Fast Phylogenetic Biodiversity Computations Under a Non-Uniform Random Distribution

Constantinos Tsirogiannis Brody Sandel

MADALGO[*] *and Department of Bioscience*
Aarhus University, Denmark

Abstract

Computing the phylogenetic diversity of a set of species is an important part of many ecological case studies. More specifically, let $\mathcal{T}$ be a phylogenetic tree, and let R be a subset of its leaves representing the species under study. Specialists in ecology want to evaluate a function $f(\mathcal{T}, R)$ (a *phylogenetic measure*) that quantifies the evolutionary distance between the elements in R. But, in most applications, it is also important to examine how $f(\mathcal{T}, R)$ behaves when R is selected at random. The standard way to do this is to compute the mean and the variance of f among all subsets of leaves in $\mathcal{T}$ that consist of exactly $|R| = r$ elements. For certain measures, there exist algorithms that can compute these statistics, under the condition that all subsets of r leaves are equiprobable. Yet, so far there are no algorithms that can do this exactly when the leaves in $\mathcal{T}$ are weighted with unequal probabilities. As a consequence, for this general setting, specialists try to compute the statistics of phylogenetic measures using methods which are both inexact and very slow.

We present for the first time exact and efficient algorithms for computing the mean and the variance of phylogenetic measures when leaf subsets of fixed size are selected from $\mathcal{T}$ under a non-uniform random distribution. In particular, let $\mathcal{T}$ be a tree that has n nodes and depth d, and let r be a non-negative integer. We show how to compute in $O((d + \log n)n \log n)$ time and $O(n)$ space the mean and the variance for any measure that belongs to a well-defined class. We show that two of the most popular phylogenetic measures belong to this class: the Phylogenetic Diversity (PD) and the Mean Pairwise Distance (MPD). The random distribution that we consider is the Poisson binomial distribution restricted to subsets of fixed size r. More than that, we provide a stronger result; specifically for the PD and the MPD we describe algorithms that compute in a batched manner the mean and variance on $\mathcal{T}$ for *all* possible leaf-subset sizes in $O((d + \log n)n \log n)$ time and $O(n)$ space.

For the PD and MPD, we implemented our algorithms that perform batched computations of the mean and variance. We also developed alternative implementations that compute in $O((d + \log n)n^2)$ time the same output. For both types of implementations, we conducted experiments and measured their performance in practice. Despite the difference in the theoretical performance, we show that the algorithms that run in $O((d+\log n)n^2)$ time are more efficient in practice, and numerically more stable. We also compared the performance of these algorithms with standard inexact methods that can be used in case studies. We show that our algorithms are outstandingly faster, making it possible to process much larger datasets than before. Our implementations will become publicly available through the R package `PhyloMeasures`.

[*]Center for Massive Data Algorithmics, a Center of the Danish National Research Foundation.

1 Introduction

One of the most important facets of biological diversity is phylogenetic diversity [4]. Given a certain set of species, ecologists often want to know whether these species are close or distant phylogenetic relatives [24]. This is relevant in estimating the importance of conserving some of these species [5], and can provide insight into the ecological mechanisms that form species communities [16].

To measure the phylogenetic diversity between a set of species, biologists use the following process: first, they choose a phylogenetic tree $\mathcal{T}$ where the examined set of species are represented by a subset of leaf nodes R. The next step is to evaluate a function $f(\mathcal{T}, R)$ which measures the distance between the nodes in R. In the related literature, such functions are referred to as a *measure of phylogenetic α-diversity*; for convenience, in this paper we call a function of this kind a *phylogenetic measure.* There is a large number of phylogenetic measures that have been proposed and used by experts in biology. Two of the most popular measures are the Phylogenetic Diversity (PD) and the Mean Pairwise Distance (MPD–see Section 2 for a formal definition of these measures). Whichever measure f we may use, in most applications it is not enough to compute only the value $f(\mathcal{T}, R)$ for the examined set R; it is also important to find out if $f(\mathcal{T}, R)$ is significantly different than the value of f on a randomly selected leaf subset in $\mathcal{T}$ that has the same size as R. To measure this, ecologists calculate an index that is defined as follows:

$$\mathrm{FI} = \frac{f(\mathcal{T}, R) - \mu_r(f, \mathcal{T})}{\sqrt{\mathrm{var}_r(f, \mathcal{T})}},$$

where $r = |R|$ is the number of elements in R, value $\mu_r(f, \mathcal{T})$ is the mean value of f among all leaf subsets in $\mathcal{T}$ that consist of r elements, and $\mathrm{var}_r(f, \mathcal{T})$ is the variance of f among these subsets. Therefore, to calculate index FI it becomes important to compute exactly and efficiently the mean and variance of f over all leaf subsets of size r. Of course, the value of these two statistical moments depends on the probability distribution that we use to select the leaf subsets. For the special case that leaf subsets are selected with equal probability, there exist several algorithms that efficiently calculate these moments [22, 23, 19].

Yet, in many cases biologists want to take into account that certain species are more abundant than others. This can be done by assigning to each leaf node v in $\mathcal{T}$ a probability value $p(v)$; this value represents the abundance in nature of the species represented by v. In this setting, the goal is to compute $\mu_r(f, \mathcal{T})$ and $\mathrm{var}_r(f, \mathcal{T})$ so that these reflect the probability values associated with the leaves in $\mathcal{T}$. We call these values the *weighted* moments of f. Computing the weighted moments requires also to define a probability distribution that assigns to each subset of r leaves a probability of selection, based on the individual leaf probabilities. However, so far there does not exist any approach that shows how to do this exactly and efficiently. Faller *et al.* provide results for the PD measure, yet they consider a more relaxed model where the mean and the variance are computed among leaf subsets of unequal size [6].

As a result, in the absence of an exact solution, researchers in ecology try to calculate the weighted moments of phylogenetic measures using an inexact sampling approach. According to this approach, several leaf subsets (usually around a thousand, and each consisting of r elements) are selected at random from $\mathcal{T}$. Then, the value of f is calculated for each of these subsets, and the mean and variance of f are approximated based on these calculated values. If r and the number of leaves in $\mathcal{T}$ are sufficiently large, this approach does not guarantee a good approximation of the weighted moments. Even worse, these methods can be very slow given that they require to select and process a large number of samples. Despite that, and due to the lack of other options, these inefficient methods have become the standard approach for estimating the weighted moments of phylogenetic measures. Therefore, there is a need for an exact and also efficient method that calculates the weighted moments of a phylogenetic measure.

Our Results. Inspired by the above, we present for the first time exact and efficient algorithms for computing the weighted moments of phylogenetic measures among leaf subsets of fixed size. More specifically, let $\mathcal{T}$ be a tree that has n nodes and depth d, and let r be a non-negative integer. We show that we can compute in $O((d+\log n)n\log n)$ time and $O(n)$ space the mean and the variance for any measure that belongs to a certain class. We call the measures of this class *edge-decomposable* measures. We show that both PD and MPD belong to this class.

For the algorithms that we propose, we calculate the mean and variance of an edge-decomposable measure based on the following distribution; leaf-subsets are conceptually selected using the Poisson binomial distribution (where each leaf v is picked in a Bernoulli trial with probability $p(v)$), and then a resulting subset is accepted only if it consists of exactly r elements. This process defines a distribution over the leaf subsets that a) takes into account the individual probability value of each leaf in $\mathcal{T}$, and b) maintains that only leaf subsets of the same size are considered.

Specifically for the PD and MPD, we yield a stronger result; we present algorithms that compute in a batched manner the weighted moments of these measures for many leaf-subset sizes. These algorithms compute the mean and variance for these measures for *all* possible leaf-subset sizes in $O((d+\log n)n\log n)$ time.

We implemented the algorithms that perform batched computations of weighted moments for the PD and MPD. We developed two kinds of implementations; we implemented the aforementioned algorithms that run in $O((d+\log n)n\log n)$ time, but also algorithms that run in $O((d+\log n)n^2)$ time and are numerically more stable. For both types of implementations, we conducted experiments and measured their performance in practice. Despite the difference in the theoretical performance, we show that the algorithms that run in $O((d+\log n)n^2)$ time perform better in practice. The latter algorithms are highly parallelisable; with simple adjustments we were able to boost their performance and process fast very large phylogenies. For a tree of 71,181 leaves, our implementations computed the weighted moments for all 71,181 subset sizes in less than two and a half minutes for the PD, and in less than six minutes for the MPD. We compared the performance of the latter implementations with a program that estimates the weighted moments based on an inexact sampling method. In this comparison our algorithms were found to be remarkably faster, making it possible to process much larger datasets than before. We intend to make our implementations publicly available through the R package `PhyloMeasures` [21].

The rest of this paper is organized as follows: in the next section we provide definitions and notation for the basic concepts used in the description of our algorithms. In Section 3 we describe in detail the algorithms that we introduce for computing the weighted moments of certain phylogenetic measures. Finally, in Section 4 we describe the implementations that we developed for computing in a batched manner the mean and variance of PD and MPD for multiple leaf-subset sizes. There we present experiments that show the huge difference in perfomance between our implementations and a standard inexact method.

2 Definitions and Notation

Notation related to phylogenetic trees. Let $\mathcal{T}$ be a phylogenetic tree. We use E to denote the edges of $\mathcal{T}$, and for any edge $e \in E$ we use $w(e)$ to represent the weight of e. We indicate the set of nodes in $\mathcal{T}$ by V, and we indicate the set of leaf nodes in this tree by S. We use n to represent the total number of nodes in $\mathcal{T}$, and we use s to indicate the number of leaves in $\mathcal{T}$.

