policy that can encourage exploration to accelerate mode discovery (see our behaviour policy choices in §4.5.3).

## 4.5 Phylogenetic inference with GFlowNets

### 4.5.1 GFlowNets for Bayesian phylogenetic inference

This section introduces PhyloGFN-Bayesian, our GFlowNet-based method for Bayesian phylogenetic inference. Given a set of observed sequences $\boldsymbol{Y}$, PhyloGFN-Bayesian learns a sampler over the joint space of tree topologies and edge lengths $\mathcal{X} = \{(z, b)|L(z) = \boldsymbol{Y}\}$ such that the sampling probability of $(z, b) \in \mathcal{X}$ approximates its posterior $P_F^{\top}(z, b) = P(z, b|\boldsymbol{Y})$.

We follow the same setup as Koptagel et al. (2022); Zhang (2023); Mimori & Hamada (2023): (i) uniform prior over tree topologies; (ii) decomposed prior $P(z, b) = P(z)P(b)$; (iii) exponential ($\lambda = 10$) prior over branch lengths; (iv) Jukes-Cantor substitution model.

**GFlowNet state and action space** The sequential procedure of constructing phylogenetic trees is illustrated in Fig. 4.1. The initial state $s_0$ is a set of $n$ rooted trees, each containing a single leaf node labeled with an observed sequence. Each action chooses a pair of trees and joins their roots by a common parent node, thus creating a new tree. The number of rooted trees in the set is reduced by 1 at every step, so after $n-1$ steps a single rooted tree with $n$ leaves is left. To obtain an unrooted tree, we simply remove the root node.

Thus, a state $s$ consists of a set of disjoint rooted trees $s = ((z_1, b_1), \ldots, (z_l, b_l)),\ \ l \leq n$ and $\bigcup_i L(z_i) = \boldsymbol{Y}$. Given a nonterminal state with $l > 1$ trees, a transition action consists of two steps: (i) choosing a pair of trees to join out of the $\binom{l}{2}$ possible pairs; and (ii) generating branch lengths for the two introduced edges between the new root and its two children. The distribution over the pair of branch lengths is modeled jointly as a

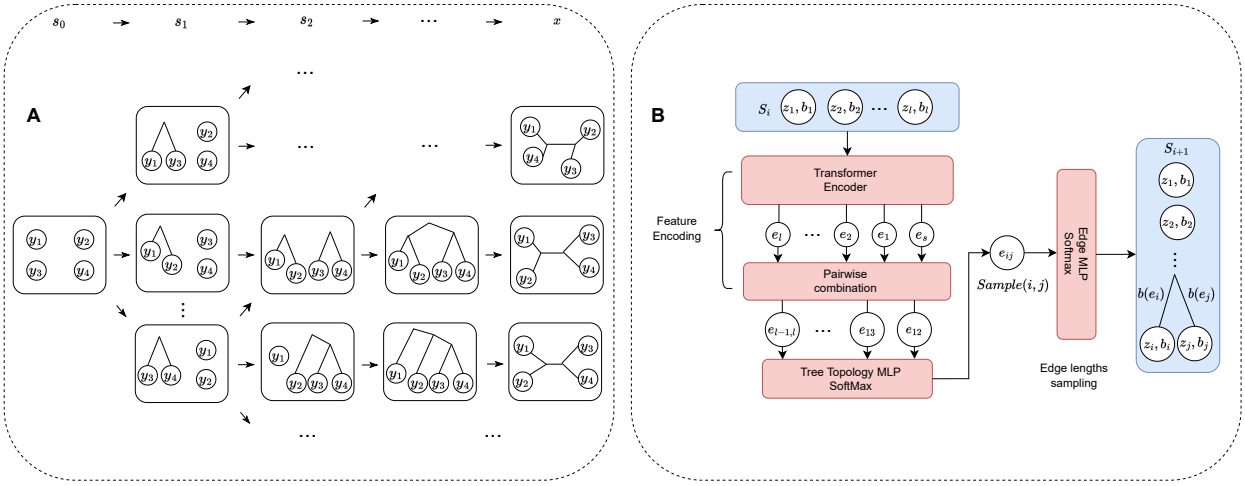



**Figure 4.1: Left:** PhyloGFN's state space on a four-sequence dataset. Initial state $s_0$ comprises leaf nodes. Successive steps merge pairs of trees until a single unrooted tree remains. **Right:** Policy model for PhyloGFN-Bayesian. Transformer encoder processes tree-level features $s_i = ((z_1, b_1) \dots (z_l, b_l))$. Pairwise features $e_{ij}$ are derived and used by MLPs to select tree pairs for merging and sample branch lengths.

discrete distribution with fixed bin size. Following the initial submission of the paper, we conducted additional experiments by employing continuous distribution to model branch lengths. Further details can be found in §4.8.5 of the appendix.

**Reward function** We define the reward function as the product of the likelihood and the edge length prior: $R(z, b) = P(\boldsymbol{Y}|z, b)P(b)$, implicitly imposing a uniform prior over tree topologies. By training with this reward, PhyloGFN learns to approximate the posterior, since $P(z, b|\boldsymbol{Y}) = R(z, b)\frac{P(z)}{P(\boldsymbol{Y})}$ and $P(z), P(\boldsymbol{Y})$ are both constant.

It is worth emphasizing that in our bottom-up construction of trees, the set of possible actions at the steps that select two trees to join by a new common root is never larger than $n^2$, even though the size of the space of all tree topologies – all of which can be reached by our sampler – is superexponential in $n$. This stands in contrast to the modeling choices of VBPI-GNN (Zhang, 2023), which constructs trees in a top-down fashion and limits the action space using a pre-generated set of trees, therefore also limiting the set of trees it can sample.

**State representation** To represent a rooted tree in a non-terminal state, we compute features for each site independently by taking advantage of the Felsenstein features (§4.4.1). Let $(z, b)$ be a weighted tree with root $u$ which has two children $v$ and $w$. Let $\boldsymbol{f}_u^i, \boldsymbol{f}_v^i, \boldsymbol{f}_w^i \in [0,1]^{|\Sigma|}$ be the Felsenstein feature on nodes $u, v, w$ at site $i$. The representation $\rho_u^i$ for site $i$ is computed as following:

$$\rho_u^i = \boldsymbol{f}_u^i \prod_{e \in E(z)} P(b_z(e))^{\frac{1}{m}} \tag{4.4}$$

where $P(b_z(e)) = \prod_{e \in b_z} P(b(e))$ is the edge length prior. The tree-level feature is the concatenation of site-level features $\rho = [\rho^1 \dots \rho^m]$. A state $s = (z_1 \dots z_l)$, which is a collection of rooted trees, is represented by the set $\{\rho_1, \dots, \rho_l\}$.

**Representation power** Although the proposed feature representation $\rho$ does not capture all the information of tree structure and leaf sequences, we show that $\rho$ indeed contains sufficient information to express the optimal policy. To elaborate, Bengio et al. (2023) proves that a unique optimal GFlowNet policy $P_F$ with a uniform backward policy $P_B$ exists for the defined problem. This optimal policy can be expressed by the transition probability function $P_F(s'|s)$, which takes the input the entirety of a non-terminal state $s$, and outputs a distribution over successor states $s'$. Proposition 1 demonstrates that given an optimal GFlowNet $P_F$ with a uniform $P_B$, two states with identical feature sets share the same transition probabilities. Therefore, this optimal transition probability function is equivalent to a transition function that only takes input of the representation feature.

**Proposition 1.** *Let* $s_1 = \{(z_1, b_1), (z_2, b_2) \dots (z_l, b_l)\}$ *and* $s_2 = \{(z'_1, b'_1), (z'_2, b'_2) \dots (z'_l, b'_l)\}$ *be two non-terminal states such that* $s_1 \neq s_2$ *but sharing the same features* $\rho_i = \rho'_i$*. Let* $\boldsymbol{a}$ *be any sequence of actions, which applied to* $s_1$ *and* $s_2$*, respectively, results in full weighted trees* $x = (z, b_z), x' = (z', b')$*, with two partial trajectories* $\tau = (s_1 \to \dots \to x), \tau' = (s_2 \to \dots \to x')$*. If* $P_F$ *is the policy of an optimal GFlowNet with uniform* $P_B$*, then* $P_F(\tau) = P_F(\tau')$*.*

Before proving proposition 1, we first prove the following two lemmas. First, we show that for two states sharing the same tree features, applying the same action to the states results in two new states still sharing the same features.

**Lemma 1.** *Let* $s_1 = \{(z_1, b_1), (z_2, b_2) \dots (z_l, b_l)\}$ *and* $s_2 = \{(z'_1, b'_1), (z'_2, b'_2) \dots (z'_l, b'_l)\}$ *be two non-terminating states sharing the same features* $\rho_i = \rho'_i$*. Let* $a$ *be the action that joins the trees with indices* $(v, w)$ *to form a new tree indexed* $u$ *with edge lengths* $(b(e_{uv}), b(e_{uw}))$*. By applying* $a$ *on* $s_1$*, we join* $(z_v, b_v)$ *and* $(z_w, b_w)$ *to form new tree* $(z_u, b_u)$*. By applying* $a$ *on* $s_2$*, we join* $(z'_v, b'_v)$ *and* $(z'_w, b'_w)$ *to form new tree* $(z'_u, b'_u)$*. Then the new trees' features are equal:* $\rho_u = \rho'_u$*.*

*Proof.* We show that $\rho_u$ can be calculated from $\rho_v$ and $\rho_w$:

$$\rho_u^i[j] = P(b(e_{uv}))^{\frac{1}{m}} \times P(b(e_{uw}))^{\frac{1}{m}} \times \sum_{k \in \Sigma} P(a_u^i = j | a_v^i = k, b_z(e_{uv})) \rho_v^i[k]$$
$$\times \sum_k P(a_u^i = j | a_w^i = k, b(e_{uw})) \rho_w^i[k]$$

since $\rho_v = \rho'_v$, $\rho_w = \rho'_w$ and $(b(e_{uv}), b(e_{uw}))$ are new branch lengths for both two trees. Therefore $\rho_u = \rho'_u$ □

Next, we show that for two states sharing the same tree features, applying the same action sequences results in two phylogenetic trees with the same reward.

**Lemma 2.** *Let* $s_1 = \{(z_1, b_1), (z_2, b_2) \dots (z_l, b_l)\}$ *and* $s_2 = \{(z'_1, b'_1), (z'_2, b_2) \dots (z'_l, b_l)\}$ *be two non-terminating states sharing the same features* $\rho_i = \rho'_i$*. Let* $\boldsymbol{a}$ *be any sequence of actions to apply on* $s_1$ *and* $s_2$ *to form full trees* $x = (z, b_z), x' = (z', b'_z)$*,* $R(z, b) = R(z', b')$*.*

*Proof.* Let $\rho_u$ denote the tree feature for $(z, b)$, $\rho_u^i = \boldsymbol{f}_u^i \prod_e P(b(e))$. We first show that the reward can be directly calculated from the root feature $\rho_u$:

$$\prod_i P(a_{2n-1}) \cdot \rho_u = \prod_e P(b(e)) \prod_i P(a_{2n-1}) \cdot f_u^i$$
$$= P(b) P(\boldsymbol{Y} | z, b)$$
$$= R(z, b),$$

where $P(a_{2n-1})$ is the constant root character assignment probability. As $\boldsymbol{a}$ is applied to $s_1$ and $s_2$ in a sequential manner, at every step we obtain two state swith the same tree features (by Lemma 1), until, at the final state, $\rho_u = \rho'_u$. It follows that $R(z, b) = R(z', b')$. □

We are now ready to prove the propositions.

*Proof.* Let $\mathcal{G}_{s_1}, \mathcal{G}_{s_2}$ be sub-graphs of the GFlowNet state graph $\mathcal{G} = (\mathcal{S}, \mathcal{A})$ defined by reachable states from $s_1$ and $s_2$ in $\mathcal{G}$. Since $s_1$ and $s_2$ have the same number of trees, $\mathcal{G}_{s_1}$ and $\mathcal{G}_{s_2}$ have the graph structure. Let $\mathcal{X}_1, \mathcal{X}_2 \subseteq \mathcal{X}$ be the terminal states reachable from $s_1$ and $s_2$. There is thus a bijective correspondence between $\mathcal{X}_1$ and $\mathcal{X}_2$: for every action set $\boldsymbol{a}$ applying on $s_1$ to obtain $x \in \mathcal{X}_1$, we obtain $x' \in \mathcal{X}_2$ by applying $\boldsymbol{a}$ on $s_2$. Let $\tau$ and $\tau'$ be the partial trajectories created by applying $\boldsymbol{a}$ on $s_1$ and $s_2$, $P_B(\tau|x) = P_B(\tau'|x')$. Moreover, $R(x) = R(x')$ since $s_1$ and $s_2$ share the same set of features. We have the following expressions for $P_F(\tau)$ and $P_F(\tau')$:

$$P_F(\tau) = \frac{R(x)P_B(\tau|x)}{\sum_{\tau_j, x_j} R(x_j)P_B(\tau_j|x_j)}, \quad P_F(\tau') = \frac{R(x')P_B(\tau'|x')}{\sum_{\tau'_j, x'_j} R(x'_j)P_B(\tau'_j|x'_j)}.$$

Hence $P_F(\tau) = P_F(\tau')$.

□

The proposition shows that our proposed features have sufficient representation power for the PhyloGFN-Bayesian policy. Furthermore, Felsenstein features and edge length priors are used in calculating reward by Felsenstein's algorithm. Therefore, computing these features does not introduce any additional variables, and computation overhead is minimized.

### 4.5.2 GFlowNets for parsimony analysis

This section introduces PhyloGFN-Parsimony, our GFlowNet-based method for parsimony analysis. We treat large parsimony analysis, a discrete optimization problem, as a

sampling problem from the energy distribution $\exp\left(\frac{-M(z|\boldsymbol{Y})}{T}\right)$ defined over tree topologies. Here, $M(z|\boldsymbol{Y})$ is the parsimony score of $z$ and $T$ is a pre-defined temperature term to control the smoothness of distribution. With sufficiently small $T$, the most parsimonious trees dominate the energy distribution. To state our goals formally, given observed sequences $\boldsymbol{Y}$, PhyloGFN-Parsimony learns a sampling policy $P_F$ over the space of tree topologies $\{z|L(z) = \boldsymbol{Y}\}$ such that $P_F^{\top}(z) \propto e^{-\frac{M(z|Y)}{T}}$. As $T \to 0$, this target distribution approaches a uniform distribution over the set of tree topologies with minimum parsimony scores.

PhyloGFN-Parsimony can be seen as a reduced version of PhyloGFN-Bayesian. The tree shape generation procedure is the same as before, but we no longer generate branch lengths. The reward is defined as $R(z) = \exp\left(\frac{C-M(z|Y)}{T}\right)$, where $C$ is an extra hyperparameter introduced for stability to offset the typically large $M(z|\boldsymbol{Y})$ values. Note that $C$ can be absorbed into the partition function and has no influence on the reward distribution.

Similar to PhyloGFN-Bayesian, a state (collection of rooted trees) is represented by the set of tree features, with each tree represented by concatenating its site-level features. With $z$ the rooted tree topology with root $u$, we represent the tree at site $i$ by its root level Fitch feature $\boldsymbol{f}_u^i$ defined in §4.4.1. The proposition below, analogous to Proposition 1, shows the representation power of the proposed feature.

**Proposition 2.** *Let $s_1 = \{z_1, z_2, \dots z_l\}$ and $s_2 = \{z'_1, z'_2, \dots z'_l\}$ be two non-terminal states such that $s_1 \neq s_2$ but sharing the same Fitch features $\boldsymbol{f}_{z_i} = \boldsymbol{f}_{z'_i}\ \forall i$. Let $\boldsymbol{a}$ be any sequence of actions, which, applied to $s_1$ and $s_2$, respectively, results in tree topologies $x, x' \in \mathcal{Z}$, with two partial trajectories $\tau = (s_1 \to \cdots \to x), \tau' = (s_2 \to \cdots \to x')$. If $P_F$ is the policy of an optimal GFlowNet with uniform $P_B$, then $P_F(\tau) = P_F(\tau')$*

*Proof.* We use the same notation as in the proof of of Proposition 1. Since $s_1$ and $s_2$ have the same number of trees, $\mathcal{G}_{s_1}$ and $\mathcal{G}_{s_2}$ have the same graph structure. Let $\mathcal{X}_1, \mathcal{X}_2 \subseteq \mathcal{X}$ be the terminal states reachable from $s_1$ and $s_2$. There is a bijective correspondence between $\mathcal{X}_1$ and $\mathcal{X}_2$: for every action set $\boldsymbol{a}$ applying on $s_1$ to obtain $x \in \mathcal{X}_1$, we obtain $x' \in \mathcal{X}_2$ by

applying $\boldsymbol{a}$ on $s_2$. Let $\tau$ and $\tau'$ be the partial trajectories created by applying $\boldsymbol{a}$ on $s_1$ and $s_2$, $P_B(\tau|x) = P_B(\tau'|x')$.