We consider that $\mathcal{T}$ is a rooted tree and we denote the root node of $\mathcal{T}$ by $\mathrm{root}(\mathcal{T})$. In the case of unrooted trees, we pick an arbitrary node which is not a leaf, and consider this node as the root. For any node $v \in V$ we use $\mathrm{Ch}(v)$ to indicate the set of the child nodes of v. For any two adjacent nodes u and v we use e_{uv} to represent the edge that connects them. We define the depth of $\mathcal{T}$ as the maximum number of edges that appear on a simple path between the root of $\mathcal{T}$ and a leaf. We consider that the maximum degree in $\mathcal{T}$ (the maximum number of nodes adjacent to a single node of $\mathcal{T}$) is upper-bounded by a constant. Except the root, all other internal nodes in $\mathcal{T}$ have degree greater than two. The results described in this work apply also to trees of non-constant maximum degree. We can convert a tree $\mathcal{T}$ of non-constant degree to a binary tree $\mathcal{T}'$ by adding edges of zero weight, in a way that the distances between the leaves are maintained. Thus, the new tree $\mathcal{T}'$ has the same mean and variance as $\mathcal{T}$ for any of the measures that we examine in this work. Tree $\mathcal{T}'$ has $O(n)$ nodes, height $O(d + \log n)$ and can be constructed in $O(n)$ time; here n and d are the number of nodes and the depth of the initial tree $\mathcal{T}$ respectively. We can then apply on $\mathcal{T}'$ any of the algorithms described in this paper; it is easy to show that the asymptotical performance bounds that we provide are still valid for the initial values of parameters n and d.

Let e be an edge in $\mathcal{T}$ and let v be the child node of e. We use interchangeably $\mathcal{T}(e)$ and $\mathcal{T}(v)$ to represent the subtree of $\mathcal{T}$ rooted at v. We use $S(e)$ and $S(v)$ to indicate the subset of leaves that appear in this subtree, and we use $s(e)$ and $s(v)$ to indicate the number of these leaves. We denote the set of edges that appear on $\mathcal{T}(v)$ by $\mathrm{Off}(e)$ and $\mathrm{Off}(v)$. We consider that $e \notin \mathrm{Off}(e)$. We use $\mathrm{Ind}(e)$ to denote the set that consists of every edge $\ell \neq e$ for which neither ℓ appears in $\mathcal{T}(e)$, nor e appears in $\mathcal{T}(\ell)$. Let R be a subset of $|R| = r$ leaves in $\mathcal{T}$, and let e be an edge in E. We use $S_R(e)$ to denote the set of leaves in R which appear in the subtree of e, that is $S_R(e) = S(e) \cap R$. We indicate the number of these leaves by $sr(e)$. We represent the (unique) minimum-size subtree of $\mathcal{T}$ that spans all the leaves in R by $\mathcal{T}(R)$. Let $u, v \in S$ be two leaves in $\mathcal{T}$ and let π be the simple path that connects these leaves. We define the *cost* of π as the sum of the weights of the edges that appear on this path. We represent this cost as $\mathrm{cost}(u, v)$.

Phylogenetic measures. Two of the most popular phylogenetic measures are the Phylogenetic Diversity (PD) and the Mean Pairwise Distance (MPD). The value of the PD for R is equal to:

$$\mathrm{PD}(\mathcal{T}, R) = \sum_{e \in \mathcal{T}(R)} w(e) \ .$$

Hence, the value of the PD for R is the cost of the minimum-size subtree $\mathcal{T}(R)$ in $\mathcal{T}$ that spans all leaves in R.

Let r be the number of elements in R. The MPD of R is equal to the average cost of a simple path in $\mathcal{T}$ that connects any two distinct leaves in R. More formally:

$$\mathrm{MPD}(\mathcal{T}, R) = \frac{2}{r(r-1)} \sum_{\{u,v\} \in R} \mathrm{cost}(u, v) \ .$$

Probability distribution. Let $\mathcal{T}$ be a phylogenetic tree, and let each leaf node $v \in S$ be associated with a probability value $p(v)$. We consider the following random process for selecting a subset of exactly r leaves; each leaf $v \in S$ is initially sampled independently with probability $p(v)$, and if the resulting subset R of sampled leaves consists of exactly r elements then we output R, otherwise we repeat the process. Therefore, the probability of selecting a subset which does not have exactly r elements is zero. The probability that a specific subset R of r leaves is selected according to this process is:

$$\frac{1}{C_r} \prod_{v \in R} p(v) \prod_{u \in S \setminus R} (1 - p(u)) \ , \tag{1}$$

where C_r is a normalising constant equal to:

$$C_r = \sum_{\substack{G \subseteq S \\ |G|=r}} \prod_{x \in G} p(x) \prod_{y \in S \setminus G} (1 - p(y)) \tag{2}$$

In other words, the probability of selecting a specific subset R of r elements is equal to the probability of selecting R with n (non-identical) independent Bernoulli trials, divided by the sum of probabilities of all possible subsets of size r that are picked with such trials. The distribution that is entailed from this model is similar to the Poisson binomial distribution, restricted to subsets of fixed size r [2, 7]. We call this constrained variant the *Restricted Poisson Binomial* distribution, or RPB for short. Note that in the RPB model the selection of two leaves u and v is not statistically independent; this is a consequence of considering only leaf subsets of the same size.

3 Description of Algorithms

3.1 Computing the Mean and Variance of Edge-Decomposable Measures

Let $\mathcal{T}$ be a phylogenetic tree of constant degree, and consider that every leaf v in this tree is assigned a probability value $p(v) \in [0, 1]$. We next focus on the problem of computing the expected value and the variance of phylogenetic measures when subsets of exactly r leaves are selected from $\mathcal{T}$ according to the RPB distribution. In particular, we examine this problem for phylogenetic measures that belong to a certain class. We call the measures of this class *edge-decomposable* measures. Intuitively, we call a measure edge-decomposable if for any input pair $\mathcal{T}$, R we can express $f(\mathcal{T}, R)$ as a sum of terms such that: each term corresponds to exactly one edge $e \in E$, and for each edge e the corresponding term can be evaluated in constant time if we know already the edge weight $w(e)$ and $sr(e)$ (the number of leaves in R appearing in the subtree of e) . More formally, we can express this as follows:

Definition 3.1. *Let $\mathcal{T}$ be a phylogenetic tree and let R be a subset of r leaves in $\mathcal{T}$. A phylogenetic measure f is edge-decomposable if:*

(I) The value of f can be expressed as a sum of the form:

$$f(\mathcal{T}, R) = \sum_{e \in E} w(e) \cdot c(sr(e), r) \ , \tag{3}$$

where c is a function whose definition does not depend on $\mathcal{T}$ or R. We call c the contribution function *of f.*

(II) Given values α and β, we can evaluate $c(\alpha, \beta)$ in constant time.

Lemma 3.2. *Measures* PD *and* MPD *are edge-decomposable.*

Proof. Let $\mathcal{T}$ be a phylogenetic tree, and let R be a subset of r leaves in $\mathcal{T}$. Measure PD can be expressed as in Equation (3) by using the following contribution function:

$$c(\alpha,\beta) = c_{\text{PD}}(\alpha,\beta) = \begin{cases} 1 & \text{if } \beta > 1 \text{ and } 1 \leq \alpha < \beta. \\ 0 & \text{otherwise.} \end{cases}$$

For the MPD it holds that:

$$\text{MPD}(\mathcal{T},R) = \frac{2}{r(r-1)} \sum_{\{u,v\}\in R} \text{cost}(u,v) = \frac{2}{r(r-1)} \sum_{e\in E} w(e) \cdot sr(e) \cdot (r - sr(e)) \ .$$

This is because each edge $e \in E$ appears in $sr(e) \cdot (r - sr(e))$ paths out of the total $r(r-1)/2$ paths between distinct pairs of leaves in R. Therefore, the MPD can be expressed as an edge-decomposable measure using the next contribution function:

$$c_{\text{MPD}}(\alpha,\beta) = \begin{cases} 2\frac{\alpha(\beta-\alpha)}{\beta(\beta-1)} & \text{if } 0 < \alpha < \beta. \\ 0 & \text{otherwise.} \end{cases}$$

Both of the above contribution functions can be evaluated in constant time, and the lemma follows. □

Next we sketch an algorithm that computes the expected value of any
edge-decomposable measure according to the RPB model. Let f be such a measure and let c be its contribution function. Let $\mathcal{T}$ be a tree such that each leaf $v \in S$ is assigned a probability value $p(v) \in [0,1]$. Let $\mathbb{E}_r[f(\mathcal{T},R)]$ represent the expected value of f among all subsets $R \subseteq S(\mathcal{T})$ of exactly r elements selected based on the RPB distribution. This expected value is equal to:

$$\mathbb{E}_r[f(\mathcal{T},R)] = \sum_{e\in E} \sum_{i=0}^{s(e)} w(e) \cdot c(i,r) \cdot \mathbb{P}(sr(e) = i) \ , \tag{4}$$

where $\mathbb{P}(sr(e) = i)$ is the probability that exactly i out of the r elements of a leaf subset fall in the subtree of e. This probability is equal to:

$$\mathbb{P}(sr(e) = i) = \frac{1}{C_r} \sum_{\substack{R\subseteq S \\ |R|=r \text{ and } sr(e)=i}} \prod_{v\in R} p(v) \prod_{u\in S\setminus R} (1 - p(u)) \ , \tag{5}$$

where C_r is the normalising constant defined in Equation (2). From (5) and (2), we observe that computing $\mathbb{P}(sr(e) = i)$ boils down to calculating two sums of products. A naive approach for computing these sums would be to explicitly construct each distinct subset of exactly r leaves in $\mathcal{T}$, and then calculate the respective product for this subset. However, this explicit computation is very inefficient; even for relatively small values of r, the number of terms in each sum becomes very large.

Yet, we can compute $\mathbb{P}(sr(e) = i)$ in a more efficient manner. The key idea is to express the sums in (5) and (2) in terms of coefficients of certain polynomials. More specifically, let G be a subset of the leaves in $\mathcal{T}$. We define Pol_G to be the following univariate polynomial:

$$\text{Pol}_G(x) = \prod_{v\in G} (p(v) \cdot x + (1 - p(v))) \ .$$

Consider rewriting the above polynomial as a sum of the form $\sum_j a_j x^j$. We call this sum the *summation* representation of Pol_G. In this representation, consider the coefficient of the k-th power of x. We indicate this coefficient by $\mathrm{CF}(G,k)$. This is equal to:

$$\mathrm{CF}(G,k) = \sum_{\substack{R \subseteq G \\ |R|=k}} \prod_{v \in R} p(v) \prod_{u \in G \setminus R} (1-p(u)) \ .$$

Based on this observation, it follows directly that $C_r = \mathrm{CF}(S,r)$. More than that, we can express the probability value $\mathbb{P}(sr(e)=i)$ by rewriting (5) as follows:

$$\mathbb{P}(sr(e)=i) = \frac{1}{C_r} \sum_{\substack{R \subseteq S \\ |R|=r \text{ and } sr(e)=i}} \prod_{v \in R} p(v) \prod_{u \in S \setminus R} (1-p(u)) =$$

$$\frac{1}{C_r} \sum_{\substack{G \subseteq S(e) \\ |G|=i}} \prod_{v \in G} p(v) \prod_{u \in S(e) \setminus G} (1-p(u)) \sum_{\substack{M \subseteq S \setminus S(e) \\ |M|=r-i}} \prod_{y \in M} p(y) \prod_{z \in (S \setminus (S(e) \cup M))} (1-p(z)) =$$

$$= \frac{\mathrm{CF}(S(e),i) \cdot \mathrm{CF}(S \setminus S(e), r-i)}{\mathrm{CF}(S,r)} \ . \tag{6}$$

Hence, to compute value $\mathbb{P}(sr(e)=i)$ it suffices to construct the summation representations of polynomials $\mathrm{Pol}_{S(e)}$, $\mathrm{Pol}_{S \setminus S(e)}$, and Pol_S, and then extract the required coefficients. We need the following definitions. From hereon, when we refer to a polynomial P we mean the summation representation of P. Let $A = \sum_j a_j x^j$ and $B = \sum_k b_k x^k$ be two polynomials. We use $A \otimes B$ to represent the *product* of these polynomials; that is polynomial $H = \sum_m h_m x^m$ for which we have $h_t = \sum_{q=0}^{t} a_q b_{i-q}$. Let $P = \sum_{j=0}^{k} a_j x^j$ be a polynomial of degree k, and let T be a vector $(t_0, t_1, \ldots, t_k)$ of $k+1$ coordinates. We use $T \odot P$ to represent polynomial $\sum_{j=0}^{k} t_j a_j x^j$. Let e be a tree edge, and f an edge-decomposable measure with a contribution function c. Let r be a non-negative integer. We use $c(e,r)$ to represent vector $(c(0,r), c(1,r), \ldots, c(\min(s(e),r,r))$. Given any polynomial P, we use $\mathrm{CF}(P,k)$ to denote the k-th coefficient of P. As mentioned already, for any leaf subset $G \subseteq S$ we use for convenience $\mathrm{CF}(G,k)$ to represent the k-th coefficient of polynomial Pol_G.

Let G be a subset of leaves $G \in S$, consisting of g leaves. We can compute Pol_G in $O(g \log^2 g)$ time in the following manner: first we construct all binomials $p(v) \cdot x + (1-p(v))$ such that $v \in G$. Then, we split these polynomials into pairs and multiply the elements of each pair using the Fast Fourrier Transform (FFT) [17]. We repeat this process on the resulting polynomials, until we end up with a single polynomial. Multiplying two polynomials of maximum degree k takes $O(k \log k)$ time with the FFT. At each repetition, the sum of the degrees of the processed polynomials is equal to g, and we perform $O(\log g)$ repetitions in total.

Using the above process, and based on (4) and (6), we could consider the following approach to compute the mean of f: first we compute $C_r = \mathrm{CF}(S,r)$ by constructing Pol_S in $O(n \log^2 n)$ time and extracting the r-th coefficient. Then, for each edge $e \in \mathcal{T}$ we construct from scratch polynomials $\mathrm{Pol}_{S(e)}$ and $\mathrm{Pol}_{S \setminus S(e)}$, and use the coefficients of these polynomials to compute values $\mathbb{P}(sr(e)=i)$ for every integer $i \in [0, s(e)]$. The described approach would require in total $O(n^2 \log^2 n)$ time; for every edge e we need to spend $O(n \log^2 n)$ time to construct $\mathrm{Pol}_{S(e)}$ and $\mathrm{Pol}_{S \setminus S(e)}$ since one of these polynomials has degree which is at least $n/2$. Yet, we can design an algorithm which is more efficient when $\mathcal{T}$ is relatively balanced. We provide the next lemma.

Lemma 3.3. *Let $\mathcal{T}$ be a phylogenetic tree that has n nodes and depth d, and let f be an edge-decomposable measure. Let r be a non-negative integer. We can compute the mean of f for leaf-subsets of size r according to the* RPB *distribution in $O((d+\log n)n\log n)$ time, using $O(n)$ space.*

Proof. From (4) and (6) we get:

$$\mathbb{E}_r[f(\mathcal{T},R)] = \frac{1}{C_r}\cdot\sum_{e\in E}\sum_{i=1}^{\min(r,s(e))} w(e)\cdot c(i,r)\cdot \mathrm{CF}(S(e),i)\cdot \mathrm{CF}(S\setminus S(e),r-i)\ . \tag{7}$$

As mentioned, we can compute C_r in $O(n\log^2 n)$ time. We focus now on the double sum in (7). This is equal to:

$$\sum_{e\in E}\sum_{i=1}^{\min(r,s(e))} w(e)\cdot c(i,r)\cdot \mathrm{CF}(S(e),i)\cdot \mathrm{CF}(S\setminus S(e),r-i) =$$

$$\sum_{e\in E}\mathrm{CF}((w(e)\cdot c(e,r)\odot \mathrm{Pol}_{S(e)})\otimes \mathrm{Pol}_{S\setminus S(e)},r)\ . \tag{8}$$

Hence, the problem boils down to computing the r-th coefficient of a certain polynomial. We denote this polynomial by $\mathrm{AP}(\mathcal{T},r)$, that means:

$$\mathrm{AP}(\mathcal{T},r) = \sum_{e\in E}(w(e)\cdot c(e,r)\odot \mathrm{Pol}_{S(e)})\otimes \mathrm{Pol}_{S\setminus S(e)}\ . \tag{9}$$

We need a few more definitions; we use $\mathrm{AP}(v,r)$ to represent $\mathrm{AP}(\mathcal{T}(v),r)$. Obviously, if v is the root of $\mathcal{T}$ we have $\mathrm{AP}(v,r)=\mathrm{AP}(\mathcal{T},r)$. If v is a leaf node, then $\mathrm{AP}(v,r)=0$. If v is an internal node, we can express $\mathrm{AP}(v,r)$ in terms of the respective values of the children of v. More specifically:

$$\mathrm{AP}(v,r) = \sum_{u\in Ch(v)}\mathrm{AP}(u,r)\otimes \mathrm{Pol}_{S(v)\setminus S(u)} + \sum_{u\in \mathrm{Ch}(v)}(w(e_{uv})\cdot c(e_{uv},r)\odot \mathrm{Pol}_{S(u)})\otimes \mathrm{Pol}_{S(v)\setminus S(u)}\ . \tag{10}$$

Recall that e_{uv} denotes the tree edge that connects nodes v and u.

Based on (10), we derive the following algorithm for computing $\mathrm{AP}(\mathcal{T},r)$; we partition the nodes in $\mathcal{T}$ into $d+1$ subsets, where each subset consists of all nodes that belong to the same level of $\mathcal{T}$ (i.e. they have the same depth). Then, we process the nodes of a single level at a time, starting from the bottom level of $\mathcal{T}$ and moving upwards. For each level that we process, we compute for each node v in that level polynomials $\mathrm{AP}(v,r)$ and $\mathrm{Pol}_{S(v)}$ based on Equation (9). The last group that we process consists only of the root node of $\mathcal{T}$. As mentioned, for this node we have $\mathrm{AP}(v,r)=\mathrm{AP}(\mathcal{T},r)$. We extract the r-th coefficient of this polynomial and together with the computed value of C_r, and using (7), (8), and (9) we calculate $\mathbb{E}_r[f(\mathcal{T},R)]$.

Next, we analyse the running-time of the described algorithm. Let v be a node in $\mathcal{T}$, and suppose that we have already computed polynomials $\mathrm{AP}(u,r)$ and $\mathrm{Pol}_{S(u)}$ for every child u of v.

To compute $\mathrm{Pol}_{S(v)}$, we need to multiply all polynomials $\mathrm{Pol}_{S(u)}$, $u\in \mathrm{Ch}(u)$. Note that v has a constant number of children, and each polynomial $\mathrm{Pol}_{S(u)}$ has degree at most $s(v)$. Therefore, computing $\mathrm{Pol}_{S(v)}$ boils down to performing a constant number of multiplications between polynomials of degree at most $s(v)$. Using FFT, we can do this in $O(s(v)\log s(v))$ time.

To construct $\mathrm{AP}(u,r)$ according to Equation (10), we compute two more polynomials for each $u \in \mathrm{Ch}(v)$: these are $\mathrm{Pol}_{S(v)\setminus S(u)}$, and $P(u) = \mathrm{AP}(u,r) + (w(e_{uv}) \cdot c(e_{uv}, r) \cdot \mathrm{Pol}_{S(u)})$. Polynomial $\mathrm{Pol}_{S(v)\setminus S(u)}$ is equal to the product of all polynomials $\mathrm{Pol}_{S(y)}$ such that y is a child of v and $y \neq u$. Based on the same arguments for calculating $\mathrm{Pol}_{S(u)}$, we can compute $\mathrm{Pol}_{S(v)\setminus S(u)}$ in $O(s(v) \log s(v))$ time. Given $\mathrm{AP}(u,r)$ and $\mathrm{Pol}_{S(u)}$, it is straightforward to construct $P(u)$ in $O(s(u))$ time. It remains to compute polynomial $P(u) \otimes \mathrm{Pol}_{S(v)\setminus S(u)}$ and add this to $\mathrm{AP}(v,r)$ (initially with all coefficients set to zero). This last step requires $O(s(v) \log s(v))$ time. Node v has a constant number of children, hence $\mathrm{AP}(v,r)$ and $\mathrm{Pol}_{S(v)}$ can be computed in $O(s(v) \log s(v))$ time given the respective polynomials of the child nodes. We charge this time to node v. Based on this scheme, the total time charged to the nodes that appear on the same level of $\mathcal{T}$ is $O(n \log n)$. Tree $\mathcal{T}$ has $d+1$ levels, and the proof for the running-time follows.