For simplicity, we denote $M(x) = M(x|L(x))$. It is worth to note that for two tree topologies sharing the same Fitch feature, their parsimony scores do not necessarily equal. However, when applying $\boldsymbol{a}$ on two states $s_1$ and $s_2$ sharing the Fitch feature, the additional parsimony scores introduced are the same:

$$M(x) - \sum_i M(z_i) = M(x') - \sum_i M(z'_i)$$

We have the following expressions for $P_F(\tau)$ and $P_F(\tau')$:

$$P_F(\tau) = \frac{e^{\frac{-M(x)}{T}} P_B(\tau|x)}{\sum_{\tau_j, x_j} e^{\frac{-M(x_j)}{T}} P_B(\tau_j|x_j)}, \quad P_F(\tau') = \frac{e^{\frac{-M(x')}{T}} P_B(\tau|x')}{\sum_{\tau'_j, x'_j} e^{\frac{-M(x'_j)}{T}} P_B(\tau'_j|x'_j)}$$

We multiply $\frac{e^{\frac{\sum_i M(z_i)}{T}}}{e^{\frac{\sum_i M(z_i)}{T}}}$ by $P_F(\tau)$ and $\frac{e^{\frac{\sum_i M(z'_i)}{T}}}{e^{\frac{\sum_i M(z'_i)}{T}}}$ by $P_F(\tau')$ and obtain:

$$P_F(\tau) = \frac{e^{\frac{\sum_i M(z_i) - M(x)}{T}} P_B(\tau|x)}{\sum_{\tau_j, x_j} e^{\frac{\sum_i M(z_i) - M(x_j)}{T}} P_B(\tau_j|x_j)}, \quad P_F(\tau') = \frac{e^{\frac{\sum_i M(z'_i) - M(x')}{T}} P_B(\tau|x')}{\sum_{\tau'_j, x'_j} e^{\frac{\sum_i M(z'_i) - M(x'_j)}{T}} P_B(\tau'_j|x'_j)}$$

Since $e^{\frac{\sum_i M(z'_i) - M(x'_j)}{T}} P_B(\tau'_j|x'_j) = e^{\frac{\sum_i M(z_i) - M(x_j)}{T}} P_B(\tau_j|x_j)$ for all $j$, $P_F(\tau) = P_F(\tau')$. □

This shows that the Fitch features contain sufficient information for PhyloGFN-Parsimony. Furthermore, the Fitch features are used in the computation of the reward by Fitch's algorithm, so their use in the policy model does not introduce additional variables or extra computation.

**Temperature-conditioned PhyloGFN** The temperature $T$ controls the trade-off between sample diversity and parsimony scores. Following Zhang et al. (2023a), we extend PhyloGFN-

Parsimony by conditioning the policy on $T$, with reward $R(z;T) = \exp\left(\frac{C-M(z|\boldsymbol{Y})}{T}\right)$, and we learn a sampler such that $P^{\top}(z;T) \propto R(z;T)$. See Appendix 4.8.4 for more details.

### 4.5.3 Model architecture and training

**Parameterization of forward transitions** We parameterize the forward transitions of tree topology construction using a Transformer-based neural network (Vaswani et al., 2017b), whose architecture is shown in Fig. 4.1. We select Transformer because the input is a set and the model needs to be order-equivariant. For a state consisting of $n$ trees, after $n$ embeddings are generated from the Transformer encoder, $\binom{n}{2}$ pairwise features are created for all possible pairs of trees, and a common MLP generates probability logits for joining every tree pair. PhyloGFN-Bayesian additionally requires generating edge lengths. Once the pair of trees to join is selected, another MLP is applied to the corresponding pair feature to generate probability logits for sampling the edge lengths. See more details in 4.8.3.

**Off-policy training** The action model $P_F(\cdot \mid s;\theta)$ is trained with the trajectory balance objective. Training data is generated from two sources: (i) A set of trajectories constructed from the currently trained GFlowNet, with actions sampled from the policy with probability $1-\epsilon$ and uniformly at random with probability $\epsilon$. The $\epsilon$ rate drops from a predefined $\epsilon_{start}$ to near 0 during the course of training. (ii) Trajectories corresponding to the best trees seen to date (replay buffer). Trajectories are sampled backward from these high-reward trees with uniform backward policy.

**Temperature annealing** For PhyloGFN-Parsimony, it is crucial to choose the appropriate temperature $T$. Large $T$ defines a flat target distribution, while small $T$ makes the reward landscape less smooth and leads to training difficulties. We cannot predetermine the ideal choice of $T$ before training, as we do not know *a priori* the parsimony score for

the dataset. Therefore, we initialize the training with large $T$ and reduce $T$ periodically during training. See Appendix 4.8.4 for details.

### 4.5.4 Marginal log-likelihood estimation

To assess how well the GFlowNet sampler approximates the true posterior distribution, we use the following importance-weighted variational lower bound on the marginal log-likelihood (MLL):

$$\log P(\boldsymbol{Y}) \geq \mathbb{E}_{\tau_1,\ldots,\tau_k \sim P_F} \log \left( P(z) \frac{1}{K} \sum_{\substack{\tau_i \\ \tau_i : s_0 \to \cdots \to (z_i, b_i)}}^{k} \frac{P_B(\tau_i | z_i, b_i) R(z_i, b_i)}{P_F(\tau_i)} \right). \tag{4.5}$$

Our estimator is computed by sampling a batch of $K$ trajectories and averaging $\frac{P(z)P_B(\tau|z,b)R(z,b)}{P_F(\tau)}$ over the batch. This expectation of this estimate is guaranteed to be a lower bound on $\log P(\boldsymbol{Y})$ and its bias decreases as $K$ grows (Burda et al., 2016).

PhyloGFN-Bayesian models branch lengths with discrete multinomial distributions, while in reality these are continuous variables. To properly estimate the MLL and compare with other methods defined over continuous space, we augment our model to a continuous sampler by performing random perturbations over edges of trees sampled from PhyloGFN-Bayesian. The perturbation follows the uniform distribution $\mathcal{U}_{[-0.5\omega, 0.5\omega]}$, where $\omega$ is the fixed bin size for edge modeling in PhyloGFN-Bayesian. The resulting model over branch lengths is then a piecewise-constant continuous distribution. We discuss the computation details as well as the derivation of (4.5) in Appendix 4.8.1.

## 4.6 Experiments

We evaluate PhyloGFN on a suite of 8 real-world benchmark datasets (Table 4.3 in Appendix 4.8.2) that is standard in the literature. These datasets feature pre-aligned sequences and vary in difficulty (27 to 64 sequences; 378 to 2520 sites). In the following

sections we present our results and analysis on Bayesian and parsimony-based phylogenetic inference.

### 4.6.1 Bayesian phylogenetic inference

PhyloGFN is compared with a variety of baselines in terms of sampling-based estimates of marginal log-likelihood (MLL; see details in §4.5.4). The baselines we compare to are MrBayes SS, Stepping-Stone sampling algorithm implemented in MrBayes (Ronquist et al., 2012), and three variational inference methods: VBPI-GNN (Zhang, 2023), $\phi$-CSMC proposed in VaiPhy (Koptagel et al., 2022), and GeoPhy (Mimori & Hamada, 2023). The sampling setup for MrBayes follows Zhang & Matsen IV (2018b) and otherwise show the highest MLL reported in the respective papers. PhyloGFN MLL is estimated following the formulation in §4.5.4, with mean and standard deviation obtained from 10 repetitions, each using 1000 samples. See Sections 4.8.4 and 4.8.6 for additional training details, hardware specifications, and running time comparison. Additional results from two repeated experiments are in Table 4.7.

The results are summarized in Table 4.6. Our PhyloGFN is markedly superior to $\phi$-CSMC across all datasets and outperforms GeoPhy on most, with the exception of DS2 and DS3 where the two perform similarly, and DS7 where GeoPhy obtains a better result. VBPI-GNN, is the only machine learning-based method that performs on par against MrBayes, the current gold standard in Bayesian phylogenetic inference. However, it should be emphasized that VBPI-GNN requires a set of pre-defined tree topologies that are likely to achieve high likelihood, and as a consequence, its training and inference are both constrained to a small space of tree topologies.

On the other hand, PhyloGFN operates on the full space of tree topologies and, in fact, achieves a closer fit to the true posterior distribution. To show this, for each dataset, we created three sets of phylogenetic trees with high/medium/low posterior probabilities and obtained the corresponding sampling probabilities from PhyloGFN and VBPI-GNN. The three classes of trees are generated from VBPI-GNN by randomly inserting uniformly

**Table 4.1:** Marginal log-likelihood estimation with different methods on real datasets DS1-DS8. PhyloGFN outperforms $\phi$-CSMC across all datasets and GeoPhy on most. *VBPI-GNN uses predefined tree topologies in training and is not directly comparable.

| | MCMC | | ML-based / amortized, full tree space | | |
|---|---|---|---|---|---|
| Dataset | MrBayes SS | VBPI-GNN* | $\phi$-CSMC | GeoPhy | PhyloGFN |
| DS1 | $-7108.42_{\pm 0.18}$ | $-7108.41_{\pm 0.14}$ | $-7290.36_{\pm 7.23}$ | $-7111.55_{\pm 0.07}$ | $\mathbf{-7108.95}_{\pm 0.06}$ |
| DS2 | $-26367.57_{\pm 0.48}$ | $-26367.73_{\pm 0.07}$ | $-30568.49_{\pm 31.34}$ | $\mathbf{-26368.44}_{\pm 0.13}$ | $\mathbf{-26368.90}_{\pm 0.28}$ |
| DS3 | $-33735.44_{\pm 0.5}$ | $-33735.12_{\pm 0.09}$ | $-33798.06_{\pm 6.62}$ | $\mathbf{-33735.85}_{\pm 0.12}$ | $\mathbf{-33735.6}_{\pm 0.35}$ |
| DS4 | $-13330.06_{\pm 0.54}$ | $-13329.94_{\pm 0.19}$ | $-13582.24_{\pm 35.08}$ | $-13337.42_{\pm 1.32}$ | $\mathbf{-13331.83}_{\pm 0.19}$ |
| DS5 | $-8214.51_{\pm 0.28}$ | $-8214.64_{\pm 0.38}$ | $-8367.51_{\pm 8.87}$ | $-8233.89_{\pm 6.63}$ | $\mathbf{-8215.15}_{\pm 0.2}$ |
| DS6 | $-6724.07_{\pm 0.86}$ | $-6724.37_{\pm 0.4}$ | $-7013.83_{\pm 16.99}$ | $-6733.91_{\pm 0.57}$ | $\mathbf{-6730.68}_{\pm 0.54}$ |
| DS7 | $-37332.76_{\pm 2.42}$ | $-37332.04_{\pm 0.12}$ | | $\mathbf{-37350.77}_{\pm 11.74}$ | $-37359.96_{\pm 1.14}$ |
| DS8 | $-8649.88_{\pm 1.75}$ | $-8650.65_{\pm 0.45}$ | $-9209.18_{\pm 18.03}$ | $-8660.48_{\pm 0.78}$ | $\mathbf{-8654.76}_{\pm 0.19}$ |

sampled actions into its sequential tree topology construction process with $0\%$, $30\%$, or $50\%$ probability, respectively, which circumvents VBPI-GNN's limitation of being confined to a small tree topology space. Table 4.2 and Fig. 4.2 show that PhyloGFN achieves higher Pearson correlation between the sampling log-probability and the unnormalized ground truth posterior log-density for the majority of datasets and classes of trees. In particular, while VBPI-GNN performs better on high-probability trees, its correlation drops significantly on lower-probability trees. On the other hand, PhyloGFN maintains a high correlation for all three classes of trees across all datasets, the only exception being the high-probability trees in DS7. See Appendix 4.8.9 for details and extended results and Appendix 4.8.8 for a short explanation of the significance of modeling suboptimal trees.

**Continuous branch length modeling** Following the initial submission, additional experiments involving continuous branch length modeling demonstrate PhyloGFN's ability to achieve state-of-the-art Bayesian inference performance. For more details, please see Appendix 4.8.5.

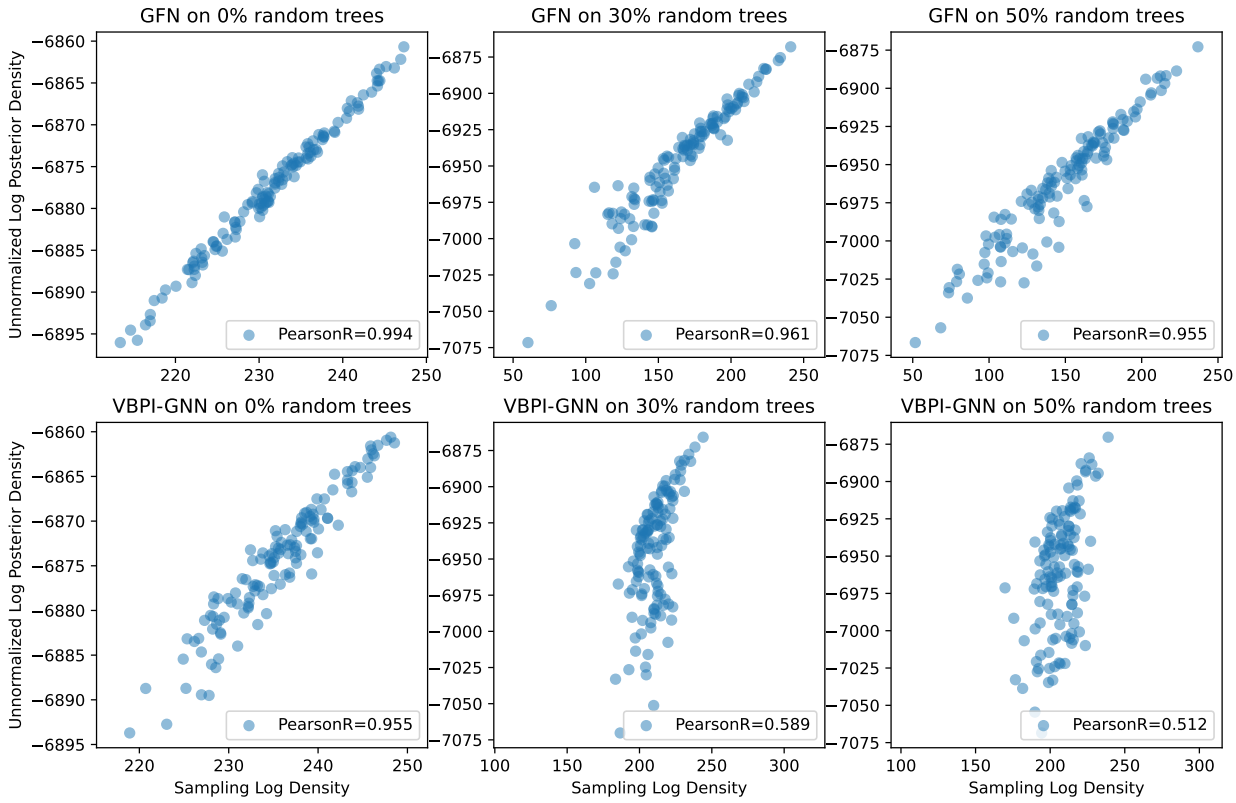



**Figure 4.2:** Model sampling log-density vs. unnormalized posterior log-density for high/medium/low-probability trees on DS1. We highlight that PhyloGFN-Bayesian performs significantly better on medium- and low-probability trees, highlighting its superiority in modeling the entire data space.

### 4.6.2 Parsimony-based phylogenetic inference

As a special case of Bayesian phylogenetic inference, the parsimony problem is concerned with finding the most-parsimonious trees – a task which is also amenable to PhyloGFN. Here, we compare to the state-of-the-art parsimony analysis software PAUP* (Swofford, 1998). For all datasets, our PhyloGFN and PAUP* are able to identify the same set of optimal solutions to the Large Parsimony problem, ranging from a single optimal tree for DS1 to 21 optimal trees for DS8.

Although the results are similar between PhyloGFN and PAUP*, once again we emphasize that PhyloGFN is based on a rigorous mathematical framework of fitting and

**Table 4.2:** Pearson correlation of model sampling log-density and ground truth unnormalized posterior log-density for each dataset on high/medium/low posterior density trees generated by VBPI-GNN. PhyloGFN achieves a good fit on both high and low posterior density regions.

| | No random | | 30% random | | 50% random | |
|---|---|---|---|---|---|---|
| Dataset | PhyloGFN | VBPI-GNN | PhyloGFN | VBPI-GNN | PhyloGFN | VBPI-GNN |
| DS1 | **0.994** | 0.955 | **0.961** | 0.589 | **0.955** | 0.512 |
| DS2 | 0.930 | **0.952** | **0.948** | 0.580 | **0.919** | 0.538 |
| DS3 | 0.917 | **0.968** | **0.963** | 0.543 | **0.950** | 0.499 |
| DS4 | 0.942 | **0.960** | **0.945** | 0.770 | **0.966** | 0.76 |
| DS5 | **0.969** | 0.965 | **0.937** | 0.786 | **0.939** | 0.773 |
| DS6 | **0.993** | 0.887 | **0.973** | 0.816 | **0.934** | 0.702 |
| DS7 | 0.624 | **0.955** | **0.787** | 0.682 | **0.764** | 0.678 |
| DS8 | **0.978** | 0.955 | **0.913** | 0.604 | **0.901** | 0.463 |

sampling from well-defined posterior distributions over tree topologies. whereas PAUP*'s relies on heuristic algorithms. To put it more concretely, we show in Fig. 4.6 that PhyloGFN is able to (i) learn a smooth echelon of sampling probabilities that distinguish the optimal trees from suboptimal ones; (ii) learn similar sampling probabilities for trees within each group of equally-parsimonious trees; and (iii) fit all $2n - 3$ rooted trees that belong to the same unrooted tree equally well.

Finally, Fig. 4.3 shows that a single temperature-conditioned PhyloGFN can sample phylogenetic trees of varying ranges of parsimony scores by providing suitable temperatures $T$ as input to the model. Also, PhyloGFN is able to sample proportionately from the Boltzmann distribution defined at different temperatures and achieves high correlation between sampling log-probability and log-reward. Although the temperature-conditioned PhyloGFN has only been trained on a small range of temperatures between 4.0 and 1.0, Fig. 4.3 shows it can also approximate the distribution defined at temperature 8.0. Further results are presented in Appendix 4.8.10.