To prove the space bound, recall that we process the nodes in $\mathcal{T}$ level-by-level. Thus, at any point during this process we need to store polynomials $\mathrm{AP}(v,r)$ and $\mathrm{Pol}_{S(v)}$ only for the nodes of two consecutive levels in $\mathcal{T}$. The total number of terms of these polynomials among all nodes in a single level of $\mathcal{T}$ is $O(n)$, and the proof follows. □

In fact, we can derive an equally efficient algorithm for computing the variance of f.

Lemma 3.4. *Let $\mathcal{T}$ be a phylogenetic tree that has n nodes and depth d, and let f be an edge-decomposable measure. Let r be a non-negative integer. We can compute the variance of f for leaf-subsets of size r according to the* RPB *distribution in $O((d + \log n) n \log n)$ time, using $O(n)$ space.*

Proof. Let $\mathbb{V}_r[f(\mathcal{T}, R)]$ represent the variance of f for subsets of r leaves picked according to the RPB distribution. Based on a standard formula that expresses the variance of a random variable, we get:

$$\mathbb{V}_r[f(\mathcal{T}, R)] = \mathbb{E}_r[f^2(\mathcal{T}, R)] - \mathbb{E}_r^2[f(\mathcal{T}, R)] \ . \tag{11}$$

We already showed in Lemma 3.3 how to calculate the expected value of f, so next we describe how to compute the expected value of f^2. We have that:

$$\mathbb{E}_r[f^2(\mathcal{T}, R)] = \sum_{e \in E} w(e) \sum_{\ell \in E} w(\ell) \sum_{i=1}^{\min(r,s(e))} c(i,r) \sum_{j=1}^{\min(r,s(\ell))} c(j,r) \cdot \mathbb{P}(s(e) = i \text{ and } s(\ell) = j) \ . \tag{12}$$

We break the sum in (12) as follows; for an edge $e \in E$, recall that $\mathrm{Off}(e)$ denotes the set of edges in $\mathcal{T}(e)$, and $\mathrm{Ind}(e)$ the set of all edges $\ell \in E \setminus \{e\}$ such that $\ell \notin \mathcal{T}(e)$ and $e \notin \mathcal{T}(\ell)$. We get that:

$$\sum_{e \in E} w(e) \sum_{\ell \in E} w(\ell) \sum_{i=1}^{\min(r,s(e))} c(i,r) \sum_{j=1}^{\min(r,s(\ell))} c(j,r) \cdot \mathbb{P}(s(e) = i \text{ and } s(\ell) = j) =$$

$$2 \sum_{e \in E} \sum_{\ell \in \mathrm{Off}(e)} M(e,\ell) + 2 \sum_{e \in E} \sum_{\ell \in \mathrm{Ind}(e)} M(e,\ell) + \sum_{e \in E} w^2(e) \sum_{i=1}^{\min(r,s(e))} c(i,r) \cdot \mathbb{P}(s(e) = i) \ , \tag{13}$$

where:

$$M(e,\ell) = w(e) \cdot w(\ell) \sum_{i=1}^{\min(r,s(e))} c(i,r) \sum_{j=1}^{\min(r,s(\ell))} c(j,r) \cdot \mathbb{P}(s(e)=i \text{ and } s(\ell)=j) \ . \tag{14}$$

For the last sum in (13), and based on Equation (9) we get that:

$$\sum_{e \in E} w^2(e) \sum_{i=1}^{\min(r,s(e))} c(i,r) \cdot \mathbb{P}(s(e)=i) = \mathbb{E}_r[f(\mathcal{T}^2, R)] = \mathrm{AP}(\mathcal{T}^2, r) \ , \tag{15}$$

where $\mathcal{T}^2$ is a tree that has the same combinatorial structure as $\mathcal{T}$, except that for every $e \in \mathcal{T}$ the corresponding edge in $\mathcal{T}^2$ has weight $w^2(e)$. Therefore, according to Lemma 3.3, we can compute $\mathbb{E}_r[f(\mathcal{T}^2, R)]$ in $O((d+\log n)n \log n)$ time. We continue with simplifying the two double sums that appear in (13). For the first double sum we have:

$$\sum_{e \in E} \sum_{\ell \in \mathrm{Off}(e)} M(e,\ell) = \sum_{e \in E} w(e) \sum_{\ell \in \mathrm{Off}(e)} w(\ell) \sum_{i=1}^{\min(r,s(e))} c(i,r) \sum_{j=1}^{\min(r,s(\ell))} c(j,r) \cdot \mathbb{P}(s(e)=i \text{ and } s(\ell)=j) \ . \tag{16}$$

Since we select subsets of r leaves according to the RPB model,
the probability $\mathbb{P}(s(e)=i \text{ and } s(\ell)=j)$ for two edges e and $\ell \in \mathrm{Off}(e)$ is equal to:

$$\mathbb{P}(s(e)=i \text{ and } s(\ell)=j) = \frac{1}{C_r} \mathrm{CF}(S(\ell), j) \cdot \mathrm{CF}(S(e) \setminus S(\ell), i-j) \cdot \mathrm{CF}(S \setminus S(e), r-i) \ . \tag{17}$$

By applying this to (16) we get:

$$\sum_{e \in E} \sum_{\ell \in \mathrm{Off}(e)} M(e,\ell) =$$

$$\frac{1}{C_r} \sum_{e \in E} w(e) \sum_{\ell \in \mathrm{Off}(e)} w(\ell) \sum_{i=1}^{\min(r,s(e))} c(i,r) \sum_{j=1}^{\min(r,s(\ell))} c(j,r) \cdot \mathrm{CF}(S(\ell), j) \cdot \mathrm{CF}(S(e) \setminus S(\ell), i-j) \cdot \mathrm{CF}(S \setminus S(e), r-i) \ . \tag{18}$$

We continue by simplifying the last quantity. To do this, we use the following simple property that derives from the definition of polynomial multiplication; let P_A and P_B be two polynomials of degrees a and b respectively, and let k be a positive integer such that $k \leq \min(a,b)$. It holds that:

$$\sum_{h=0}^{k} \mathrm{CF}(P_A, h) \cdot \mathrm{CF}(P_B, k-h) = \mathrm{CF}(P_A \otimes P_B, k) \ . \tag{19}$$

Using this property on (18) we get:

$$\frac{1}{C_r}\sum_{e\in E} w(e) \sum_{\ell\in \mathrm{Off}(e)} w(\ell) \sum_{i=1}^{\min(r,s(e))} c(i,r) \sum_{j=1}^{\min(r,s(\ell))} c(j,r)\cdot \mathrm{CF}(S(\ell),j)\cdot \mathrm{CF}(S(e)\setminus S(\ell), i-j)\cdot \mathrm{CF}(S\setminus S(e), r-i) =$$

$$\frac{1}{C_r}\sum_{e\in E} w(e) \sum_{i=1}^{\min(r,s(e))} c(i,r)\cdot \mathrm{CF}(S\setminus S(e), r-i)\cdot \mathrm{CF}(\sum_{\ell\in \mathrm{Off}(e)} w(\ell)\cdot c(e,r)\odot \mathrm{Pol}_{S(\ell)})\otimes \mathrm{Pol}_{S(e)\setminus S(\ell)}, i) =$$

$$\frac{1}{C_r}\sum_{e\in E} w(e) \sum_{i=1}^{\min(r,s(e))} c(i,r)\cdot \mathrm{CF}(S\setminus S(e), r-i)\cdot \mathrm{CF}(\mathrm{AP}(e,r), i)\ , \tag{20}$$

where $\mathrm{AP}(e,r) = \mathrm{AP}(\mathcal{T}(e), r)$ is the polynomial that we defined in the proof of Lemma 3.3. Applying once more the property described in (19), we get:

$$\frac{1}{C_r}\sum_{e\in E} w(e) \sum_{i=1}^{\min(r,s(e))} c(i,r)\cdot \mathrm{CF}(S\setminus S(e), r-i)\cdot \mathrm{CF}(\mathrm{AP}(e,r)\otimes \mathrm{Pol}_{S(e)}, i) =$$

$$\frac{1}{C_r}\sum_{e\in E} \mathrm{CF}((w(e)\cdot c(i,r)\odot \mathrm{AP}(e,r))\otimes \mathrm{Pol}_{S\setminus S(e)}, r)\ . \tag{21}$$

We denote the polynomial that appears in (21) (without the $1/C_r$ constant) by $\mathrm{OP}(\mathcal{T}, r)$. That is:

$$\mathrm{OP}(\mathcal{T}, r) = \sum_{e\in E}(w(e)\cdot c(i,r)\odot \mathrm{AP}(e,r))\otimes \mathrm{Pol}_{S\setminus S(e)}\ . \tag{22}$$

Later in this proof we describe how to efficiently compute this polynomial. We continue now with simplifying the last double sum that remains in Equation (13). This is:

$$\sum_{e\in E}\sum_{\ell\in \mathrm{Ind}(e)} M(e,\ell) = \sum_{e\in E} w(e) \sum_{\ell\in \mathrm{Ind}(e)} w(\ell) \sum_{i=1}^{\min(r,s(e))} c(i,r) \sum_{j=1}^{\min(r,s(\ell))} c(j,r)\cdot \mathbb{P}(sr(e)=i \text{ and } sr(\ell)=j) =$$

$$\sum_{(e,\ell)\in \mathrm{IPairs}(E)} w(e)\cdot w(\ell) \sum_{i=1}^{\min(r,s(e))} c(i,r) \sum_{j=1}^{\min(r,s(\ell))} c(j,r)\cdot \mathbb{P}(sr(e)=i \text{ and } sr(\ell)=j)\ , \tag{23}$$

where $\mathrm{IPairs}(E)$ indicates the set of distinct pairs of edges (e,ℓ) for which it holds $\ell\in \mathrm{Ind}(e)$. For two edges e and $\ell\in \mathrm{Ind}(e)$, probability $\mathbb{P}(sr(e)=i \text{ and } sr(\ell)=j)$ is equal to:

$$\mathbb{P}(s(e)=i \text{ and } s(\ell)=j) = \frac{1}{C_r}\mathrm{CF}(S(\ell), j)\cdot \mathrm{CF}(S(e), i)\cdot \mathrm{CF}(S\setminus (S(e)\cup S(\ell)), r-i-j)\ . \tag{24}$$

Combining (23) with (24) we get:

$$\sum_{e \in E} \sum_{\ell \in \mathrm{Ind}(e)} M(e,\ell) =$$

$$\sum_{(e,\ell) \in \mathrm{IPairs}(E)} w(e) \cdot w(\ell) \sum_{i=1}^{\min(r,s(e))} c(i,r) \sum_{j=1}^{\min(r,s(\ell))} c(j,r) \cdot \mathrm{CF}(S(\ell),j) \cdot \mathrm{CF}(S(e),i) \cdot \mathrm{CF}(S \setminus (S(e) \cup S(\ell)), r-i-j) \ . \tag{25}$$

Applying the property described in (19), we get:

$$\sum_{(e,\ell) \in \mathrm{IPairs}(E)} w(e) \cdot w(\ell) \sum_{i=1}^{\min(r,s(e))} c(i,r) \sum_{j=1}^{\min(r,s(\ell))} c(j,r) \cdot \mathrm{CF}(S(\ell),j) \cdot \mathrm{CF}(S(e),i) \cdot \mathrm{CF}(S \setminus (S(e) \cup S(\ell)), r-i-j) =$$

$$\sum_{(e,\ell) \in \mathrm{IPairs}(E)} w(e) \sum_{i=1}^{\min(r,s(e))} c(i,r) \cdot \mathrm{CF}(S(e),i) \cdot \mathrm{CF}((w(\ell) \cdot c(\ell,r) \odot \mathrm{Pol}_{S(\ell)}) \otimes \mathrm{Pol}_{S \setminus (S(e) \cup S(\ell))}, r-i) =$$

$$\sum_{(e,\ell) \in \mathrm{IPairs}(E)} \mathrm{CF}((w(e) \cdot c(e,r) \odot \mathrm{Pol}_{S(e)}) \otimes (w(\ell) \cdot c(\ell,r) \odot \mathrm{Pol}_{S(\ell)}) \otimes \mathrm{Pol}_{S \setminus (S(e) \cup S(\ell))}, r) \ . \tag{26}$$

We denote the polynomial that appears in the last expression by $\mathrm{IP}(\mathcal{T}, r)$. Hence:

$$\mathrm{IP}(\mathcal{T}, r) = \sum_{(e,\ell) \in \mathrm{IPairs}(E)} (w(e) \cdot c(e,r) \odot \mathrm{Pol}_{S(e)}) \otimes (w(\ell) \cdot c(\ell,r) \odot \mathrm{Pol}_{S(\ell)}) \otimes \mathrm{Pol}_{S \setminus (S(e) \cup S(\ell))} \ . \tag{27}$$

To compute the variance of f it remains to compute polynomials $\mathrm{OP}(\mathcal{T}, r)$ and $\mathrm{IP}(\mathcal{T}, r)$, extract the r-th coefficient from those and plug them in the place of the double sums in (13). Next we show how to efficiently compute these two polynomials.

Let v be a vertex in $\mathcal{T}$ and define $\mathrm{OP}(v,r) = \mathrm{OP}(\mathcal{T}(v), r)$, and $\mathrm{IP}(v,r) = \mathrm{IP}(\mathcal{T}(v), r)$. Observe that for the root node $\mathrm{root}(\mathcal{T})$ we have $\mathrm{OP}(\mathrm{root}(\mathcal{T}), r) = \mathrm{OP}(\mathcal{T}, r)$ and $\mathrm{IP}(\mathrm{root}(\mathcal{T}), r) = \mathrm{IP}(\mathcal{T}, r)$. Our approach is to compute $\mathrm{OP}(v,r)$ and $\mathrm{IP}(v,r)$ for every $v \in V$, and then extract the polynomials that we computed for the root node. To do this, we process $\mathcal{T}$ bottom-up, and for each node v we compute $\mathrm{OP}(v,r)$ and $\mathrm{IP}(v,r)$ based on the corresponding polynomials of the child nodes of v. For this reason, we need to express $\mathrm{OP}(v,r)$ and $\mathrm{IP}(v,r)$ in terms of polynomials $\mathrm{OP}(u,r)$ and $\mathrm{IP}(u,r)$, $u \in \mathrm{Ch}(v)$. In particular, for $\mathrm{OP}(v,r)$ we have:

$$\mathrm{OP}(v,r) = \sum_{u \in \mathrm{Ch}(v)} (\mathrm{OP}(u,r) + w(e_{uv}) \cdot c(e_{uv}, r) \odot \mathrm{AP}(u,r)) \otimes \mathrm{Pol}_{S(v) \setminus S(u)} \tag{28}$$

Before we show how to construct $\mathrm{IP}(u,r)$, we need to refine the expression of this polynomial. Let $E(v)$ denote the set of edges in $\mathcal{T}(v)$. Let v be a node in $\mathcal{T}$ and let (e,ℓ) be two edges in $\mathrm{IPairs}(E(v))$. We use $\mathrm{anc}(e,\ell)$ to indicate the deepest node u in $\mathcal{T}(v)$ such that both edges e and ℓ belong to subtree $\mathcal{T}(u)$. Based on this concept, we partition the pairs in $\mathrm{IPairs}(E(v))$ into two subsets; the pairs for which $\mathrm{anc}(e,\ell) = v$, and the rest of the pairs. According to this partition, we can express $\mathrm{IP}(v,r)$ as follows:

$$\mathrm{IP}(v,r) = \sum_{(e,\ell)\in\mathrm{IPairs}(E(v))} (w(e)\cdot c(e,r)\odot \mathrm{Pol}_{S(e)}) \otimes (w(\ell)\cdot c(\ell,r)\odot \mathrm{Pol}_{S(\ell)}) \otimes \mathrm{Pol}_{S\setminus(S(e)\cup S(\ell))} =$$

$$\sum_{\substack{(e,\ell)\in\mathrm{IPairs}(E(v))\\ \mathrm{anc}(e,\ell)=v}} (w(e)\cdot c(e,r)\odot \mathrm{Pol}_{S(e)}) \otimes (w(\ell)\cdot c(\ell,r)\odot \mathrm{Pol}_{S(\ell)}) \otimes \mathrm{Pol}_{S\setminus(S(e)\cup S(\ell))}$$

$$+ \sum_{\substack{(e,\ell)\in\mathrm{IPairs}(E(v))\\ \mathrm{anc}(e,\ell)\neq v}} (w(e)\cdot c(e,r)\odot \mathrm{Pol}_{S(e)}) \otimes (w(\ell)\cdot c(\ell,r)\odot \mathrm{Pol}_{S(\ell)}) \otimes \mathrm{Pol}_{S\setminus(S(e)\cup S(\ell))}\ . \tag{29}$$

For the first sum in (29) we have:

$$\sum_{\substack{(e,\ell)\in\mathrm{IPairs}(E(v))\\ \mathrm{anc}(e,\ell)=v}} (w(e)\cdot c(e,r)\odot \mathrm{Pol}_{S(e)}) \otimes (w(\ell)\cdot c(\ell,r)\odot \mathrm{Pol}_{S(\ell)}) \otimes \mathrm{Pol}_{S\setminus(S(e)\cup S(\ell))} =$$

$$\sum_{u\in\mathrm{Ch}(v)} \sum_{y\in\mathrm{Ch}(v)\setminus\{u\}} \mathrm{AP}(u,r)\otimes \mathrm{AP}(y,r)\otimes \mathrm{Pol}_{S(v)\setminus(S(u)\cup S(y))}\ . \tag{30}$$

For the second sum in (29) we have:

$$\sum_{\substack{(e,\ell)\in\mathrm{IPairs}(E(v))\\ \mathrm{anc}(e,\ell)\neq v}} (w(e)\cdot c(e,r)\odot \mathrm{Pol}_{S(e)}) \otimes (w(\ell)\cdot c(\ell,r)\odot \mathrm{Pol}_{S(\ell)}) \otimes \mathrm{Pol}_{S\setminus(S(e)\cup S(\ell))} =$$

$$\sum_{u\in\mathrm{Ch}(v)} \mathrm{IP}(u,r)\otimes \mathrm{Pol}_{S(v)\setminus S(u)}\ . \tag{31}$$

Combining the resulting quantities in (30) and (31) with (29) we get:

$$\mathrm{IP}(v,r) = \sum_{\substack{u,y\in\mathrm{Ch}(v)\\ u\neq y}} \mathrm{AP}(u,r)\otimes \mathrm{AP}(y,r)\otimes \mathrm{Pol}_{S(v)\setminus(S(u)\cup S(y))} + \sum_{u\in\mathrm{Ch}(v)} \mathrm{IP}(u,r)\otimes \mathrm{Pol}_{S(v)\setminus S(u)}\ . \tag{32}$$