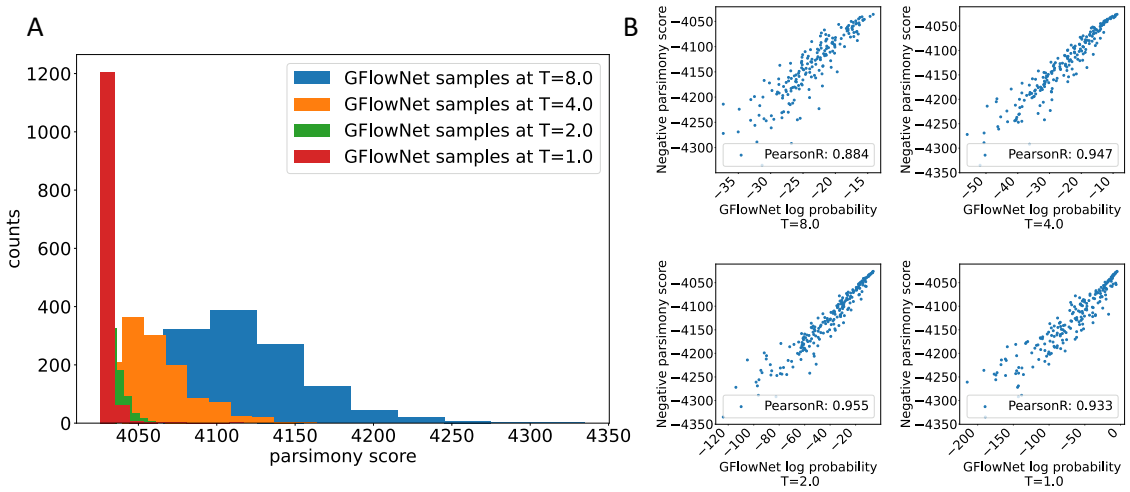



**Figure 4.3:** A temperature-conditioned PhyloGFN is trained on DS1 using temperatures sampled between 4.0 and 1.0. **(A)** The distribution of parsimony scores can be controlled by varying the statistical temperature – an input variable to the PhyloGFN policy – from 8.0 to 1.0. 10,000 trees are randomly sampled at each temperature. **(B)** PhyloGFN achieves high Pearson correlation for trees sampled within each temperature range.

## 4.7 Discussion and future work

In this paper, we propose PhyloGFN, a GFlowNet-based generative modeling algorithm, to solve parsimony-based and Bayesian phylogenetic inference. We design an intuitive yet effective tree construction procedure to efficiently model the entire tree topology space. We propose a novel tree representation based on Fitch's and Felsenstein's algorithms to represent rooted trees without introducing additional learnable parameters, and we show the sufficiency of our features for the purpose of phylogenetic tree generation. We apply our algorithm on eight real datasets, demonstrating that PhyloGFN is competitive with or superior to prior works in terms of marginal likelihood estimation, while achieving a closer fit to the target distribution compared to state-of-the-art variational inference methods. While our initial experiments with continuous branch length sampling have demonstrated notable performance enhancements, there remains a need for future research to address training efficiency. In addition, we plan to explore the use of conditional GFlowNets to amortize the dependence on the sequence dataset itself. This would

allow a single trained GFlowNet to sample phylogenetic trees for sets of sequences that were not seen during training.

## 4.8 Appendix

### 4.8.1 Marginal log-likelihood estimation

**Estimating the sampling likelihood of a terminal state** In the PhyloGFN state space (in both the Bayesian and parsimony-based settings), there exist multiple trajectories leading to the same terminal state $x$, hence the sampling probability of $x$ is calculated as: $P^{\top}(x) = \sum_{\tau:s_0\to\cdots\to x} P_F(\tau)$. This sum is intractable for large-scale problems. However, we can estimate the sum using importance sampling(Zhang et al., 2022):

$$P_F^{\top}(x) \approx \frac{1}{K} \sum_{\tau_i:s_0\to\cdots\to x} \frac{P_F(\tau_i)}{P_B(\tau_i|x)}, \tag{4.6}$$

where the trajectories $\tau_i$ are sampled from $P_B(\tau_i \mid x)$. The logarithm of the right side of (4.6) is, in expectation, a lower bound on the logarithm of the true sampling likelihood on the left side of (4.6).

**Estimating the MLL** The lower bound on MLL can be evaluated using the importance-weighted bound $\log P(\boldsymbol{Y}) \geq \mathbb{E}_{x_1\ldots x_k\sim P_F} \log \frac{1}{K} \sum \frac{P(x,\boldsymbol{Y})}{P^T(x)}$(Burda et al., 2016). However, we cannot use it for PhyloGFN since we cannot compute the exact $P^T(x)$, only get a lower bound on it using (4.6). Therefore, we propose the following variational lower bound:

$$\log P(\boldsymbol{Y}) \geq \mathbb{E}_{\tau_1,\ldots,\tau_k\sim P_F} \log \left( P(z) \frac{1}{K} \sum_{\substack{\tau_i \\ \tau_i:s_0\to\cdots\to(z_i,b_i)}}^{k} \frac{P_B(\tau_i|z_i,b_i)R(z_i,b_i)}{P_F(\tau_i)} \right)$$

We show the derivation of the estimator and thus prove its consistency:

$$
\begin{aligned}
P(\boldsymbol{Y}) &= \int_{z,b} P(\boldsymbol{Y}|z,b)P(b|z)P(z) \\
&= \int_{z,b} R(z,b)P(z) \\
&= \int_{z,b} R(z,b)P(z) \sum_{\tau:s_0\ldots x=(z,b)} P_B(\tau|z,b) \\
&= \int_{z,b} R(z,b)P(z) \sum_{\tau:s_0\ldots x=(z,b)} P_B(\tau|z,b)\frac{P_F(\tau)}{P_F(\tau)} \\
&= \int_{z,b} \sum_{\tau:s_0\ldots x=(z,b)} P_F(\tau)\frac{P_B(\tau|z,b)R(z,b)P(z)}{P_F(\tau)} \\
&= \int_{\tau:s_0\ldots x_\tau=(z_\tau,b_\tau)} P_F(\tau)\frac{P_B(\tau|z_\tau,b_\tau)R(z_\tau,b_\tau)P(z)}{P_F(\tau)} \\
&= P(z)\mathbb{E}_{\tau\sim P_F}\frac{P_B(\tau|z_\tau,b_\tau)R(z_\tau,b_\tau)}{P_F(\tau)} \\
&\approx P(z)\frac{1}{K}\sum_{\substack{\tau_i\sim P_F \\ \tau_i:s_0\ldots(z_i,b_i)}}^{k} \frac{P_B(\tau_i|z_i,b_i)R(z_i,b_i)}{P_F(\tau_i)}.
\end{aligned}
$$

One can show, in a manner identical to the standard importance-weighted bound, that this estimate is a lower bound on $\log P(\boldsymbol{Y})$.

PhyloGFN-Bayesian models edge lengths using discrete distributions. To estimate our algorithm's MLL, we augment the sampler to a continuous sampler by modeling branch lengths with a piecewise-constant continuous distribution based on the fixed-bin multinomial distribution of PhyloGFN. We can still use the above formulation to estimate the lower bound. However, each trajectory now has one extra step: $\tau' = s_0 \to \cdots \to (z,b) \to (z,b_i)$ where $(z,b_i)$ is obtained from $z,b$ by randomly perturbing each branch length by adding noise from $U[-0.5\omega, 0.5\omega]$, where $\omega$ is the bin size used for PhyloGFN. Let $\tau = s_0 \to \cdots \to (z,b)$ be the original trajectory in the discrete PhyloGFN, we can compute $P_F(\tau'), P_B(\tau')$ from $P_F(\tau), P_B(\tau)$:

$$
P_F(\tau') = P_F(\tau)\frac{1}{\omega^{|E(z)|}}, \quad P_B(\tau') = P_B(\tau)
$$

The term $\frac{1}{\omega^{|E(z)|}}$ is introduced in $P_F$ because we additionally sample over a uniform range $\omega$ for all $|E(z)|$ edges. The backward probability $P_B$ stays unchanged because given $(z, b)$ generated from the discrete GFN, for any $(z, b')$ resulting from perturbing edges, $(z, b)$ is the the only possible ancestral state.

### 4.8.2 Dataset information

**Table 4.3:** Statistics of the benchmark datasets from DS1 to DS8.

| Dataset | # Species | # Sites | Reference |
|---|---|---|---|
| DS1 | 27 | 1949 | Hedges et al. (1990) |
| DS2 | 29 | 2520 | Garey et al. (1996) |
| DS3 | 36 | 1812 | Yang & Yoder (2003) |
| DS4 | 41 | 1137 | Henk et al. (2003) |
| DS5 | 50 | 378 | Lakner et al. (2008) |
| DS6 | 50 | 1133 | Zhang & Blackwell (2001) |
| DS7 | 59 | 1824 | Yoder & Yang (2004) |
| DS8 | 64 | 1008 | Rossman et al. (2001) |

### 4.8.3 Modeling

Given the character set $\Sigma$, we use one-hot encoding to represent each site in a sequence. To deal with wild characters in the dataset, for parsimony analysis we consider them as one special character in $\Sigma$, while in Bayesian inference, we represent them by a vector of $\mathbb{1}$.

For both PhyloGFN-Parsimony and PhyloGFN-Bayesian, given a state with $l$ rooted trees, its representation feature is the set $\{\rho_1 \dots \rho_l\}$, where $\rho$ is a vector of length $m|\Sigma|$. For example, for DNA sequences of 1000 sites, each $\rho$ would have length 4000. Therefore, before passing these features to the Transformer encoder, we first use a linear transformation to obtain lower-dimensional embeddings of the input features.

We use the Transformer architecture (Vaswani et al., 2017a) to build the **Transformer encoder**. For a state with $l$ trees, the output is a set of $l+1$ features $\{e_s, e_1, \dots, e_l\}$ where $e_s$ denotes the summary feature (*i.e.*, the [CLS] token of the Transformer encoder input).

To select pairs of trees to join, we evaluate tree-pair features for every pair of trees in the state and pass these tree-pair features as input to the **tree MLP** to generate probability logits for all pairs of trees. The tree-pair feature for a tree pair $(i, j)$ with representations $e_i$, $s_j$ is the concatenation of $e_i + e_j$ with the summary embedding of the state, *i.e.*, the feature is $[e_s; e_i + e_j]$, where $[\cdot;\cdot]$ denotes vector direct sum (concatenation). For a state with $l$ trees, $\binom{l}{2} = \frac{l(l-1)}{2}$ such pairwise features are generated for all possible pairs.

To generate edge lengths for the joined tree pair $(i, j)$, we pass $[e_s; e_i; e_j]$ – the concatenation of the summary feature with the tree-level features of trees $i$ and $j$ – as input to the **edge MLP**. For unrooted tree topologies we need to distinguish two scenarios: (i) when only two rooted trees are left in the state (*i.e.*, at the last step of PhyloGFN state transitions), we only need to generate a single edge; and (ii) when there are more than two rooted trees in the state, a pair of edges is required. Therefore, two separate edge MLPs are employed, as edge length is modeled by $k$ discrete bins, the edge MLP used at the last step has an output size of $k$ (to model a distribution over a single edge length) whereas

the other edge MLP would have an output size of $k^2$ (to model a joint distribution over two edge lengths).

For the temperature-conditioned PhyloGFN, as then temperature $T$ is passed to our PhyloGFN as an input, two major modifications are required: (i) the estimation of the partition $Z_\theta$ is now a function of $T$: $Z_\theta(T)$, which is modeled by a **Z MLP**; (ii) the summary token to the Transformer encoder also captures the temperature information by replacing the usual [CLS] token with a **temp MLP** that accepts $T$ as input.

### 4.8.4 Training details

Here, we describe the training details for our PhyloGFN. For PhyloGFN-Bayesian, our models are trained with fixed 500 epochs. For PhyloGFN-Parsimony, our models are trained until the probability of sampling the optimal trees, or the most parsimonious trees our PhyloGFN has seen so far, is above 0.001. Each training epoch consists of 1000 gradient update steps using a batch of 64 trajectories. For $\epsilon$-greedy exploration, the $\epsilon$ value is linearly annealed from 0.5 to 0.001 during the first 240 epochs. All common hyperparameters for PhyloGFN-Bayesian and PhyloGFN-Parsimony are shown in Table 4.5.

**Temperature annealing** For PhyloGFN-Bayesian, the initial temperature is set to $16$ for all experiments. For PhyloGFN-Parsimony, $T$ is initialized at 4. Under the cascading temperature annealing scheme, $T$ is reduced by half per every 80 epochs of training. For PhyloGFN-Bayesian, $T$ is always reduced to and then fixed at 1, whereas for PhyloGFN-Parsimony, $T$ is only reduced when the most parsimonious trees seen by our PhyloGFN so far are sampled with a probability below 0.001.

**Hyperparameter $C$ selection** For PhyloGFN-Parsimony, the reward is defined as $R(x) = \exp\left(\frac{C - M(x|\boldsymbol{Y})}{T}\right)$, where $C$ is a hyperparameter introduced for training stability and it controls the magnitude of the partition function $Z = \sum_x R(x)$. Given that we cannot determine the precise value of $C$ prior to training since we do know the value of $Z$ as a priori,

we use the following heuristic to choose $C$: 1000 random tree topologies are generated via stepwise-addition, and we set $C$ to the $10\%$ quantile of the lowest parsimony score.

Similarly, $C$ is employed for PhyloGFN-Bayesian under the same goal of stabilizing the training and reducing the magnitude of the partition function. Recall the reward function $R(z, b)$ defined in section 4.5.1, it can be rewritten as $R(z, b) = \exp\left(\frac{C-(-\log P(\boldsymbol{Y}|z,b)P(b))}{T}\right)$ when $T = 1$. Note that $\exp\left(\frac{C}{T}\right)$ can be absorbed into the partition function. For selecting the $C$, once again we randomly sample 1000 weighted phylogenetic trees via stepwise-addition and with random edge length, followed by setting $C$ to the $10\%$ quantile of the lowest $-\log P(\boldsymbol{Y}|z, b)P(b)$.

**Temperature-conditioned PhyloGFN-Parsimony** The temperature-conditioned PhyloGFN is introduced so that a single trained PhyloGFN can be used to sample from a series of reward distributions defined by different $T$. We modify the fixed-temperature PhyloGFN-Parsimony by introducing $T$ as input in 3 places: (i) the reward function $R(x; T)$; (ii) the forward transition policy $P_F(x; T)$; and (iii) the learned partition function estimate $Z_\theta(T)$. To train the temperature-conditioned PhyloGFN, the TB objective also needs to be updated accordingly:

$$\mathcal{L}_{\text{TB}}(\tau; T) = \left(\log \frac{Z_\theta(T) \prod_{i=0}^{n-1} P_F(s_{i+1} \mid s_i; \theta, T)}{R(x, T) P_B(\tau \mid x)}\right)^2, \quad P_B(\tau \mid x) := \prod_{i=1}^{n} \frac{1}{|\text{Pa}(s_i)|},$$

Note that $P_B$ is unaffected.

During training, values for $T$ are randomly selected from the range $[T_{\min}, T_{\max}]$. When training with a state from the replay buffer, the temperature used for training is resampled and may be different than the one originally used when the state was added to the buffer.

We also employ a scheme of gradually reducing the average of sampled $T$ values through the course of training: $T$'s are sampled from a truncated normal distribution with a fixed pre-defined variance and a moving mean $T_\mu$ that linearly reduces from $T_{\max}$ to $T_{\min}$.

**Modeling branch lengths with discrete multinomial distribution** When a pair of trees are joined at the root, the branch lengths of the two newly formed edges are modeled jointly by a discrete multinomial distribution. The reason for using a joint distribution is because under the Jukes-Cantor evolution model, the likelihood at the root depends on the sum of the two branch lengths.

We use a different set of maximum edge length, bin number and bin size depending on each dataset, and by testing various combinations we have selected the set with optimal performance. The maximum branch length is chosen among $\{0.05, 0.1, 0.2\}$, and bin size $\omega$ is chosen among $\{0.001, 0.002, 0.004\}$. Table 4.4 shows the our final selected bin size and bin number for each dataset.

**Table 4.4:** Bin sizes and bin numbers used to model branch lengths for DS1 to DS8.

| Dataset | Bin Size | # Bins |
|---|---|---|
| DS1 | 0.001 | 50 |
| DS2 | 0.004 | 50 |
| DS3 | 0.004 | 50 |
| DS4 | 0.002 | 100 |
| DS5 | 0.002 | 100 |
| DS6 | 0.001 | 100 |
| DS7 | 0.001 | 200 |
| DS8 | 0.001 | 100 |

**Table 4.5:** Common hyperparameters for PhyloGFN-Bayesian and PhyloGFN-Parsimony.

| Transformer encoder | |
|---|---|
| hidden size | 128 |
| number of layers | 6 |
| number of heads | 4 |
| learning rate (model) | 5e-5 |
| learning rate (Z) | 5e-3 |
| **tree MLP** | |
| hidden size | 256 |
| number of layers | 3 |
| **edge MLP** | |
| hidden size | 256 |
| number of layers | 3 |
| **Z MLP (in temperature-conditioned PhyloGFN)** | |
| hidden size | 128 |
| number of layers | 1 |
| **temp MLP (in temperature-conditioned PhyloGFN)** | |
| hidden size | 256 |
| number of layers | 3 |

### 4.8.5 Continuous branch length modeling with PhyloGFN

While the original PhyloGFN-Bayesian only samples discrete branch lengths, we have experimented with a new version of PhyloGFN that uses mixture of Gaussian to model and sample branch lengths — effectively treating them as continuous variables. We refer to this new version as PhyloGFN-Continuous and we point out that the edge modeling is the only implementational detail that differs from the previous models.

Specifically, the **edge MLP** of PhyloGFN-Continuous now outputs the parameters of the Gaussian mixture which models the logarithm of branch length. These include (1) logits for selecting the components of the Gaussian mixture, (2) mean of the log branch length in each mixture, and (3) log variance of the log branch length in each mixture.

Although PhyloGFN-Continuous continues to employ two separate **edge MLP**, one for the last step of state transition where only a single edge needs to be sampled and the other for sampling a pair of edges at the intermediate step, PhyloGFN-Continuous now samples each edge independently as we have found this to benefit training stability.

It is also worth pointing out that we no longer perform $\epsilon$-greedy random exploration on branch lengths, again for the sake of training stability, and the mean of the log branch length is initialized at $-4.0$ instead of $0.0$, which is a more reasonable value for log branch length. There are five components in each Gaussian mixture.

Results are shown below. We are delighted to report that PhyloGFN-Continuous has eliminated the quantization error introduced in the earlier discrete PhyloGFN-Bayesian and therefore, our new model is effectively performing on par with the state of the art MrBayes and VBPI-GNN models. In particular, PhyloGFN-Continuous has the smallest standard deviation for the MLL estimation across all datasets, with the only exception being DS7 where PhyloGFN-Continuous closely matches the performance of VBPI-GNN.

Note that PhyloGFN-Continuous has undergone the same training routine that is described in §4.8.4, as is PhyloGFN-Bayesian.