To compute $\mathrm{OP}(\mathrm{root}(\mathcal{T}),r)$ and $\mathrm{IP}(\mathrm{root}(\mathcal{T}),r)$ we process the nodes in $\mathcal{T}$ level by level, starting from the bottom of the tree and moving upwards one level at a time; for each node v that we process, we compute $\mathrm{OP}(v,r)$ and $\mathrm{IP}(v,r)$ according to Equation (28) and Equation (32), and using the corresponding polynomials of the children of v. Based on these equations, and given that every node in $\mathcal{T}$ has a constant number of children, we conclude that we can compute either of $\mathrm{OP}(v,r)$ and $\mathrm{IP}(v,r)$ with a constant number of multiplications between polynomials of maximum degree $s(v)$ (and also a constant number of scalar products with vectors of the same size). Using the FFT, we can do this in $O(s(v)\log s(v))$ time. For all nodes that appear in the same level of $\mathcal{T}$ we have $\sum s(v) = O(n)$; therefore the total time needed for computing pairs of polynomials OP and IP for all nodes in a level is $O(n\log n)$. Since $\mathcal{T}$ has d levels, the total time for computing these polynomials, and therefore $\mathrm{OP}(\mathcal{T},r)$ and $\mathrm{IP}(\mathcal{T},r)$ is $O((d+\log n)n\log n)$. Because we process the nodes one level at a time, at any point we need to store polynomials OP, IP, but also AP for all the nodes of only two levels in $\mathcal{T}$. The sum of the degrees (and therefore the representation size) of all these polynomials among the nodes of a single level in $\mathcal{T}$ is $O(n)$, and the theorem follows. $\square$

3.2 Batched Computations for the Moments of Popular Measures

So far, we showed that we can efficiently compute the mean and the variance for any edge-decomposable measure f; we showed that this can be done for a single subset size r in $O((d + \log n)n \log n)$ time according to the RPB model. The described algorithms are generic, and work for any edge-decomposable measure f; these algorithms use the contribution function of f as a black box for their computations, and they do not take into account how this function is defined. This gives rise to the following question; given a specific phylogenetic measure, can we derive more efficient algorithms which are especially designed for this measure? Indeed, in the rest of this section we show that we can do this for two popular measures; these are the PD and MPD. In particular, let $\mathcal{T}$ be a tree that has s leaves and n nodes in total. For each of these measures we provide algorithms that calculate the mean and the variance in $O((d + \log n)n \log n)$ time for *all* subset sizes in the range $\{1, 2, \ldots, s\}$. That means, instead of spending $O((d + \log n)n \log n)$ time for computing the statistical moments of a measure for a single subset size r, we can compute the moments for all s possible leaf subset sizes in asymptotically the same time. We state formally our results for PD and MPD in the following theorem.

Theorem 3.5. *Let $\mathcal{T}$ be a phylogenetic tree that has n nodes and depth d. We can compute the mean and the variance of the* PD *and* MPD *on $\mathcal{T}$ in the* RPB *model for all possible leaf-subset sizes in $O((d + \log n)n \log n)$ time in total. The space required for these computations is $O(n)$.*

Proof. In the approach described in Section 3.1, we computed the moments of an edge-decomposable measure f for a single subset size r as follows; we constructed certain polynomials ($\mathrm{AP}(\mathcal{T}, r)$ for the mean, and $\mathrm{AP}(\mathcal{T}^2, r)$, $\mathrm{OP}(\mathcal{T}, r)$ and $\mathrm{IP}(\mathcal{T}, r)$ for the variance), and then we used only one coefficient from each polynomial to derive the value of the required moment. The question that naturally arises is if we can use the rest of the coefficients from these polynomials to compute the moments for other leaf subset sizes. Unfortunately, this does not seem possible for all measures; looking at the definition of $\mathrm{AP}(\mathcal{T}, r)$, $\mathrm{OP}(\mathcal{T}, r)$ and $\mathrm{IP}(\mathcal{T}, r)$ (–see Equations (9), (22) and (27)) we see that these polynomials are constructed using vectors of the form $c(e, r)$, and the values in these vectors depends on r. Yet, for certain measures we can efficiently process the rest of the coefficients in $\mathrm{AP}(\mathcal{T}, r)$, $\mathrm{OP}(\mathcal{T}, r)$ and $\mathrm{IP}(\mathcal{T}, r)$ to derive the moments for other leaf subset sizes. Next we show how to do this for the PD, and then we continue with the MPD.

Let $c_{\mathrm{PD}}(e, r)$ be the contribution function of PD and let $\mathrm{AP}_{\mathrm{PD}}(\mathcal{T}, r)$, $\mathrm{OP}_{\mathrm{PD}}(\mathcal{T}, r)$ and $\mathrm{IP}_{\mathrm{PD}}(\mathcal{T}, r)$ be the polynomials $\mathrm{AP}(\mathcal{T}, r)$, $\mathrm{OP}(\mathcal{T}, r)$ and $\mathrm{IP}(\mathcal{T}, r)$ that we get in the specific case that f is the PD.

The moments of PD. We start with the computation of the mean for the PD. According to (7)-(9), this is equal to:

$$\frac{1}{C_r}\mathrm{AP}_{\mathrm{PD}}(\mathcal{T}, r) = \frac{1}{C_r}\sum_{e \in E}(w(e) \cdot c_{\mathrm{PD}}(e, r) \odot \mathrm{Pol}_{S(e)}) \otimes \mathrm{Pol}_{S \setminus S(e)} \ , \tag{33}$$

where $c_{\mathrm{PD}}(e, r)$ is the contribution vector of PD for edge e. Let $c_{\mathrm{PD}}(e, r)[i]$ denote the i-th coordinate of this vector, with $0 \leq i \leq s(e)$ Based on the definition of the PD, we have that $c_{\mathrm{PD}}(e, r)[i] = 1$ for every $i \in \{2, \ldots, r - 1\}$, otherwise this value zero. Let $G \subseteq S$ be a subset of the leaves in $\mathcal{T}$, and let i be an integer such that $i \leq |G|$. We use to Pol^i_G to indicate the following polynomial:

$$\mathrm{Pol}^i_G = \sum_{j=1}^{i} \mathrm{CF}(G, j)x^j \ . \tag{34}$$

Using this and the coordinates in $c_{\mathrm{PD}}(e,r)$, we can rewrite the last quantity in (33) as:

$$\frac{1}{C_r}\sum_{e\in E}(w(e)\cdot c(e,r)\odot \mathrm{Pol}_{S(e)})\otimes \mathrm{Pol}_{S\setminus S(e)} = \frac{1}{C_r}\sum_{e\in E}(w(e)\cdot \mathrm{Pol}_{S(e)}^{r-1})\otimes \mathrm{Pol}_{S\setminus S(e)}\ . \tag{35}$$

Let k be a non-negative integer such that $k < r$. The value of the k-th coefficient in $\mathrm{AP}(\mathcal{T}, r)$ is:

$$\mathrm{CF}(\mathrm{AP}_{\mathrm{PD}}(\mathcal{T},r),k) = \mathrm{CF}(\sum_{e\in E}(w(e)\cdot \mathrm{Pol}_{S(e)}^{k})\otimes \mathrm{Pol}_{S\setminus S(e)},k)\ . \tag{36}$$

Based on (35) and (36), the expected value of the PD among leaf subsets of size k is equal to:

$$\mathbb{E}_k[\mathrm{PD}(\mathcal{T},R)] = \frac{1}{C_k}\mathrm{CF}(\mathrm{AP}_{\mathrm{PD}}(\mathcal{T},k),k) = \frac{1}{C_k}\sum_{e\in E}\mathrm{CF}((w(e)\cdot \mathrm{Pol}_{S(e)}^{k-1})\otimes \mathrm{Pol}_{S\setminus S(e)},k) =$$

$$\frac{1}{C_k}\sum_{e\in E}\mathrm{CF}((w(e)\cdot \mathrm{Pol}_{S(e)}^{k})\otimes \mathrm{Pol}_{S\setminus S(e)},k) - w(e)\cdot \mathrm{CF}(S(e),k)\cdot \mathrm{CF}(S\setminus S(e),0) =$$

$$\frac{C_r}{C_k}\mathrm{CF}(\mathrm{AP}_{\mathrm{PD}}(\mathcal{T},r),k) - \sum_{e\in E}\mathrm{CF}(S(e),k)\cdot \mathrm{CF}(S\setminus S(e),0)\ . \tag{37}$$

Define $\mathrm{VA}(\mathcal{T})$ as a vector of $n+1$ coordinates, where the i-th value is equal to:

$$\mathrm{VA}(\mathcal{T})[i] = \sum_{e\in E}\mathrm{CF}(S(e),i)\cdot \mathrm{CF}(S\setminus S(e),0)\ .$$

Recall also that C_k is the k-th coefficient of polynomial Pol_S. According to (37), if we have already computed polynomials $\mathrm{AP}_{\mathrm{PD}}(\mathcal{T},r)$ and Pol_S, and vector $\mathrm{VA}(\mathcal{T})$ then we can calculate $\mathbb{E}_k[\mathrm{PD}(\mathcal{T},R)]$ in constant time. We already showed that polynomials like $\mathrm{AP}_{\mathrm{PD}}(\mathcal{T},r)$ and Pol_S can be constructed in $O(n\log n(d+\log n))$ time. To prove the part of the lemma that refers to the mean of the PD, it remains to construct $\mathrm{VA}(\mathcal{T})$ using the same asymptotic time and linear space. We do this in the following manner; let $\mathrm{VA}(v)$ denote vector $\mathrm{VA}(\mathcal{T}(v))$. This vector $\mathrm{VA}(v)$ can be expressed as follows:

$$\mathrm{VA}(v)[i] = \sum_{u\in \mathrm{Ch}(v)}(\mathrm{VA}(u)[i] + \mathrm{CF}(w(e_{uv})\cdot \mathrm{Pol}_{S(u)},i))\cdot \mathrm{CF}(S(v)\setminus S(u),0)\ . \tag{38}$$

Similar to the method described for computing $\mathrm{AP}(\mathcal{T},r)$, we compute $\mathrm{VA}(v)$ for all nodes $v\in V$ by processing bottom-up $\mathcal{T}$ level by level, and then extract vector $\mathrm{VA}(\mathrm{root}(v)) = \mathrm{VA}(\mathcal{T})$. For each processed node v we compute $\mathrm{VA}(v)$ based on (38) and $\mathrm{Pol}_{S(v)}$, using values $\mathrm{Pol}_{S(u)}$ and $\mathrm{VA}(u)$. With the same arguments we used for computing $\mathrm{AP}(v,r)$, it is easy to conclude that we can construct $\mathrm{VA}(v)$, $\forall v\in V$, and therefore $\mathrm{VA}(\mathcal{T})$ in $O((d+\log n)n\log n)$ time using $O(n)$ space.

As for the variance of the PD, recall that from (11) we only need to focus on $\mathbb{E}_r[\mathrm{PD}^2(\mathcal{T},R)]$, the expectation of the square of this measure. By applying the contribution function of PD to equations (11)-(27), and for leaf subset size r we get:

$$\mathbb{E}_r[\mathrm{PD}^2(\mathcal{T},R)] = \frac{1}{C_r}\mathrm{CF}(\mathrm{AP}_{\mathrm{PD}}(\mathcal{T}^2,r) + 2\cdot \mathrm{OP}_{\mathrm{PD}}(\mathcal{T},r) + 2\cdot \mathrm{IP}_{\mathrm{PD}}(\mathcal{T},r),r)\ , \tag{39}$$

where:

$$\mathrm{OP}_{\mathrm{PD}}(\mathcal{T},r) = \sum_{e\in E} w(e)\cdot \mathrm{AP}(\mathcal{T}(e),r)^{r-1}\otimes \mathrm{Pol}_{S\setminus S(e)}\ \text{, and}$$

$$\mathrm{IP}_{\mathrm{PD}}(\mathcal{T},r) = \sum_{(e,\ell)\in \mathrm{IPairs}(E)} (w(e)\cdot \mathrm{Pol}^{r-1}_{S(e)})\otimes (w(\ell)\cdot \mathrm{Pol}^{r-1}_{S(\ell)})\otimes \mathrm{Pol}_{S\setminus(S(e)\cup S(\ell))}\ . \tag{40}$$

For convenience, we define $\mathrm{EP}(\mathcal{T},r) = \mathrm{AP}_{\mathrm{PD}}(\mathcal{T}^2,r) + 2\cdot\mathrm{OP}_{\mathrm{PD}}(\mathcal{T},r) + 2\cdot\mathrm{IP}_{\mathrm{PD}}(\mathcal{T},r)$. The k-th coefficient of $\mathrm{OP}_{\mathrm{PD}}(\mathcal{T},r)$ is equal to:

$$\mathrm{CF}(\mathrm{OP}_{\mathrm{PD}}(\mathcal{T},r),k) = \sum_{e\in E}\mathrm{CF}((w(e)\cdot \mathrm{AP}(\mathcal{T}(e),r)^k)\otimes \mathrm{Pol}_{S\setminus S(e)},k) =$$

$$\mathrm{CF}(\mathrm{OP}_{\mathrm{PD}}(\mathcal{T},k),k) + \sum_{e\in E}\mathrm{CF}(w(e)\cdot \mathrm{AP}(\mathcal{T}(e),r),k)\cdot \mathrm{CF}(S\setminus S(e),0)\ . \tag{41}$$

The k-th coefficient of $\mathrm{IP}_{\mathrm{PD}}(\mathcal{T},r)$ is equal to:

$$\mathrm{CF}(\mathrm{IP}_{\mathrm{PD}}(\mathcal{T},r),k) = \sum_{(e,\ell)\in \mathrm{IPairs}(E)}\mathrm{CF}((w(e)\cdot \mathrm{Pol}^{k}_{S(e)})\otimes (w(\ell)\cdot \mathrm{Pol}^{k}_{S(\ell)})\otimes \mathrm{Pol}_{S\setminus(S(e)\cup S(\ell))},k)\ . \tag{42}$$

In the last expression, from property (19) we get that:

$$\mathrm{CF}((w(e)\cdot \mathrm{Pol}^{k}_{S(e)})\otimes (w(\ell)\cdot \mathrm{Pol}^{k}_{S(\ell)})\otimes \mathrm{Pol}_{S\setminus(S(e)\cup S(\ell))},k) =$$

$$\sum_{i=1}^{k}\sum_{j=1}^{k}\mathrm{CF}(w(e)\cdot \mathrm{Pol}_{S(e)},i)\cdot \mathrm{CF}(w(e)\cdot \mathrm{Pol}^{k}_{S(\ell)},j)\sum_{\substack{h=k-i-j\\ h\geq 0}}\mathrm{CF}(S\setminus(S(e)\cup S(\ell)),h) =$$

$$\sum_{i=1}^{k-1}\sum_{j=1}^{k-1}\mathrm{CF}(w(e)\cdot \mathrm{Pol}_{S(e)},i)\cdot \mathrm{CF}(w(e)\cdot \mathrm{Pol}^{k}_{S(\ell)},j)\sum_{\substack{h=k-i-j\\ h\geq 0}}\mathrm{CF}(S\setminus(S(e)\cup S(\ell)),h) =$$

$$\mathrm{CF}((w(e)\cdot \mathrm{Pol}^{k-1}_{S(e)})\otimes (w(\ell)\cdot \mathrm{Pol}^{k-1}_{S(\ell)})\otimes \mathrm{Pol}_{S\setminus(S(e)\cup S(\ell))},k) = \mathrm{CF}(\mathrm{IP}_{\mathrm{PD}}(\mathcal{T},k),k)\ . \tag{43}$$

Therefore, the k-th coefficients of polynomials $\mathrm{IP}_{\mathrm{PD}}(\mathcal{T},r)$ and $\mathrm{IP}_{\mathrm{PD}}(\mathcal{T},k)$ are equal. Suppose we have already computed polynomials $\mathrm{AP}_{\mathrm{PD}}(\mathcal{T},r)$, $\mathrm{OP}_{\mathrm{PD}}(\mathcal{T},r)$, and $\mathrm{IP}_{\mathrm{PD}}(\mathcal{T},r)$. For a non-negative integer $k<r$ we get:

$$\mathbb{E}_k[\mathrm{PD}^2(\mathcal{T},R)] =$$

$$\frac{1}{C_k}(\mathrm{CF}(\mathrm{AP}_{\mathrm{PD}}(\mathcal{T}^2,r),k) - \mathrm{VA}(\mathcal{T}^2)[k] + 2\cdot \mathrm{CF}(\mathrm{OP}_{\mathrm{PD}}(\mathcal{T},r),k)-$$

$$2\sum_{e\in E}\mathrm{CF}(w(e)\cdot \mathrm{AP}(\mathcal{T}(e),k),k)\cdot \mathrm{CF}(S\setminus S(e),0) + 2\cdot \mathrm{CF}(\mathrm{IP}_{\mathrm{PD}}(\mathcal{T},r),k)) =$$

$$\frac{C_r}{C_k}(\mathrm{CF}(\mathrm{EP}(\mathcal{T},r),k) - \mathrm{VA}(\mathcal{T}^2) - 2\sum_{e\in E} w(e)\cdot \mathrm{CF}(\mathrm{AP}(\mathcal{T}(e),k),k)\cdot \mathrm{CF}(S\setminus S(e),0))\ . \tag{44}$$

Let $\mathrm{VI}(\mathcal{T})$ be a vector of $r+1$ coordinates such that the i-th coordinate is equal to:

$$\mathrm{VI}(\mathcal{T})[i] = 2\sum_{e\in E}\mathrm{CF}(w(e)\cdot \mathrm{AP}(\mathcal{T}(e),i)\cdot \mathrm{CF}(S\setminus S(e),0) \tag{45}$$

Similar to $\mathrm{VA}(\mathcal{T})$, it is easy to show that we can compute $\mathrm{VI}(\mathcal{T})$ in $O((d+\log n)n\log n)$ time and $O(n)$ space. From Equation 44, we conclude that given vectorgs $\mathrm{VA}(\mathcal{T})$ and $\mathrm{VI}(\mathcal{T})$, and polynomials $\mathrm{AP}_{\mathrm{PD}}(\mathcal{T}^2,r)$, $\mathrm{OP}_{\mathrm{PD}}(\mathcal{T},r)$, $\mathrm{IP}_{\mathrm{PD}}(\mathcal{T},r)$, we can compute $\mathbb{E}_k[\mathrm{PD}^2(\mathcal{T},R)]$ and therefore the variance of PD for all leaf subset sizes $k\le r$ in $O(r)$ time in total. This concludes the theorem for the PD.