**Table 4.6:** Marginal log-likelihood estimation with different methods on real datasets DS1-DS8. PhyloGFN-C(ontinuous) now outperforms $\phi$-CSMC, GeoPhy and PhyloGFN-B(ayesian) across all datasets and it is effectively performing on par with the state of the arts MrBayes and VBPI-GNN.

| | MCMC | | ML-based / amortized, full tree space | | | |
|---|---|---|---|---|---|---|
| Dataset | MrBayes SS | VBPI-GNN* | $\phi$-CSMC | GeoPhy | PhyloGFN-B | PhyloGFN-C |
| DS1 | $-7108.42_{\pm 0.18}$ | $-7108.41_{\pm 0.14}$ | $-7290.36_{\pm 7.23}$ | $-7111.55_{\pm 0.07}$ | $-7108.95_{\pm 0.06}$ | $-7108.40_{\pm 0.04}$ |
| DS2 | $-26367.57_{\pm 0.48}$ | $-26367.73_{\pm 0.07}$ | $-30568.49_{\pm 31.34}$ | $-26368.44_{\pm 0.13}$ | $-26368.90_{\pm 0.28}$ | $-26367.70_{\pm 0.04}$ |
| DS3 | $-33735.44_{\pm 0.5}$ | $-33735.12_{\pm 0.09}$ | $-33798.06_{\pm 6.62}$ | $-33735.85_{\pm 0.12}$ | $-33735.6_{\pm 0.35}$ | $-33735.11_{\pm 0.02}$ |
| DS4 | $-13330.06_{\pm 0.54}$ | $-13329.94_{\pm 0.19}$ | $-13582.24_{\pm 35.08}$ | $-13337.42_{\pm 1.32}$ | $-13331.83_{\pm 0.19}$ | $-13329.91_{\pm 0.02}$ |
| DS5 | $-8214.51_{\pm 0.28}$ | $-8214.64_{\pm 0.38}$ | $-8367.51_{\pm 8.87}$ | $-8233.89_{\pm 6.63}$ | $-8215.15_{\pm 0.2}$ | $-8214.38_{\pm 0.16}$ |
| DS6 | $-6724.07_{\pm 0.86}$ | $-6724.37_{\pm 0.4}$ | $-7013.83_{\pm 16.99}$ | $-6733.91_{\pm 0.57}$ | $-6730.68_{\pm 0.54}$ | $-6724.17_{\pm 0.10}$ |
| DS7 | $-37332.76_{\pm 2.42}$ | $-37332.04_{\pm 0.12}$ | | $-37350.77_{\pm 11.74}$ | $-37359.96_{\pm 1.14}$ | $-37331.89_{\pm 0.14}$ |
| DS8 | $-8649.88_{\pm 1.75}$ | $-8650.65_{\pm 0.45}$ | $-9209.18_{\pm 18.03}$ | $-8660.48_{\pm 0.78}$ | $-8654.76_{\pm 0.19}$ | $-8650.46_{\pm 0.05}$ |

### 4.8.6 Running time and hardware requirements

PhyloGFN is trained on virtual machines equipped with 10 CPU cores and 10GB RAM for all datasets. We use one V100 GPU for datasets DS1-DS6 and one A100 GPU for DS7-DS8, although the choice of hardware is not essential for running our training algorithms.

For Bayesian inference, the models used for the MLL estimation in Table 4.6 of the paper are trained on a total of 32 million examples, with a training wall time ranging from 3 to 8 days across the eight datasets. However, our algorithm demonstrates the capacity to achieve similar performance levels with significantly reduced training data. The table 4.7 below presents the performance of PhyloGFN with 32 million training examples (**PhyloGFN Full**) and with only $40\%$ of the training trajectories (**PhyloGFN Short**). Each type of experiment is repeated 3 times. Table 4.8 compares running time of the full experiment with the shorter experiment. The tables show that the shorter runs exhibit comparable performance to our full run experiments, and all conclude within 3 days.

We compare the running time of PhyloGFN with VI baselines (VBPI-GNN, Vaiphy, and GeoPhy) using the DS1 dataset. VBPI-GNN and GeoPhy are trained using the same virtual machine configuration as PhyloGFN (10 cores, 10GB ram, 1xV100 GPU). The training setup for both algorithms mirrors the one that yielded the best performance as doc-

umented in their respective papers. As for VaiPhy, we employed the recorded running time from the paper on a machine with 16 CPU cores and 96GB RAM. For PhyloGFN, we calculate the running time of the full training process (PhyloGFN-Full) and four shorter experiments with 40%, 24%, 16% and 8% training examples. The table 4.9 documents both the running time and MLL estimation for each experiment. While our most comprehensive experiment, PhyloGFN Full, takes the longest time to train, our shorter runs – all of which conclude training within a day – show only a marginal degradation in performance. Remarkably, even our shortest run, PhyloGFN - 8%, outperforms both GeoPhy and $\phi$-CSMC, achieving this superior performance with approximately half the training time of GeoPhy.

**Table 4.7:** PhyloGFN-Bayesian MLL estimation on 8 datasets. We repeat both full experiment and short experiment (with 40% training examples) 3 times

| Experiment | PhyloGFN Full | | | PhyloGFN Short (40% Training data) | | |
|---|---|---|---|---|---|---|
| Repeat | 1 | 2 | 3 | 1 | 2 | 3 |
| DS1 | -7108.95 ±0.06 | -7108.97 ±0.05 | -7108.94 ±0.05 | -7108.97 ±0.14 | -7108.94 ±0.22 | -7109.04 ±0.08 |
| DS2 | -26368.9 ±0.28 | -26368.77 ±0.43 | -26368.89 ±0.29 | -26368.9 ±0.39 | -26369.03 ±0.31 | -26368.88 ±0.32 |
| DS3 | -33735.6 ±0.35 | -33735.60 ±0.40 | -33735.68 ±0.64 | -33735.9 ±0.91 | -33735.83 ±0.62 | -33735.76 ±0.75 |
| DS4 | -13331.83 ±0.19 | -13331.80 ±0.31 | -13331.94 ±0.42 | -13332.04 ±0.57 | -13331.87 ±0.31 | -13331.78 ±0.37 |
| DS5 | -8215.15 ±0.2 | -8214.92 ±0.27 | 8214.85 ±0.28 | -8215.38 ±0.27 | -8215.37 ±0.26 | -8215.38 ±0.25 |
| DS6 | -6730.68 ±0.54 | -6730.72 ±0.26 | -6730.89 ±0.22 | -6731.35 ±0.31 | -6731.2 ±0.4 | -6731.1 ±0.38 |
| DS7 | -37359.96 ±1.14 | -37360.59 ±1.62 | -37361.51 ±2.89 | -37362.03 ±5.2 | -37363.43 ±2.2 | -37362.37 ±2.65 |
| DS8 | -8654.76 ±0.19 | -8654.67 ±0.39 | -8654.86 ±0.15 | -8655.8 ±0.95 | -8655.65 ±0.37 | -8654.96 ±0.46 |

**Table 4.8:** PhyloGFN-Bayesian training time

| Dataset | PhyloGFN Full | PhyloGFN Short |
|---|---|---|
| DS1 | 62h40min | 20h40min |
| DS2 | 69h16min | 28h |
| DS3 | 80h20min | 35h40min |
| DS4 | 103h54min | 44h30min |
| DS5 | 127h50min | 51h40min |
| DS6 | 135h10min | 53h10min |
| DS7 | 174h3min | 60h20min |
| DS8 | 190h25min | 61h40min |

**Table 4.9:** Running time of PhyloGFN-Bayesian and VI baseline methods on DS1

| | Running Time | MLL |
|---|---|---|
| VBPI-GNN | 16h10min | -7108.41 $\pm 0.14$ |
| GeoPhy | 12h50min | -7111.55 $\pm 0.07$ |
| $\phi$-CSMC** | $\sim$2h | -7290.36 $\pm 7.23$ |
| PhyloGFN - Full | 62h40min | -7108.97 $\pm 0.05$ |
| PhyloGFN - 40% | 20h40min | -7108.97 $\pm 0.14$ |
| PhyloGFN - 24% | 15h40min | -7109.01 $\pm 0.15$ |
| PhyloGFN - 16% | 10h50min | -7109.15 $\pm 0.23$ |
| PhyloGFN - 8% | 5h10min | -7110.65 $\pm 0.39$ |

### 4.8.7 Ablation study

#### Branch lengths model hyperparameters

PhyloGFN models branch lengths using discrete multinomial distributions. When estimating MLL, the branch length model is transformed into a piecewise-constant continuous form, introducing a small quantization error. Two hyperparameters define the multinomial distribution: bin size and bin number. This analysis investigates the impact of bin size and bin number on MLL estimation.

For the fixed bin size of 0.001, we assess four sets of bin numbers: 50, 100, 150, and 200, and for the fixed bin number of 50, we evaluate three sets of bin sizes: 0.001, 0.002, and 0.004. For each setup, we train a PhyloGFN-Bayesian model on the dataset DS1 using 12.8 millions training examples.

Table 4.10 displays the MLL obtained in each setup. A noticeable decline in MLL estimation occurs as the bin size increases, which is expected due to the increased quantization error. However, MLL estimation also significantly declines as the number of bins increases over 100 under the fixed bin size of 0.001. We conjecture this is because the size of the action pool for sampling the pair of branch lengths increases quadratically by the number of bins. For example at 200 bins, the MLP head that generates branch lengths has 40,000 possible outcomes, leading to increased optimization challenges.

#### Exploration policies

PhyloGFN employs the following to techniques to encourage exploration during training:

- $\epsilon$-greedy annealing: a set of trajectories are generated from the GFlowNet that is being trained, with actions sampled from the GFlowNet policy with probability $1-\epsilon$ and uniformly at random with probability $\epsilon$. The $\epsilon$ rate drops from a pre-defined $\epsilon_{start}$ to near 0 through the course of training.

- Replay Buffer: a replay buffer is used to store the best trees seen to date, and to use them for training, random trajectories are constructed from these high-reward trees using the backward probability $P_B$.
- Temperature annealing: the training of GFlowNet begins at a large temperature $T$ and it is divided in half periodically during training.

To assess the effectiveness of various exploration methods, we train PhyloGFN-Bayesian on DS1 with the following setups:

1. All trajectories are generated strictly on-policy training without any exploration methods (On-policy).
2. All trajectories are generated with epsilon-greedy annealing ($\epsilon$).
3. Half of trajectories are generated with $\epsilon$-greedy annealing, and the other half are constructed from the replay buffer ($\epsilon$ + RP).
4. The same setup as 3, with the addition of temperature annealing. $T$ is initialized at 16 and reduced by half per every 1.5 million training examples until reaching 1 ($\epsilon$ + RP + T Cascading).
5. The same setup as 4, except the temperature drops linearly from 16 to 1 ($\epsilon$ + RP + T Linear).

For each setup, we train a model using 12.8 millions training examples. Table 4.11 displays the MLL estimation for each setup. On-policy training without any additional exploration strategies results in the poorest model performance. Epsilon-greedy annealing significantly enhances performance, and combining all three strategies yields the optimal results. There is no significant difference between cascading and linear temperature drops.

**Table 4.10:** MLL estimation of PhyloGFN-Bayesian on DS1 with different combinations of bin size and bin number to model edge lengths.

| Bin Size | Bin Number | MLL |
|---|---|---|
| 0.001 | 50 | -7108.98 $\pm$0.14 |
| 0.001 | 100 | -7108.96 $\pm$0.18 |
| 0.001 | 150 | -7109.17 $\pm$1.14 |
| 0.001 | 200 | -7109.3 $\pm$7.23 |
| 0.002 | 50 | -7109.95 $\pm$0.29 |
| 0.004 | 50 | -7114.48 $\pm$0.52 |

**Table 4.11:** MLL estimation of PhyloGFN-Bayesian on DS1 with different exploration policies

| Training Policies | MLL |
|---|---|
| On policy | -7307.99 $\pm$0.26 |
| $\epsilon$ | -7109.92 $\pm$0.18 |
| $\epsilon$ + RP | -7109.95 $\pm$0.22 |
| $\epsilon$ + RP + T Cascading | 7108.98 $\pm$0.14 |
| $\epsilon$ + RP + T Linear | 7108.96 $\pm$0.15 |

### 4.8.8 Significance of modeling suboptimal trees well

Several downstream tasks that rely on phylogenetic inference require not only the identification of optimal (maximum likelihood or most parsimonious) trees, but also the proper sampling of suitably weighted suboptimal trees. In evolutionary studies, this includes the computation of branch length support (i.e., the probability that a given subtree is present in the true tree), as well as the estimation of confidence intervals for the timing of specific evolutionary events (Drummond et al., 2006). These are particularly important in the very common situations where the data only weakly define the posterior on tree topologies (e.g. small amount of data per species, or high degree of divergence between sequences). In such cases, because suboptimal trees vastly outnumber optimal trees, they can contribute to a non-negligible extent to the probability mass of specific marginal probabilities. Full modeling of posterior distributions on trees is also critical in studying tumor or virus evolution within a patient (McGranahan et al., 2015; Buendia et al., 2009), e.g., to properly assess the probability of the ordered sequence of driver or passenger mutations that may have led to metastasis or drug resistance (Fisher et al., 2013).

### 4.8.9 Assessing sampling density against unnormalized posterior density

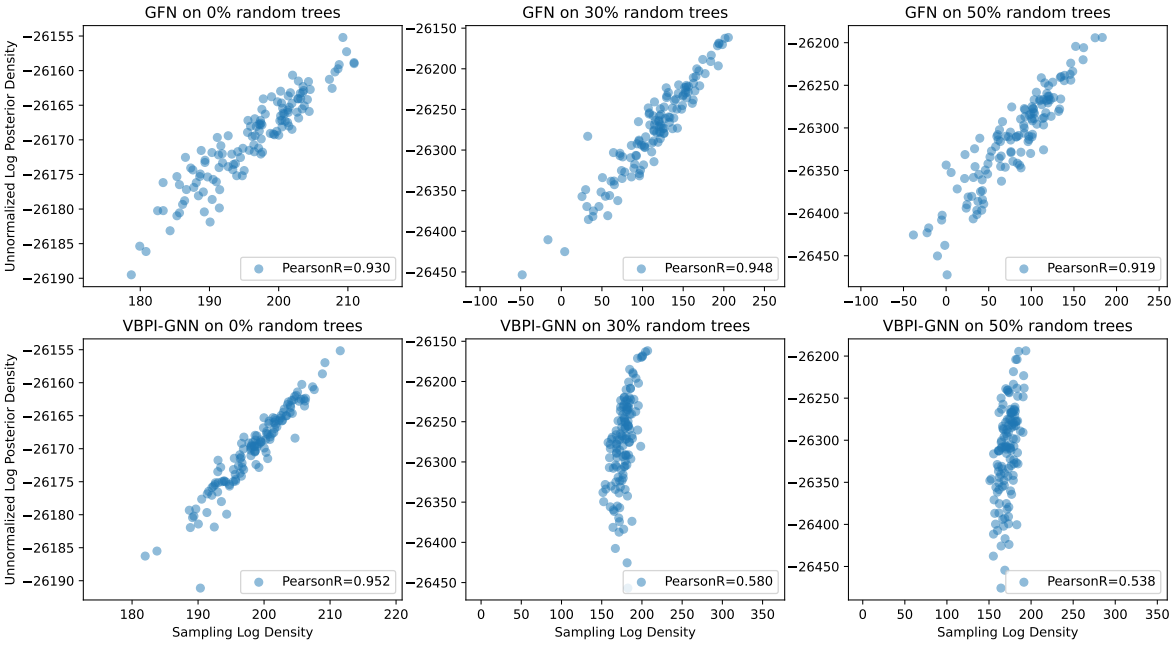
Here, we assess PhyloGFN-Bayesian's ability to model the entire phylogenetic tree space by comparing sampling density against unnormalized posterior over high /medium /low-probability regions of the tree space. To generate trees from medium and low probability regions, we use a trained VBPI-GNN model, the state-of-the-art variational inference algorithm.

We first provide a brief introduction to how VBPI-GNN generates a phylogenetic tree in two phases: (i) it first uses SBN to sample tree topology; (ii) followed by a GNN-based model to sample edge lengths over the tree topology. SBN constructs tree topologies in a sequential manner. Starting from the root node and a set of sequences $\boldsymbol{Y}$ to be labeled at the leaves, the algorithm iteratively generates the child nodes by splitting and distributing the sequence set among the child nodes. The action at each step is thus to choose how to split a set of sequences into two subsets.

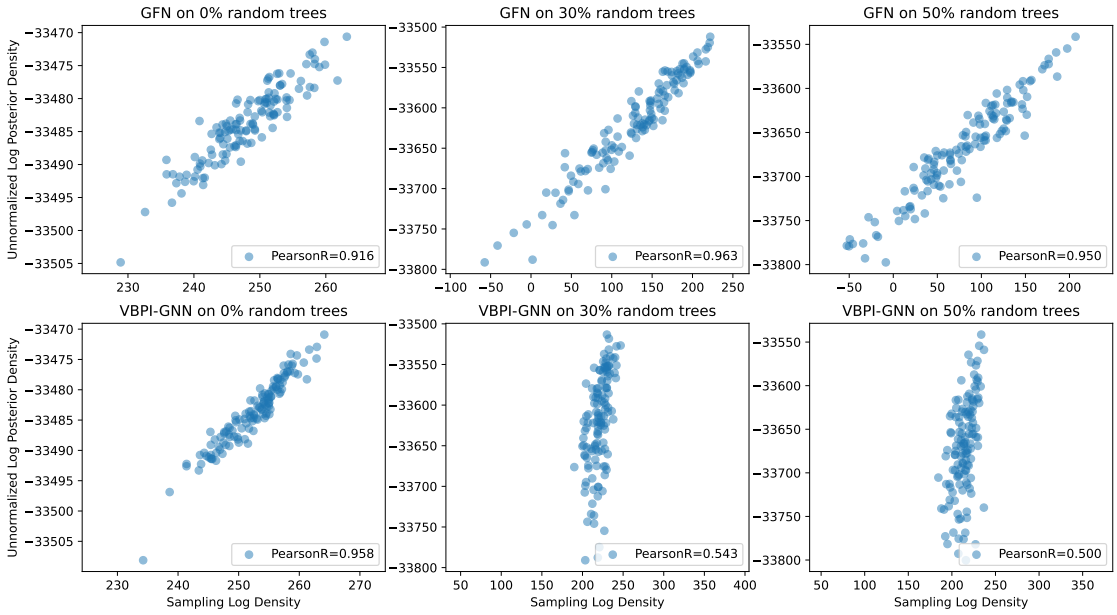
To introduce random perturbations during tree construction, at every step, with probability $\epsilon$, the action is uniformly sampled from the support choices. Given a well-trained VBPI-GNN model, we sample from high/medium/low-probability regions with $0\%$, $30\%$ and $50\%$ probability. We apply PhyloGFN-Bayesian and VBPI-GNN on these sets to compute sampling density and compare with the ground truth unnormalized posterior density computed as $P(\boldsymbol{Y}|z,b)P(z,b)$. Note that VBPI-GNN models the *log-edge length* instead of the edge length. Hence we perform a change of variables when computing both the sampling probability and the unnormalized prior.

Fig. 4.4 shows scatter plots of sampling density versus ground-truth unnormalized posterior density of datasets DS2 to DS8, complementing the result on DS1 shown in Fig. 4.2. We can see that while VBPI-GNN performs well on high-probability regions, our method is better at modeling the target distribution overall. Fig. 4.4f shows that our model performs poorly at modeling the high-probability region for DS7. The result aligns

with our MLL estimation, shown in Table 4.6. Further work is required to investigate the cause of PhyloGFN's poor modeling on DS7.

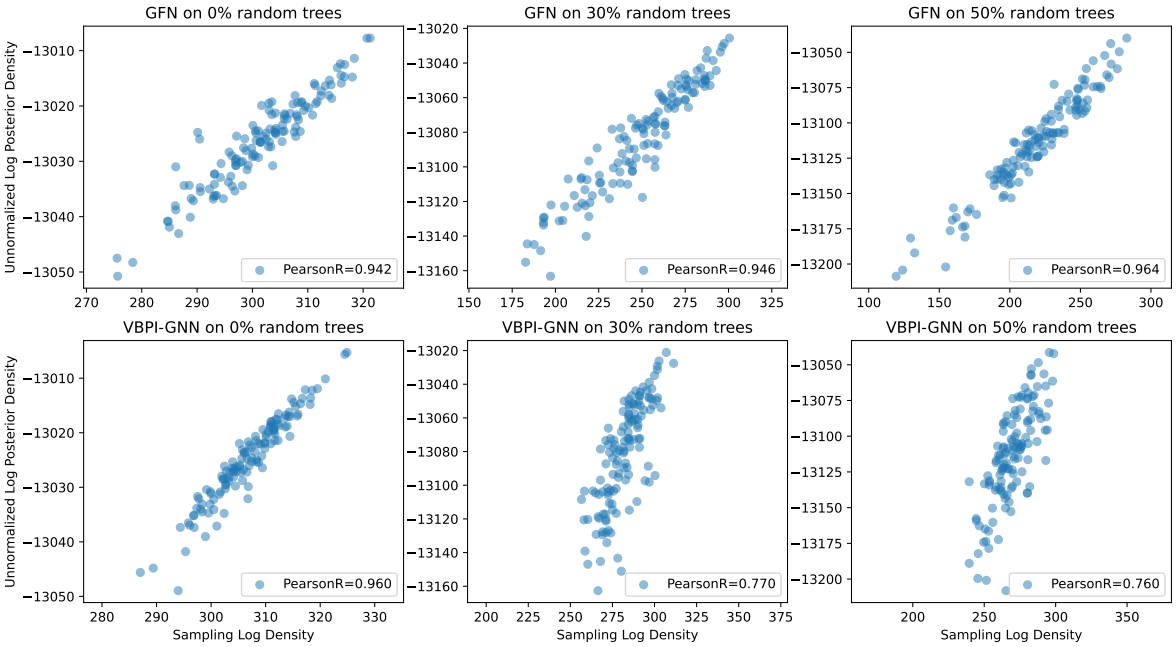



**(a)** DS2

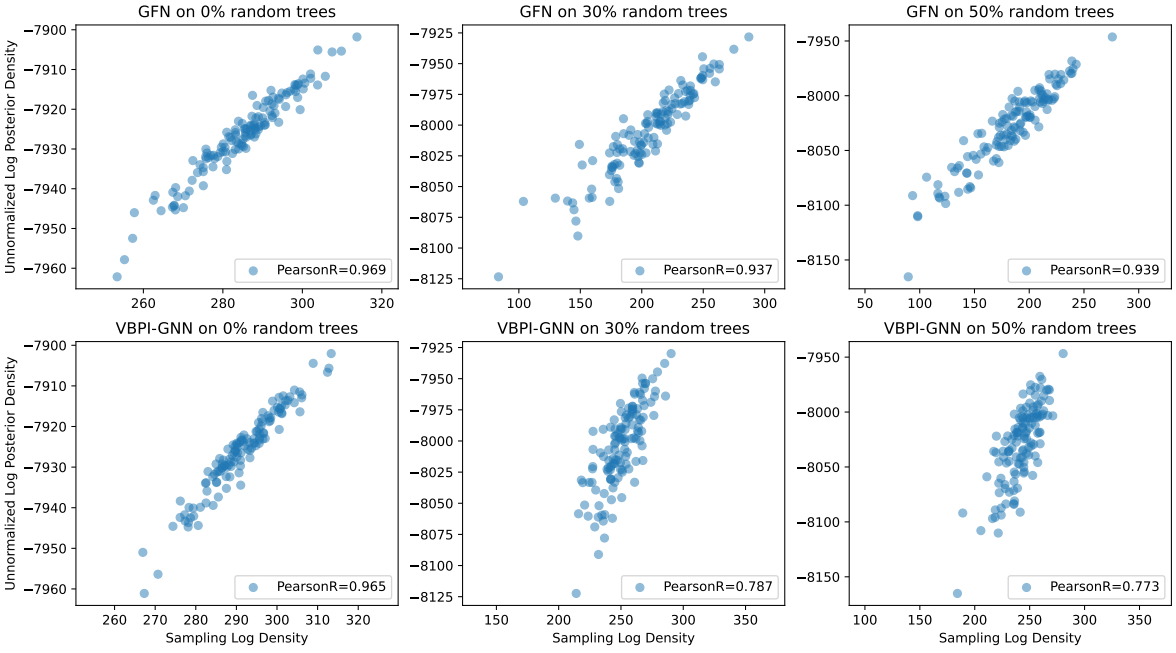



**(b)** DS3

**Figure 4.4:** Sampling log density vs. ground truth unnormalized posterior log density for DS2-DS7

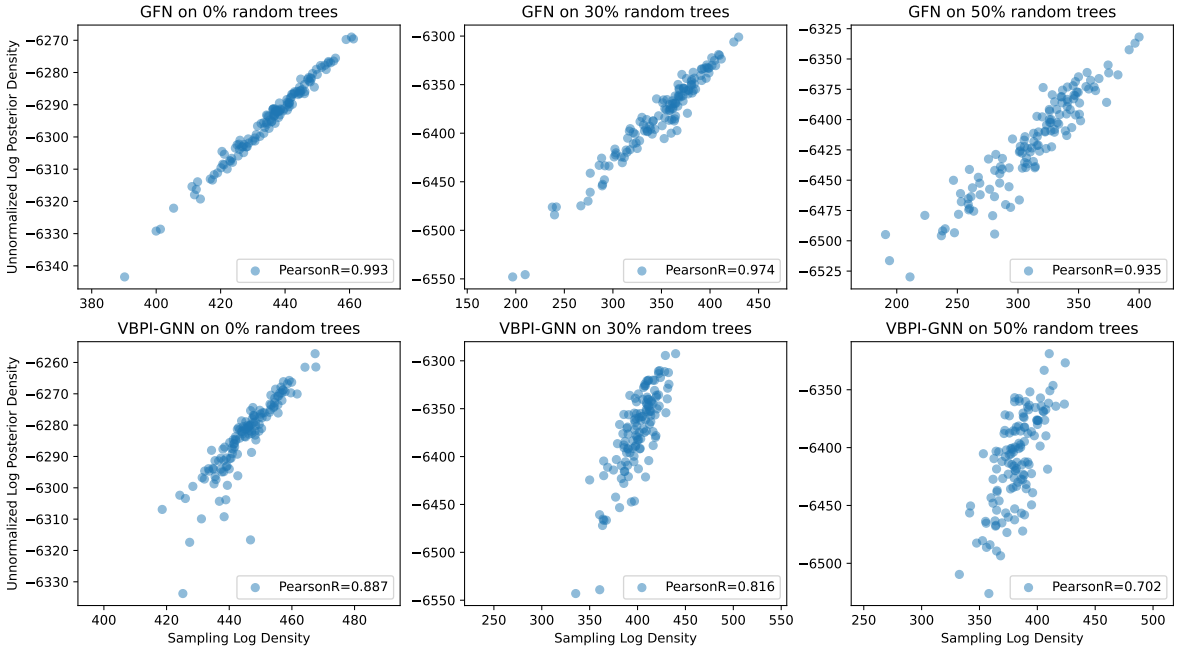



**(c)** DS4

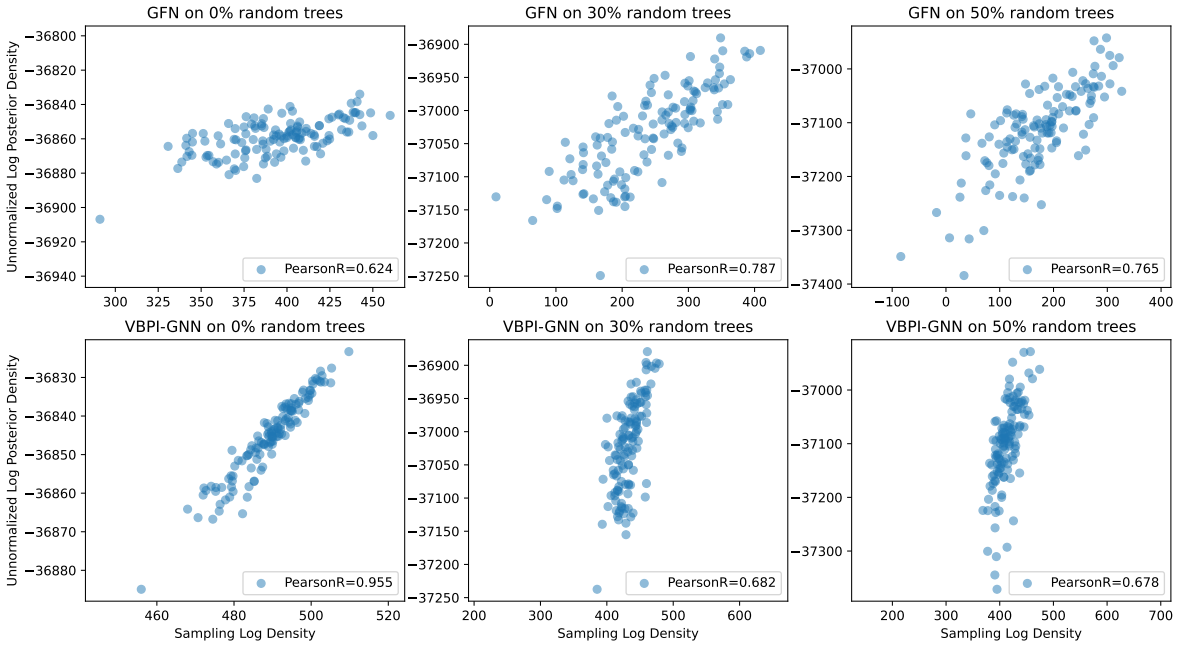



**(d)** DS5

**Figure 4.4:** (cont.)

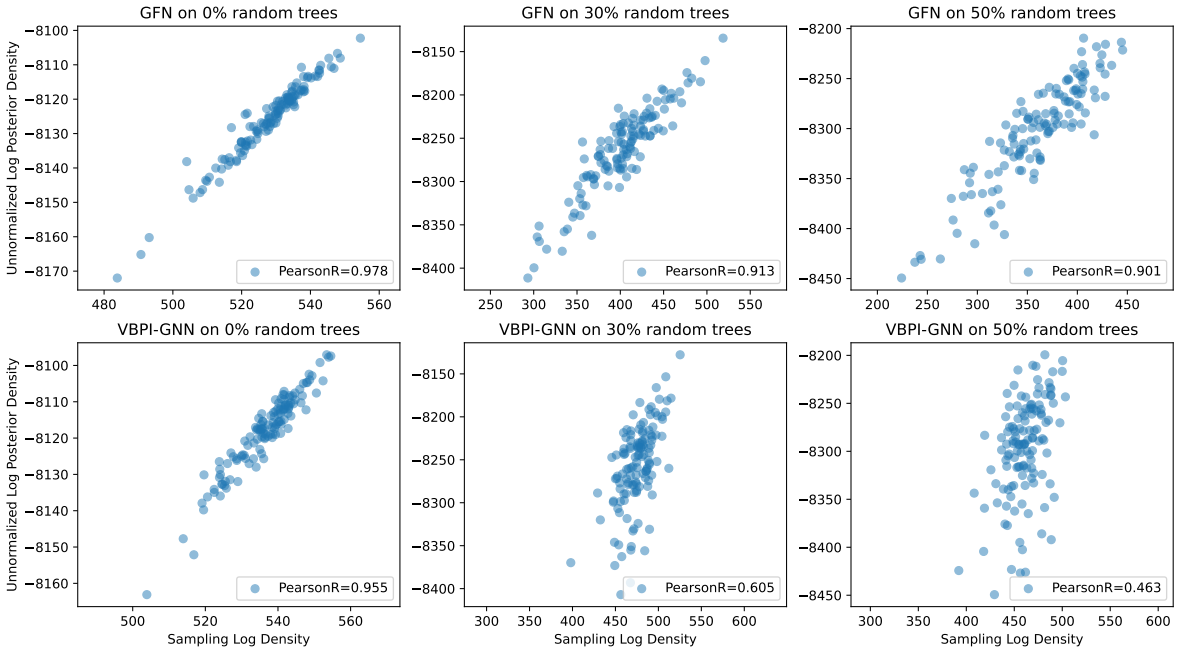



**(e)** DS6




**(f)** DS7

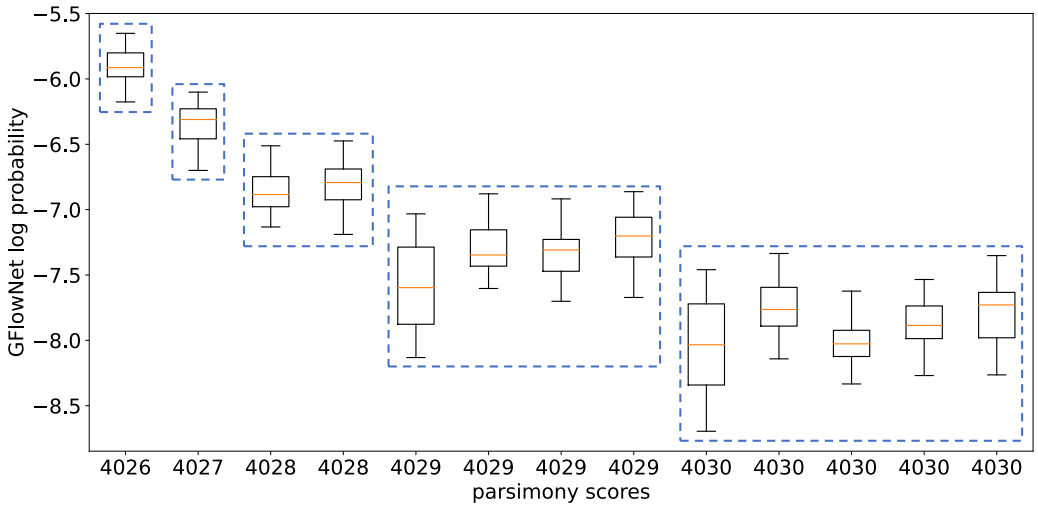
**Figure 4.4:** (cont.)



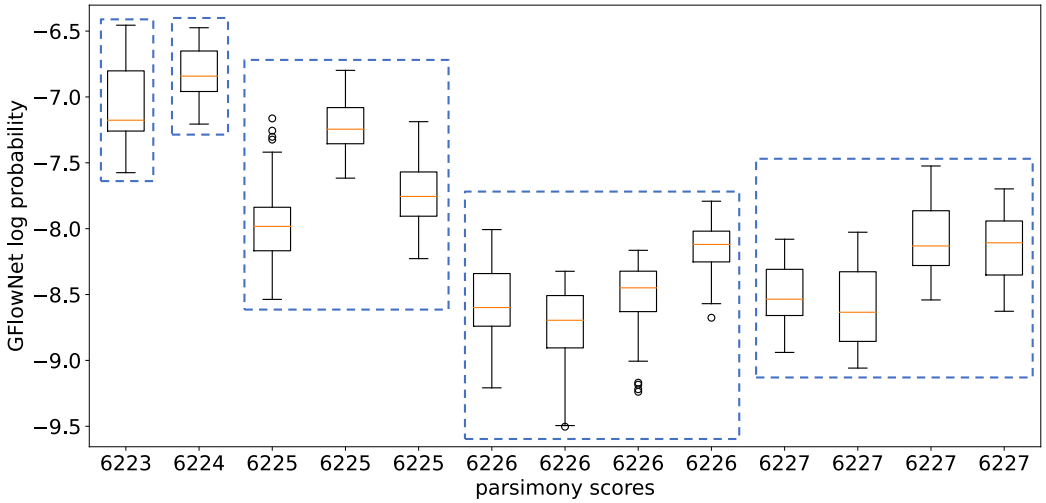

**(a)** DS8

**Figure 4.5:** (cont.)

### 4.8.10 Parsimony analysis results

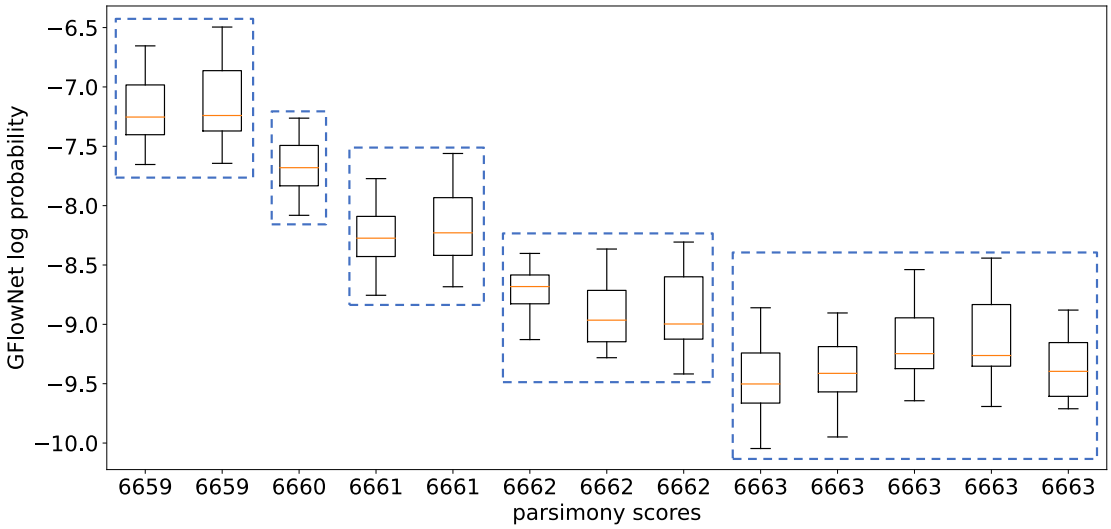



**(a)** A single most-parsimonious tree with a score of 4026 has been identified for DS1.

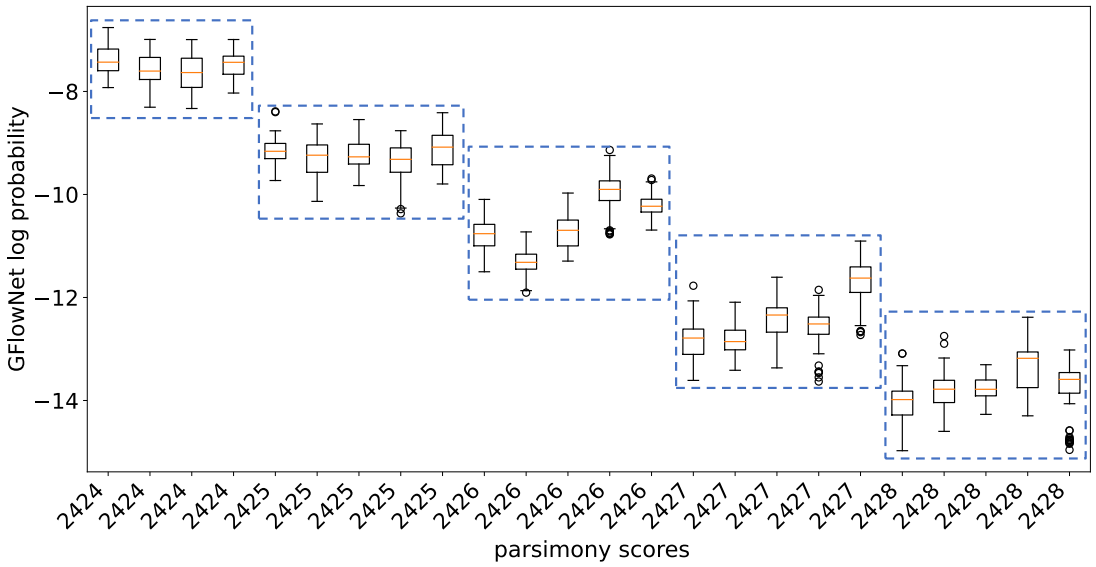



**(b)** A single most-parsimonious tree with a score of 6223 has been identified for DS2.

**Figure 4.6:** For each dataset, the sampling probabilities for the most-parsimonious as well as several suboptimal trees are estimated from our learned PhyloGFNs. Each boxplot represents a single unique unrooted tree and shows a distribution of the log-probabilities over all its $2n - 3$ rooted versions where $n$ denotes the number of species. The dashed bounding box groups the set of equally-parsimonious trees.

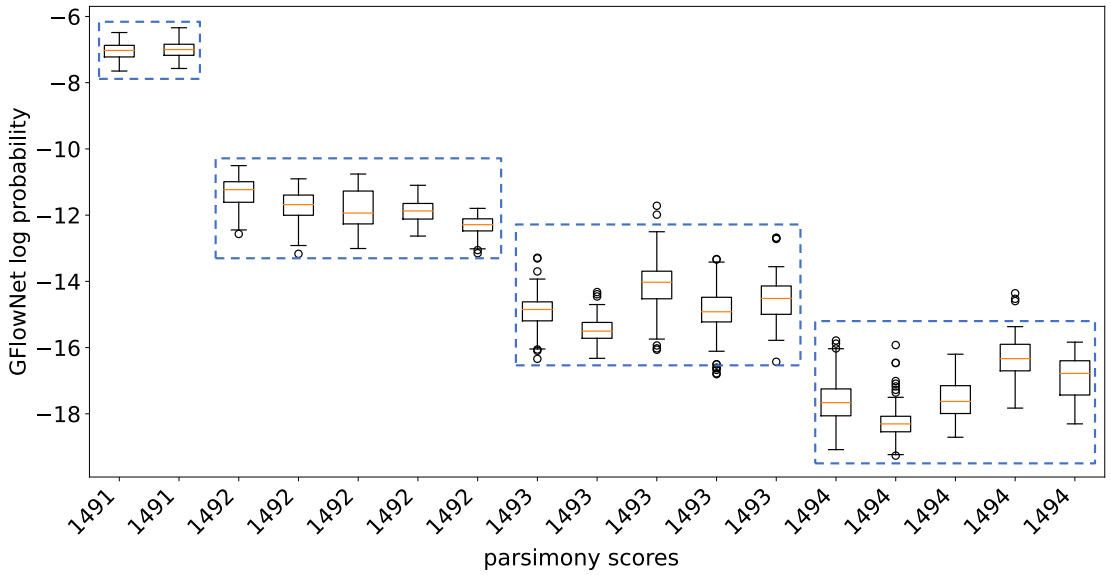



**(c)** Two most-parsimonious trees with a score of 6659 have been identified for DS3.

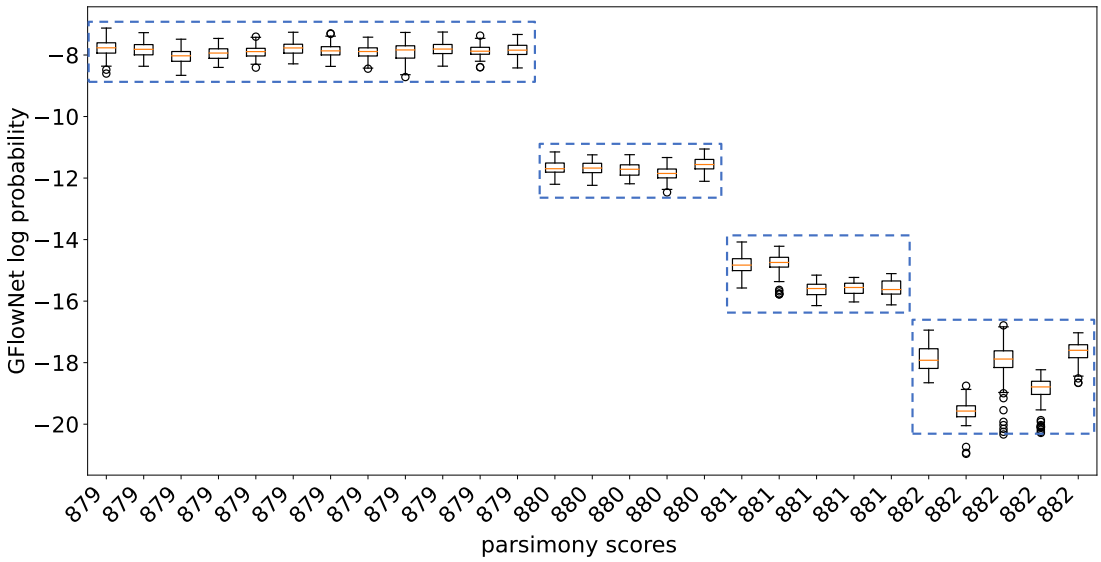



**(d)** Four most-parsimonious trees with a score of 2424 have been identified for DS4.

**Figure 4.6:** (cont.)

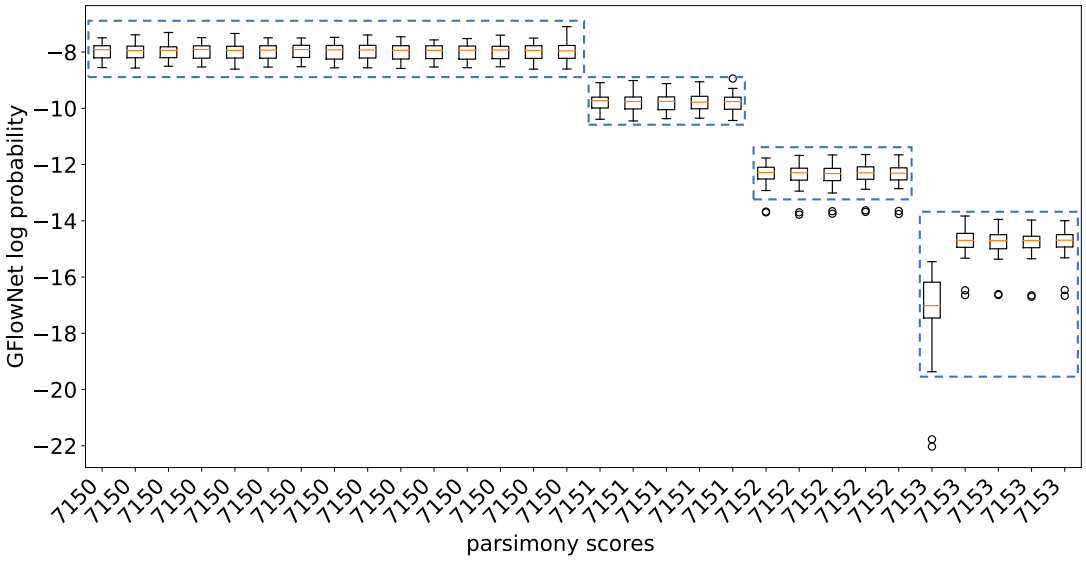



**(e)** Two most-parsimonious trees with a score of 1491 have been identified for DS5.

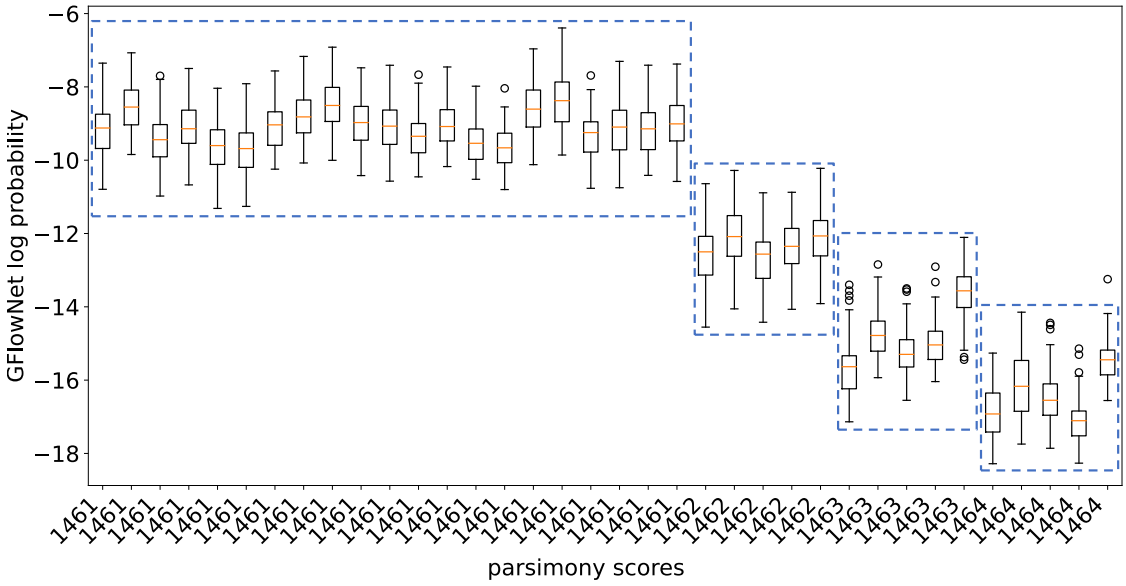



**(f)** 12 most-parsimonious trees with a score of 879 have been identified for DS6.

**Figure 4.6:** (cont.)




**(g)** 15 most-parsimonious trees with a score of 7150 have been identified for DS7.

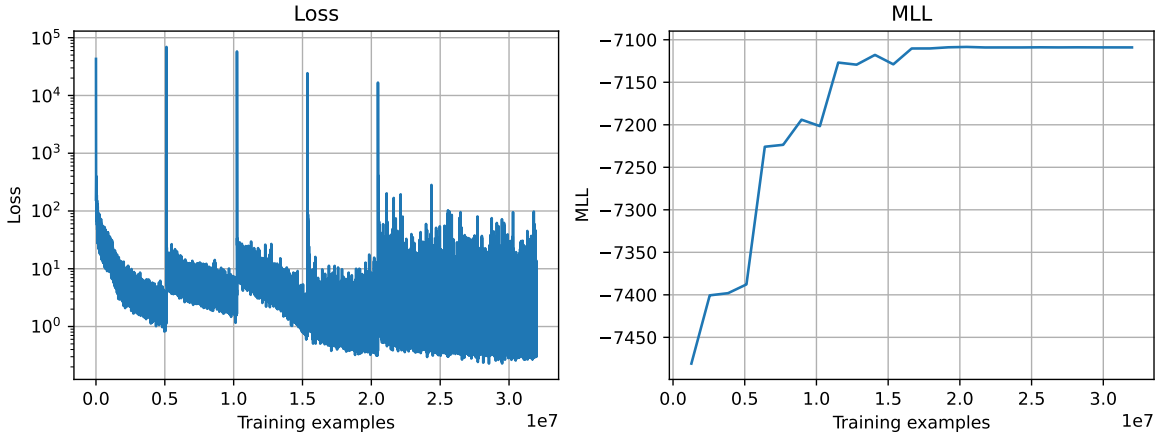



**(h)** 21 most-parsimonious trees with a score of 1461 have been identified for DS8.

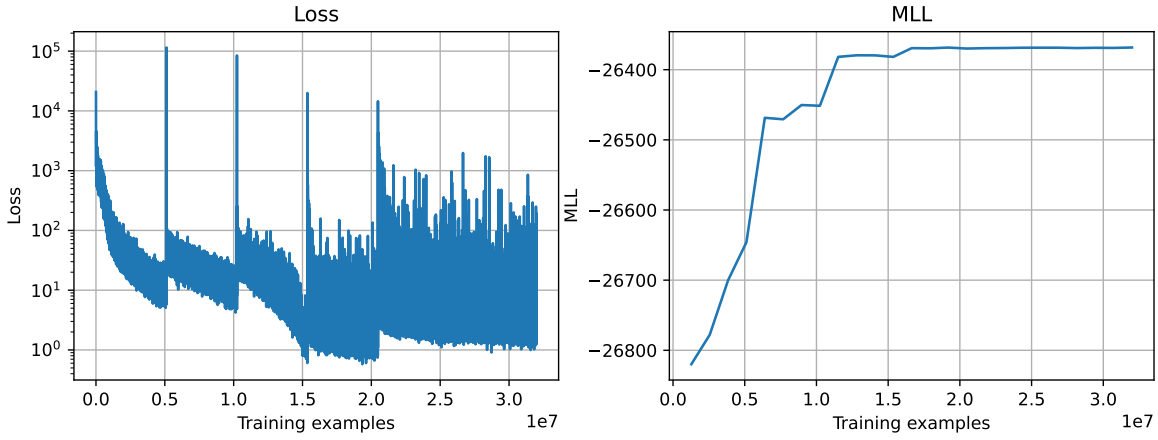
**Figure 4.6:** (cont.)

### 4.8.11 Training curves

The plot in Figure 4.7 illustrates the training curves for PhyloGFN-Bayesian across all eight datasets. In each dataset plot, the left side displays the training loss for every batch of 64 examples, while the right side shows the MLL computed for every 1.28 million training examples.

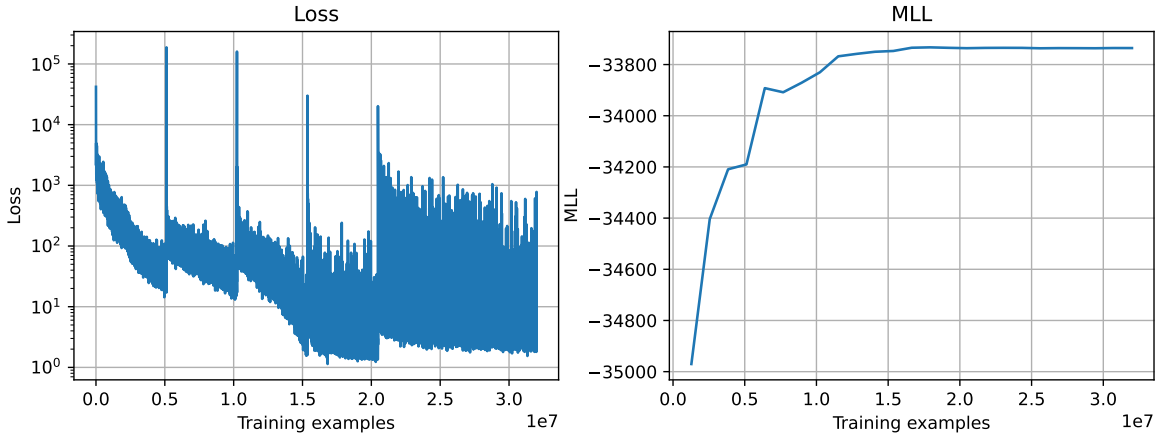



**(a)** DS1

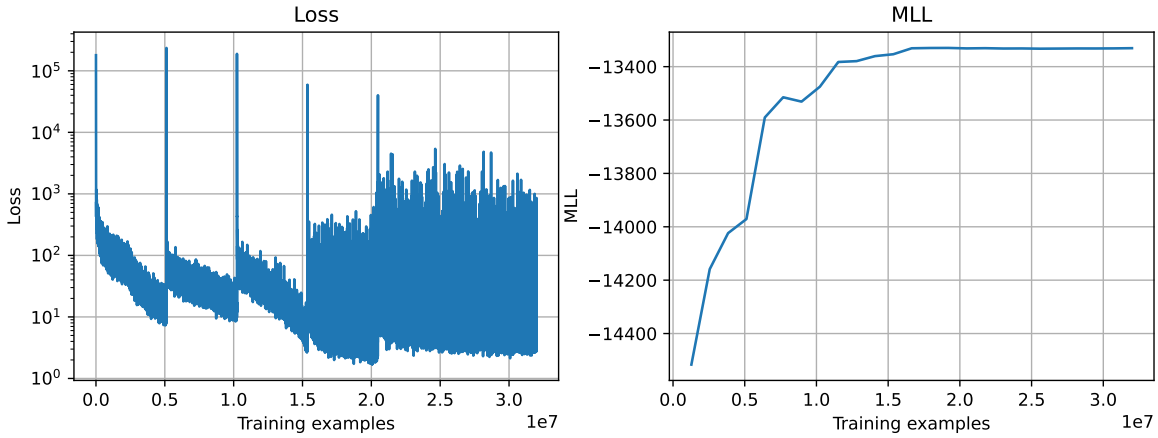



**(b)** DS2

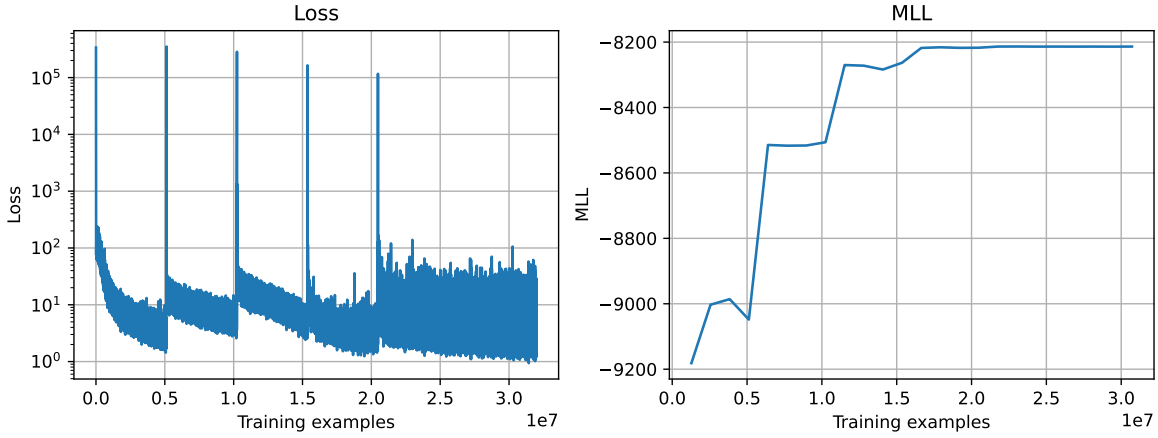
**Figure 4.7:** Training curves for DS1-DS8



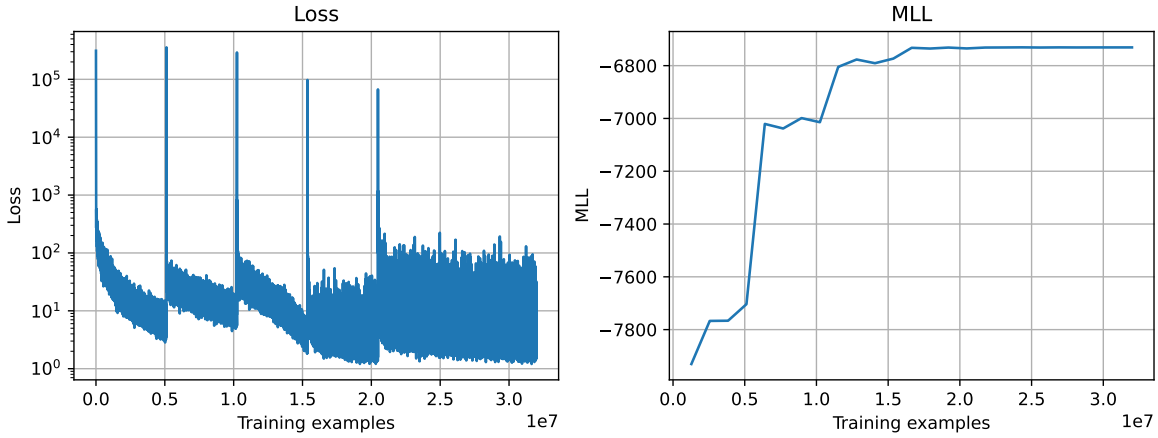

**(c)** DS3



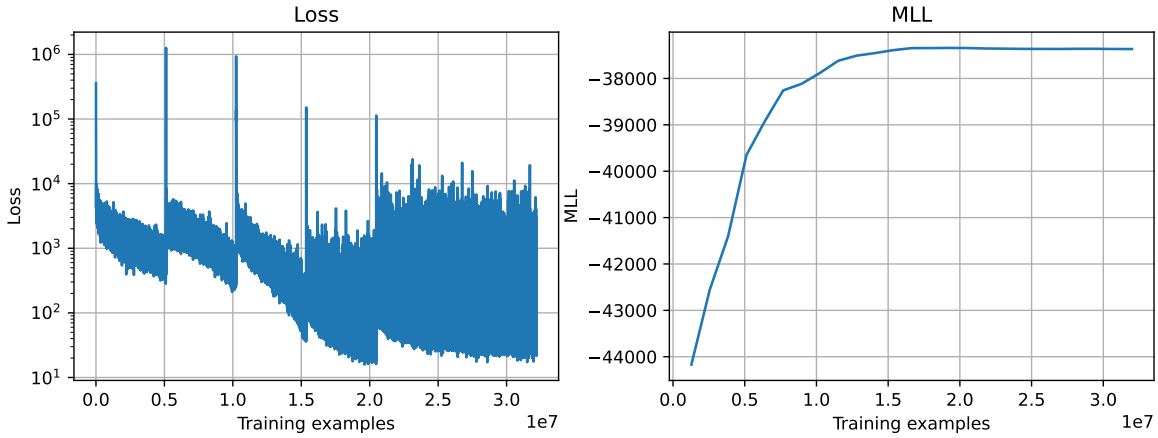

**(d)** DS4



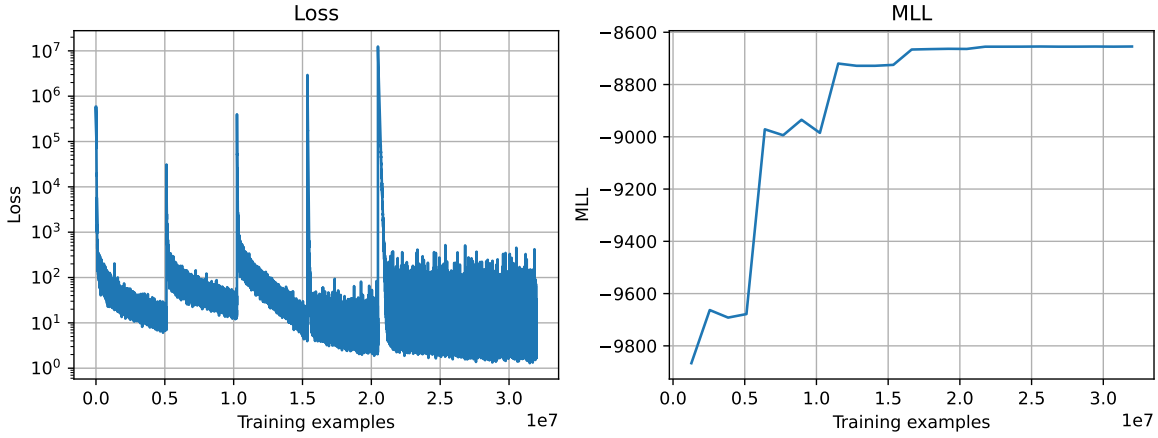

**(e)** DS5




**(f)** DS6




**(g)** DS7




**(h)** DS8

# Chapter 5

# Conclusion

## 5.1 Summary of Contributions

This thesis introduces PhyloGFN, a phylogenetic inference algortihm based on generative flow networks. We focus on two types of phylogenetic inference: parsimony-based analysis and Bayesian probabilistic analysis. We design an acylcic MDP to model the sampling process of phylogenetic trees, utilizing a bottom-up construction approach. Two GFlowNets, *PhyloGFN-Bayesian* and *PhyloGFN-Parsimony*, are designed using this MDP framework, each with its own reward function. We propose two novel representations based on Fitch and Felsenstein's algorithm to represent rooted trees without introducing additional learnable parameters. We then prove that the proposed representations encapsulate all necessary information for an optimal GFlowNet sampling policy. We put PhyloGFN to the test using 8 real benchmark datasets. For parsimony analysis, we demonstrate that PhyloGFN-Parsimony can successfully retrieve all optimal solutions across all datasets. Additionally, the model can sample tree topologies inversely proportional to their parsimony scores. In Bayesian analysis, PhyloGFN performs competitively with state-of-the-art MCMC-based and VI-based methods in terms of estimating MLL. Comparing our proposed method with VBPI-GNN (Zhang, 2023), a non-MCMC algorithm

that achieves the highest MLL estimation, our approach substantially outperforms in its ability to estimate the posterior probability of sub-optimal trees.

## 5.2 Impact of the work

PhyloGFN is the first GFlowNet-based algorithm for phylogenetic inference. Prior to our work, the predominant approach for Bayesian phylogenetic inference relied on MCMC-based algorithms. As detailed in Section 3.2.1, MCMC-based methods suffer from various limitations, including slow convergence when dealing with complex, multi-modal problems. In recent years, the scientific community has increasingly turned to variational inference-based algorithms to address these shortcomings. Among these algorithms, the current state-of-the-art VI method, VBPI-GNN, focuses on modeling the posterior distribution within a limited subset of the phylogenetic tree space and fails to adequately represent sub-optimal trees. There exist VI algorithms (VaiPhy, GeoPhy) that model the entire phylogenetic trees space. However, they generally underperform in terms of estimating MLL when compared to the state-of-the-art. PhyloGFN demonstrates that GFlowNet-based algorithm is an alternative approach for bayesian phylogenetic inference. The amortized GFlowNet sampler learns the posterior distribution over the entire phylogenetic trees space. It improves MLL estimation significantly for 5 out of 8 real benchmark datasets comparing with VI algorithms that also model the entire space. The analysis of sampling probability vs target un-normalized posterior reveals that, while our algorithm may not achieve the state-of-the-art MLL estimation for some datasets, the high correlation between sampling probability vs target un-normalized posterior shows that our algorithm estimates well the posterior distribution for 7 out of 8 datasets. Moreover, comparing with VBPI-GNN, our algorithm is significantly superior at modeling sub-optimal trees. This highlights the GFlowNet-based method's ability to comprehensively represent the entire sample space, as opposed to a narrower focus on high-likelihood trees.

In PhyloGFN, we employ a bottom-up approach to construct phylogenetic trees by iteratively merging pairs of trees. This approach has two main advantages: 1. at each step, the number of actions is bounded by $O(N^2)$. 2. The conditional likelihood for Bayesian analysis or the optimal internal sequence assignments for parsimony analysis can be computed directly for each new node introduced during the tree construction. When a complete tree is formed, we can readily obtain the parsimony score or un-normalized posterior by considering the root node's score. In contrast, when using alternative construction procedures proposed in Vaiphy, GeoPhy, or VBPI-GNN, one must re-traverse the entire tree each time a tree is constructed, significantly slowing down the training process. The bottom-up procedure also serves as the basis for designing greedy algorithms for multiple related biological problems. For example, in the context of multiple sequence alignment, the progressive alignment construction algorithm iteratively combines pairs of alignments until a complete alignment is achieved for all sequences (Feng & Doolittle, 1987). The construction MDP used in PhyloGFN can be extended to develop GFlowNet-based solutions for these related problems.

The bottom up construction process enables us to design two rooted tree representations based on Fitch and Felsenstein's algorithms. While we prove that they provide sufficient information for an optimal GFlowNet sampling policy, we think they actually provide sufficient information for *any* optimal sampling policy with the bottom-up construction MDP. Consequently, they hold the potential to be harnessed by other phylogenetic inference algorithms utilizing the bottom-up MDP framework. In section 4.5.4, we show that when estimating MLL with GFlowNets, we cannot use the importance weighted bound that is commonly used to evaluate VI algorithms. We derive lower bound with formation 4.5 and prove its consistency. The formulation is not restricted to phylognetic inference, it can be used to calculate MLL lower bound for any GFlowNet model that learns posterior distributions.

## 5.3 Limitations and Future works

The most significant drawback of PhyloGFN lies in its MLL estimation, which still lags behind the state-of-the-art. This discrepancy can be attributed to two primary factors: 1. error caused by modeling edge lengths with discrete distributions, 2. error caused by the model not optimally trained. To address the first issue, we can enhance our algorithm by adopting continuous distributions to model edge lengths. VBPI-GNN models branch lengths with lognormal distribution. We can adapt this modeling approach to devise a continuous GFlowNet, in contrast to the discrete GFlowNet extensively discussed in this thesis. The theoretical framework for the continuous GFlowNet is studied in detail in (Lahlou et al., 2023). Additional works are required to design appropriate training mechanisms that can effectively train the continuous GFlowNet. In tackling the second issue, a comprehensive review of the entire training pipeline is necessary to identify less efficient components. For instance, we may experiment with recently proposed loss functions, such as SubTB (Madan et al., 2023) or FL loss (Pan et al., 2023), to improve training efficiency.

One additional concern pertains to the substantial amount of RAM required for training PhyloGFN. This RAM usage can be significantly minimized to near-zero by offloading the majority of computations onto the GPU. Many operations related to the MDP environment are amenable to GPU acceleration: state initialization can be executed as a GPU tensor initialization, and state transition can be realized by merging the corresponding two tensors' columns within the GPU. In the case of the Fitch algorithm, the merging operation involves a boolean operation on two tensors. Meanwhile, for Felsenstein's algorithm, the merging operation can be efficiently implemented using tensor calculation packages, such as Einops (Rogozhnikov, 2022).

Additionally, certain architectural choices in PhyloGFN could be optimized. For instance, the tree feature, consisting of $M$ sites, is currently flattened into a vector of size $M|\Sigma|$ when passed as input to the GFlowNet model. A more suitable approach would

be to utilize a model component designed for sequential data input, such as an RNN or transformer, considering that the feature is actually a sequence of $M$ tokens of size $\Sigma$.

Furthermore, the current model computes probabilistic logits for all $\binom{N}{2}$ pairs of trees in a single step, which becomes impractical for a large number of trees. An alternative solution involves a two-step process: 1. Choosing a candidate tree and 2. Selecting a tree from the remaining ones to join with the candidate. This approach limits the action space at each step to $O(N)$ instead of $O(N^2)$

Another potential improvement for PhyloGFN is to design a more efficient tree construction process. For example, as described by Durbin et al. (1998), a top-down tree generation process involves fixing the order of sequences and initializing the process with a tree formed by the first 3 sequences. Subsequently, at each step, the next sequence in order is selected, and it is joined with one of the edges in the current constructed tree. The advantage of this process is that, for a problem with $N$ sequences, a tree can be constructed in exactly $N - 3$ steps. At each step, the number of choices is limited to the number of edges of the current tree, which is bounded by $O(N)$.

Finally, it's worth noting that this work makes several assumptions about the evolution model, such as a uniform prior over tree topologies, a decomposed prior $P(z, b) = P(z)P(b)$, an exponential ($\lambda = 10$) prior over branch lengths, and the Jukes-Cantor substitution model. The tree representations and GFlowNet rewards are closely aligned with these assumptions. Future research should aim to make PhyloGFN adaptable to other evolution models.

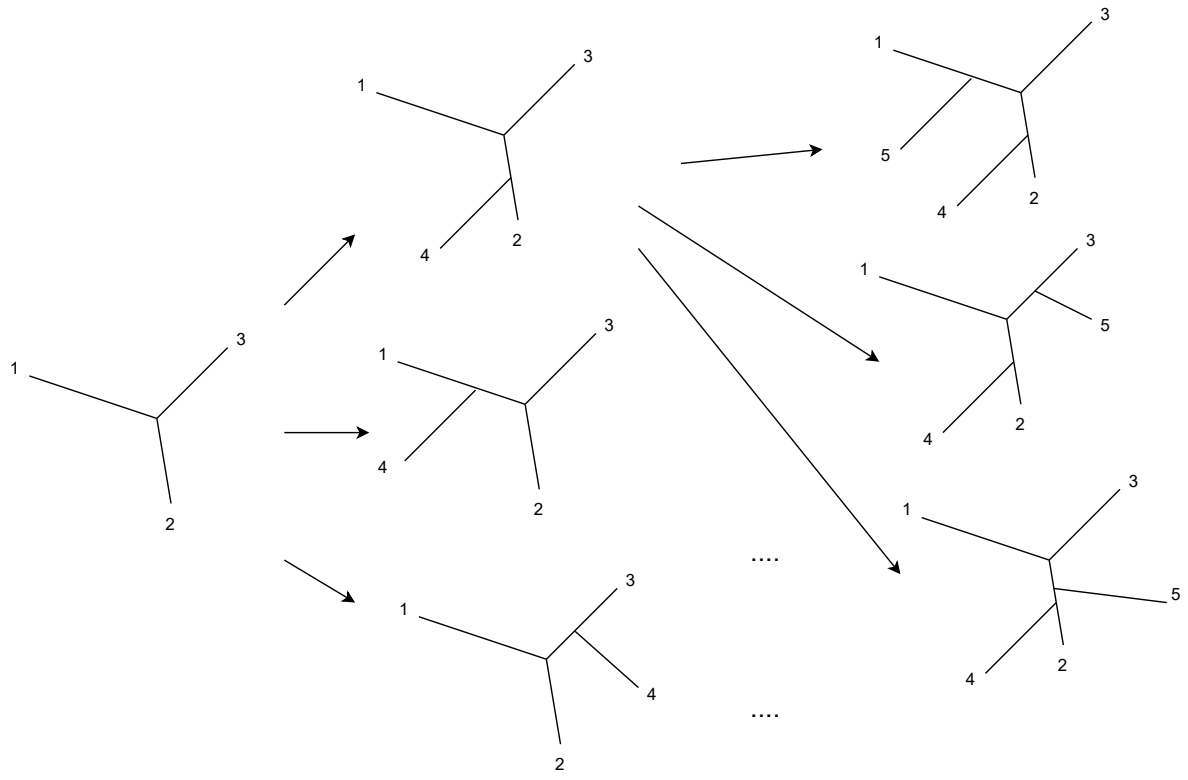



**Figure 5.1:** A tree construction procedure described in Durbin et al. (1998). The advantage of this procedure is that at each step, the action space is $O(N)$ and each tree is uniquely defined by one construction trajectory.

# Bibliography

Rustam I Aminov and Roderick I Mackie. Evolution and ecology of antibiotic resistance genes. *FEMS microbiology letters*, 271(2):147–161, 2007.

Marc G. Bellemare, Will Dabney, and Remi Munos. A distributional perspective on reinforcement learning. *International Conference on Machine Learning (ICML)*, 2017.

Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow network based generative models for non-iterative diverse candidate generation. *Neural Information Processing Systems (NeurIPS)*, 2021.

Yoshua Bengio, Salem Lahlou, Tristan Deleu, Edward J Hu, Mo Tiwari, and Emmanuel Bengio. GFlowNet foundations. *Journal of Machine Learning Research*, (24):1–76, 2023.

Mathieu Blanchette, W James Kent, Cathy Riemer, Laura Elnitski, Arian FA Smit, Krishna M Roskin, Robert Baertsch, Kate Rosenbloom, Hiram Clawson, Eric D Green, et al. Aligning multiple genomic sequences with the threaded blockset aligner. *Genome research*, 14(4):708–715, 2004.

Patricia Buendia, Brice Cadwallader, and Victor DeGruttola. A phylogenetic and Markov model approach for the reconstruction of mutational pathways of drug resistance. *Bioinformatics*, 25(19):2522–2529, 08 2009.

Yuri Burda, Roger Baker Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *International Conference on Learning Representations (ICLR)*, 2016.

Gershon Celniker, Guy Nimrod, Haim Ashkenazy, Fabian Glaser, Eric Martz, Itay Mayrose, Tal Pupko, and Nir Ben-Tal. Consurf: using evolutionary data to raise testable hypotheses about protein function. *Israel Journal of Chemistry*, 53(3-4):199–206, 2013.

Benny Chor and Tamir Tuller. Maximum likelihood of evolutionary trees is hard. In *Annual International Conference on Research in Computational Molecular Biology*, pp. 296–310. Springer, 2005.

Flaviu Cipcigan, Jonathan Booth, Rodrigo Neumann Barros Ferreira, Carine Ribeiro dos Santo, and Mathias Steiner. Discovery of novel reticular materials for carbon dioxide capture using gflownets. *arXiv preprint arXiv:2310.07671*, 2023.

William HE Day. Computational complexity of inferring phylogenies from dissimilarity matrices. *Bulletin of mathematical biology*, 49(4):461–467, 1987.

Tristan Deleu, António Góis, Chris Emezue, Mansi Rankawat, Simon Lacoste-Julien, Stefan Bauer, and Yoshua Bengio. Bayesian structure learning with generative flow networks. *Uncertainty in Artificial Intelligence (UAI)*, 2022.

Tristan Deleu, Mizu Nishikawa-Toomey, Jithendaraa Subramanian, Nikolay Malkin, Laurent Charlin, and Yoshua Bengio. Joint Bayesian inference of graphical structure and parameters with a single generative flow network. *arXiv preprint arXiv:2305.19366*, 2023.

Amrit Dhar and Vladimir N Minin. Maximum likelihood methods for phylogenetic inference. 2015.

Erika Dort, Elliot Layne, Nicolas Feau, Alexander Butyaev, Bernard Henrissat, Francis Martin, Sajeet Haridas, Asaf Salamov, Igor Grigoriev, Mathieu Blanchette, and Hamelin Richard. Large-scale genomic analyses with machine learning uncover predictive patterns associated with fungal phytopathogenic lifestyles and traits. *Scientific Reports*, 13 (1), 2023.

Alexei J Drummond, Simon Y. W Ho, Matthew J Phillips, and Andrew Rambaut. Relaxed phylogenetics and dating with confidence. *PLOS Biology*, 4(5):null, 03 2006. doi: 10.1371/journal.pbio.0040088.

Richard Durbin, Sean R Eddy, Anders Krogh, and Graeme Mitchison. *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge university press, 1998.

J. Felsenstein. *Inferring Phylogenies*. Sinauer, 2003. ISBN 9780878931774. URL `https://books.google.ca/books?id=GI6PQgAACAAJ`.

Joseph Felsenstein. Maximum likelihood and minimum-steps methods for estimating evolutionary trees from data on discrete characters. *Systematic Biology*, 22(3):240–249, 1973.

Da-Fei Feng and Russell F Doolittle. Progressive sequence alignment as a prerequisitetto correct phylogenetic trees. *Journal of molecular evolution*, 25:351–360, 1987.

R. Fisher, L. Pusztai, and C. Swanton. Cancer heterogeneity: implications for targeted therapeutics. *British Journal of Cancer*, 108(3):479–485, Feb 2013.

Walter M Fitch. Toward defining the course of evolution: minimum change for a specific tree topology. *Systematic Biology*, 20(4):406–416, 1971.

Walter M Fitch and Etan Markowitz. An improved method for determining codon variability in a gene and its application to the rate of fixation of mutations in evolution. *Biochemical genetics*, 4:579–593, 1970.

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

Richard C Hamelin, Guillaume J Bilodeau, Renate Heinzelmann, Kelly Hrywkiw, Arnaud Capron, Erika Dort, Angela L Dale, Emilie Giroux, Stacey Kus, Nick C Carleson, et al.

Genomic biosurveillance detects a sexual hybrid in the sudden oak death pathogen. *Communications Biology*, 5(1):477, 2022.

S B Hedges, K D Moberg, and L R Maxson. Tetrapod phylogeny inferred from 18S and 28S ribosomal RNA sequences and a review of the evidence for amniote relationships. *Molecular Biology and Evolution*, 7(6):607–633, 11 1990.

Daniel A Henk, Alex Weir, and Meredith Blackwell. Laboulbeniopsis termitarius, an ectoparasite of termites newly recognized as a member of the laboulbeniomycetes. *Mycologia*, 95(4):561–564, July 2003.

Diep Thi Hoang, Olga Chernomor, Arndt Von Haeseler, Bui Quang Minh, and Le Sy Vinh. Ufboot2: improving the ultrafast bootstrap approximation. *Molecular biology and evolution*, 35(2):518–522, 2018.

Edward J Hu, Nikolay Malkin, Moksh Jain, Katie Everett, Alexandros Graikos, and Yoshua Bengio. GFlowNet-EM for learning compositional latent variable models. *International Conference on Machine Learning (ICML)*, 2023.

Sebastian Höhna, Michael J. Landis, Tracy A. Heath, Bastien Boussau, Nicolas Lartillot, Brian R. Moore, John P. Huelsenbeck, and Fredrik Ronquist. RevBayes: Bayesian Phylogenetic Inference Using Graphical Models and an Interactive Model-Specification Language. *Systematic Biology*, 65(4):726–736, 05 2016. ISSN 1063-5157. doi: 10.1093/sysbio/syw021. URL `https://doi.org/10.1093/sysbio/syw021`.

Moksh Jain, Emmanuel Bengio, Alex Hernandez-Garcia, Jarrid Rector-Brooks, Bonaventure F.P. Dossou, Chanakya Ekbote, Jie Fu, Tianyu Zhang, Micheal Kilgour, Dinghuai Zhang, Lena Simine, Payel Das, and Yoshua Bengio. Biological sequence design with GFlowNets. *International Conference on Machine Learning (ICML)*, 2022.

Moksh Jain, Tristan Deleu, Jason Hartford, Cheng-Hao Liu, Alex Hernandez-Garcia, and Yoshua Bengio. Gflownets for ai-driven scientific discovery. *Digital Discovery*, 2(3): 557–577, 2023.

Thomas H Jukes, Charles R Cantor, et al. Evolution of protein molecules. *Mammalian protein metabolism*, 3:21–132, 1969.

Motoo Kimura. A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *Journal of molecular evolution*, 16: 111–120, 1980.

Hazal Koptagel, Oskar Kviman, Harald Melin, Negar Safinianaini, and Jens Lagergren. VaiPhy: a variational inference based algorithm for phylogeny. *Neural Information Processing Systems (NeurIPS)*, 2022.

Salem Lahlou, Tristan Deleu, Pablo Lemos, Dinghuai Zhang, Alexandra Volokhova, Alex Hernández-García, Léna Néhale Ezzine, Yoshua Bengio, and Nikolay Malkin. A theory of continuous generative flow networks. *International Conference on Machine Learning (ICML)*, 2023.

David Lähnemann, Johannes Köster, Ewa Szczurek, Davis J McCarthy, Stephanie C Hicks, Mark D Robinson, Catalina A Vallejos, Kieran R Campbell, Niko Beerenwinkel, Ahmed Mahfouz, et al. Eleven grand challenges in single-cell data science. *Genome biology*, 21 (1):1–35, 2020.

Clemens Lakner, Paul Van Der Mark, John P Huelsenbeck, Bret Larget, and Fredrik Ronquist. Efficiency of Markov chain Monte Carlo tree proposals in Bayesian phylogenetics. *Systematic biology*, 57(1):86–103, 2008.

Elliot Layne, Erika N Dort, Richard Hamelin, Yue Li, and Mathieu Blanchette. Supervised learning on phylogenetically distributed data. *Bioinformatics*, 36(Supplement_2):i895–i902, 2020.

John DC Little, Katta G Murty, Dura W Sweeney, and Caroline Karel. An algorithm for the traveling salesman problem. *Operations research*, 11(6):972–989, 1963.

Jian Ma, Louxin Zhang, Bernard B Suh, Brian J Raney, Richard C Burhans, W James Kent, Mathieu Blanchette, David Haussler, and Webb Miller. Reconstructing contiguous regions of an ancestral genome. *Genome research*, 16(12):1557–1565, 2006.

Kanika Madan, Jarrid Rector-Brooks, Maksym Korablyov, Emmanuel Bengio, Moksh Jain, Andrei Cristian Nica, Tom Bosc, Yoshua Bengio, and Nikolay Malkin. Learning gflownets from partial episodes for improved convergence and stability. In *International Conference on Machine Learning*, pp. 23467–23483. PMLR, 2023.

Nikolay Malkin, Moksh Jain, Emmanuel Bengio, Chen Sun, and Yoshua Bengio. Trajectory balance: Improved credit assignment in GFlowNets. *Neural Information Processing Systems (NeurIPS)*, 2022.

Nikolay Malkin, Salem Lahlou, Tristan Deleu, Xu Ji, Edward Hu, Katie Everett, Dinghuai Zhang, and Yoshua Bengio. GFlowNets and variational inference. *International Conference on Learning Representations (ICLR)*, 2023.

Bob Mau, Michael A Newton, and Bret Larget. Bayesian phylogenetic inference via markov chain monte carlo methods. *Biometrics*, 55(1):1–12, 1999.

Nicholas McGranahan, Francesco Favero, Elza C. de Bruin, Nicolai Juul Birkbak, Zoltan Szallasi, and Charles Swanton. Clonal status of actionable driver events and the timing of mutational processes in cancer evolution. *Science Translational Medicine*, 7(283): 283ra54–283ra54, 2015.

Takahiro Mimori and Michiaki Hamada. Geophy: Differentiable phylogenetic inference via geometric gradients of tree topologies. *arXiv preprint arXiv:2307.03675*, 2023.

Bui Quang Minh, Heiko A Schmidt, Olga Chernomor, Dominik Schrempf, Michael D Woodhams, Arndt Von Haeseler, and Robert Lanfear. Iq-tree 2: new models and efficient methods for phylogenetic inference in the genomic era. *Molecular biology and evolution*, 37(5):1530–1534, 2020.

Antonio Khalil Moretti, Liyi Zhang, Christian A. Naesseth, Hadiah Venner, David Blei, and Itsik Pe'er. Variational combinatorial sequential Monte Carlo methods for Bayesian phylogenetic inference. *Uncertainty in Artificial Intelligence (UAI)*, 2021.

Jerzy Neyman. Molecular studies of evolution: a source of novel statistical problems. In *Statistical decision theory and related topics*, pp. 1–27. Elsevier, 1971.

Ling Pan, Nikolay Malkin, Dinghuai Zhang, and Yoshua Bengio. Better training of GFlowNets with local credit and incomplete trajectories. *International Conference on Machine Learning (ICML)*, 2023.

Reza Ranjbar, Sedigheh Nazari, and Omid Farahani. Phylogenetic analysis and antimicrobial resistance profiles of escherichia coli strains isolated from uti-suspected patients. *Iranian Journal of Public Health*, 49(9):1743, 2020.

Sebastien Roch. A short proof that phylogenetic tree reconstruction by maximum likelihood is hard. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3(1): 92–94, 2006.

Alex Rogozhnikov. Einops: Clear and reliable tensor manipulations with einstein-like notation. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=oapKSVM2bcj`.

Fredrik Ronquist, Maxim Teslenko, Paul Van Der Mark, Daniel L Ayres, Aaron Darling, Sebastian Höhna, Bret Larget, Liang Liu, Marc A Suchard, and John P Huelsenbeck. MrBayes 3.2: efficient Bayesian phylogenetic inference and model choice across a large model space. *Systematic biology*, 61(3):539–542, 2012.

Amy Y. Rossman, John M. McKemy, Rebecca A. Pardo-Schultheiss, and Hans-Josef Schroers. Molecular studies of the bionectriaceae using large subunit rdna sequences. *Mycologia*, 93(1):100–110, 2001.

Naruya Saitou and Masatoshi Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular biology and evolution*, 4(4):406–425, 1987.

David Sankoff. Minimal mutation trees of sequences. *SIAM Journal on Applied Mathematics*, 28(1):35–42, 1975.

Diane I Scaduto, Jeremy M Brown, Wade C Haaland, Derrick J Zwickl, David M Hillis, and Michael L Metzker. Source identification in two criminal cases using phylogenetic analysis of hiv-1 dna sequences. *Proceedings of the National Academy of Sciences*, 107(50): 21242–21247, 2010.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

R.R. Sokal, C.D. Michener, and University of Kansas. *A Statistical Method for Evaluating Systematic Relationships*. University of Kansas science bulletin. University of Kansas, 1958. URL `https://books.google.ca/books?id=o1BlHAAACAAJ`.

David L Swofford. Phylogenetic analysis using parsimony. 1998.

Hakon Tjelmeland and Bjorn Kare Hegstad. Mode jumping proposals in mcmc. *Scandinavian journal of statistics*, 28(1):205–223, 2001.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Neural Information Processing Systems (NIPS)*, 2017a.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017b.

Martin J Wainwright and Michael I Jordan. Introduction to variational methods for graphical models. *Foundations and Trends in Machine Learning*, 1:1–103, 2008.

Ziheng Yang and Anne D. Yoder. Comparison of Likelihood and Bayesian Methods for Estimating Divergence Times Using Multiple Gene Loci and Calibration Points, with Application to a Radiation of Cute-Looking Mouse Lemur Species. *Systematic Biology*, 52(5):705–716, 10 2003.

Anne D. Yoder and Ziheng Yang. Divergence dates for malagasy lemurs estimated from multiple gene loci: geological and evolutionary context. *Molecular Ecology*, 13(4):757–773, 2004.

Cheng Zhang. Learnable topological features for phylogenetic inference via graph neural networks. *arXiv preprint arXiv:2302.08840*, 2023.

Cheng Zhang and Frederick A Matsen IV. Generalizing tree probability estimation via bayesian networks. *Neural Information Processing Systems (NIPS)*, 2018a.

Cheng Zhang and Frederick A Matsen IV. Variational bayesian phylogenetic inference. *International Conference on Learning Representations (ICLR)*, 2018b.

David Zhang, Corrado Rainone, Markus Peschl, and Roberto Bondesan. Robust scheduling with GFlowNets. *International Conference on Learning Representations (ICLR)*, 2023a.

Dinghuai Zhang, Nikolay Malkin, Zhen Liu, Alexandra Volokhova, Aaron Courville, and Yoshua Bengio. Generative flow networks for discrete probabilistic modeling. *International Conference on Machine Learning (ICML)*, 2022.

Dinghuai Zhang, Hanjun Dai, Nikolay Malkin, Aaron Courville, Yoshua Bengio, and Ling Pan. Let the flows tell: Solving graph combinatorial problems with GFlowNets. *arXiv preprint arXiv:2305.17010*, 2023b.

Dinghuai Zhang, L. Pan, Ricky T. Q. Chen, Aaron C. Courville, and Yoshua Bengio. Distributional gflownets with quantile flows. *arXiv preprint arXiv:2302.05793*, 2023c.

Ning Zhang and Meredith Blackwell. Molecular phylogeny of dogwood anthracnose fungus (discula destructiva) and the diaporthales. *Mycologia*, 93(2):355–365, 2001.

Yiheng Zhu, Jialu Wu, Chaowen Hu, Jiahuan Yan, Chang-Yu Hsieh, Tingjun Hou, and Jian Wu. Sample-efficient multi-objective molecular optimization with gflownets. *arXiv preprint arXiv:2302.04040*, 2023.