The moments of MPD**.** The contribution function of the MPD is:

$$c_{\mathrm{MPD}}(\alpha,\beta) = \begin{cases} 2\frac{\alpha(\beta-\alpha)}{\beta(\beta-1)} & \text{if } 0<\alpha<\beta. \\ 0 & \text{otherwise.}\end{cases}$$

Let N_i denote the vector of dimension i whose j-th element is equal to j. Let N_i^2 denote the vector of the same dimension whose j-th element is equal to j^2. By plugging c_{MPD} to (7)- (9), we get that the mean of MPD for subset size r is equal to:

$$\frac{1}{C_r}\mathrm{CF}(\mathrm{AP}_{\mathrm{MPD}}(\mathcal{T},r),r) = \frac{1}{C_r}\mathrm{CF}(\sum_{e\in E}(w(e)\cdot c_{\mathrm{MPD}}(e,r)\odot \mathrm{Pol}_{S(e)})\otimes \mathrm{Pol}_{S\setminus S(e)},r) =$$

$$\frac{1}{C_r}\frac{2}{r(r-1)}\sum_{e\in E}\mathrm{CF}((w(e)\cdot((r\cdot \mathrm{N}_{r-1})+\mathrm{N}_{r-1}^2)\odot \mathrm{Pol}_{S(e)}^{r-1})\otimes \mathrm{Pol}_{S\setminus S(e)},r) =$$

$$\frac{1}{C_r}\frac{2}{r(r-1)}\mathrm{CF}(\mathrm{NAP}(\mathcal{T},r),r) + \mathrm{CF}(\mathrm{RAP}(\mathcal{T},r),r)\ , \tag{46}$$

where:

$$\mathrm{NAP}(\mathcal{T},r) = \sum_{e\in E}(w(e)\cdot \mathrm{N}_{r-1}^2\odot \mathrm{Pol}_{S(e)}^{r-1})\otimes \mathrm{Pol}_{S\setminus S(e)}\ \text{ and },$$

$$\mathrm{RAP}(\mathcal{T},r) = r\cdot\sum_{e\in E}(w(e)\cdot \mathrm{N}_{r-1}\odot \mathrm{Pol}_{S(e)}^{r-1})\otimes \mathrm{Pol}_{S\setminus S(e)}\ . \tag{47}$$

As with the PD, we next sketch an approach to compute the expectation of the MPD for any $k \leq r$ using the k-th coefficients of polynomials of $\mathrm{NAP}(\mathcal{T}, r)$ and $\mathrm{RAP}(\mathcal{T}, r)$. For $k \leq r$, the k-th coefficient of polynomial $\mathrm{NAP}(\mathcal{T}, r)$ is:

$$\mathrm{CF}(\mathrm{NAP}(\mathcal{T}, r), k) = \sum_{e \in E} \mathrm{CF}((w(e) \cdot \mathrm{N}_k^2 \odot \mathrm{Pol}_{S(e)}^k) \otimes \mathrm{Pol}_{S \setminus S(e)}, k) =$$

$$\sum_{e \in E} \mathrm{CF}((w(e) \cdot \mathrm{N}_{k-1}^2 \odot \mathrm{Pol}_{S(e)}^{k-1}) \otimes \mathrm{Pol}_{S \setminus S(e)}, k) + w(e) \cdot k^2 \cdot \mathrm{CF}(S(e), k) \cdot \mathrm{CF}(S \setminus S(e), 0) =$$

$$\mathrm{CF}(\mathrm{NAP}(\mathcal{T}, k), k) + \mathrm{NVA}(\mathcal{T})[k] \ , \tag{48}$$

where $\mathrm{NVA}(\mathcal{T})$ is an vector of dimension n, where the i-th coordinate is:

$$\mathrm{NVA}(\mathcal{T})[i] = w(e) \cdot i^2 \cdot \mathrm{CF}(S(e), i) \cdot \mathrm{CF}(S \setminus S(e), 0) \ . \tag{49}$$

Similarly, for the k-th coefficient of $\mathrm{RAP}(\mathcal{T}, r)$ we get:

$$\mathrm{CF}(\mathrm{RAP}(\mathcal{T}, r), k) = \mathrm{CF}(\mathrm{RAP}(\mathcal{T}, k), k) + r \cdot \mathrm{RVA}(\mathcal{T})[k] \ , \tag{50}$$

where $\mathrm{RVA}(\mathcal{T})$ is a vector of the same dimension as $\mathrm{NVA}(\mathcal{T})[k]$, where the i-th coordinate is:

$$\mathrm{RVA}(\mathcal{T})[i] = \cdot w(e) \cdot i \cdot \mathrm{CF}(S(e), i) \cdot \mathrm{CF}(S \setminus S(e), 0) \ . \tag{51}$$

With the arguments that we used in our analyses for the PD, we can easily show that polynomials $\mathrm{NAP}(\mathcal{T}, r)$, $\mathrm{RAP}(\mathcal{T}, r)$ and vectors $\mathrm{NVA}(\mathcal{T})$ and $\mathrm{RVA}(\mathcal{T})$ can be computed in a similar way as $\mathrm{AP}_{\mathrm{PD}}(\mathcal{T}, r)$ and $\mathrm{VA}(\mathcal{T})$, in $O((d + \log n) n \log n)$ time and using $O(n)$ space. Using Equation (46) together with (47)-(51) we infer that we can compute the mean of MPD for all subset sizes $k \leq r$ with asymptotically the same running-time and space.

For the variance of MPD, we need again to compute the mean of the square of the measure. We avoid to describe in detail the tedious derivation of polynomial expressions, so we provide directly the following conclusion:

$$\mathbb{E}_r[\mathrm{MPD}^2(\mathcal{T}, r)] = \frac{4}{C_r \cdot r^2 (r-1)^2} (\mathrm{CF}(\mathrm{OP}_1(\mathcal{T}, r), r) + \mathrm{CF}(\mathrm{IP}_1(\mathcal{T}, r), r) + \mathrm{CF}(\mathrm{OP}_2(\mathcal{T}, r), r) +$$

$$\mathrm{CF}(\mathrm{IP}_2(\mathcal{T}, r), r) - \mathrm{CF}(\mathrm{OP}_3(\mathcal{T}, r), r) - \mathrm{CF}(\mathrm{IP}_3(\mathcal{T}, r), r)) \ , \tag{52}$$

where:

$$\begin{aligned}
\mathrm{OP}_1(\mathcal{T},r) &= 2r^2\sum_{e\in E}\sum_{\ell\in \mathrm{Off}(e)}(w(e)\cdot N_{r-1}\odot \mathrm{Pol}_{S(e)})\otimes(w(\ell)\cdot N_{r-1}\odot \mathrm{Pol}_{S(\ell)})\otimes \mathrm{Pol}_{S\setminus S(e)}+\\
&\qquad r^2\sum_{e\in E}(w(e)^2\cdot N_{r-1}^2\odot \mathrm{Pol}_{S(e)})\otimes \mathrm{Pol}_{S\setminus S(e)}\ ,\\
\mathrm{IP}_1(\mathcal{T},r) &= 2r^2\sum_{e\in E}\sum_{\ell\in \mathrm{Ind}(e)}(w(e)\cdot N_{r-1}\odot \mathrm{Pol}_{S(e)})\otimes(w(\ell)\cdot N_{r-1}\odot \mathrm{Pol}_{S(\ell)})\otimes \mathrm{Pol}_{S\setminus(S(e)\cup S(\ell))}\ ,\\
\mathrm{OP}_2(\mathcal{T},r) &= 2\sum_{e\in E}\sum_{\ell\in \mathrm{Off}(e)}(w(e)\cdot N_{r-1}^2\odot \mathrm{Pol}_{S(e)})\otimes(w(\ell)\cdot N_{r-1}^2\odot \mathrm{Pol}_{S(\ell)})\otimes \mathrm{Pol}_{S\setminus S(e)}+\\
&\qquad \sum_{e\in E}(w(e)^2\cdot N_{r-1}^4\odot \mathrm{Pol}_{S(e)})\otimes \mathrm{Pol}_{S\setminus S(e)}\ ,\\
\mathrm{IP}_2(\mathcal{T},r) &= 2\sum_{e\in E}\sum_{\ell\in \mathrm{Ind}(e)}(w(e)\cdot N_{r-1}^2\odot \mathrm{Pol}_{S(e)})\otimes(w(\ell)\cdot N_{r-1}^2\odot \mathrm{Pol}_{S(\ell)})\otimes \mathrm{Pol}_{S\setminus(S(e)\cup S(\ell))}\ ,\\
\mathrm{OP}_3(\mathcal{T},r) &= 2r\sum_{e\in E}\sum_{\ell\in \mathrm{Off}(e)}[(w(e)\cdot N_{r-1}\odot \mathrm{Pol}_{S(e)})\otimes(w(\ell)\cdot N_{r-1}^2\odot \mathrm{Pol}_{S(\ell)})\otimes \mathrm{Pol}_{S\setminus S(e)}]+\\
&\qquad\qquad [(w(e)\cdot N_{r-1}^2\odot \mathrm{Pol}_{S(e)})\otimes(w(\ell)\cdot N_{r-1}\odot \mathrm{Pol}_{S(\ell)})\otimes \mathrm{Pol}_{S\setminus S(e)}]+\\
&\qquad 2r\sum_{e\in E}(w(e)^2\cdot N_{r-1}^3\odot \mathrm{Pol}_{S(e)})\otimes \mathrm{Pol}_{S\setminus S(e)}\ \textit{and,}\\
\mathrm{IP}_3(\mathcal{T},r) &= 2r\sum_{e\in E}\sum_{\ell\in \mathrm{Ind}(e)}[(w(e)\cdot N_{r-1}\odot \mathrm{Pol}_{S(e)})\otimes(w(\ell)\cdot N_{r-1}^2\odot \mathrm{Pol}_{S(\ell)})\otimes \mathrm{Pol}_{S\setminus(S(e)\cup S(\ell))}]+\\
&\qquad\qquad [(w(e)\cdot N_{r-1}^2\odot \mathrm{Pol}_{S(e)})\otimes(w(\ell)\cdot N_{r-1}\odot \mathrm{Pol}_{S(\ell)})\otimes \mathrm{Pol}_{S\setminus(S(e)\cup S(\ell))}]\ .
\end{aligned} \tag{53}$$

Let $k < r$ be non-negative integer. For the k-th coefficient of each of the above polynomials we get the following:

$$
\begin{aligned}
\mathrm{CF}(\mathrm{OP}_1(\mathcal{T},r),k) &= \frac{r^2}{k^2}(\mathrm{CF}(\mathrm{OP}_1(\mathcal{T},k),k) + \mathrm{VA}_1(\mathcal{T})[k])\ , \\
\mathrm{CF}(\mathrm{IP}_1(\mathcal{T},r),k) &= \frac{r^2}{k^2}\mathrm{CF}(\mathrm{IP}_1(\mathcal{T},k),k)\ , \\
\mathrm{CF}(\mathrm{OP}_2(\mathcal{T},r),k) &= \mathrm{CF}(\mathrm{OP}_2(\mathcal{T},k),k) + \mathrm{VA}_2(\mathcal{T})[k]\ , \\
\mathrm{CF}(\mathrm{IP}_2(\mathcal{T},r),k) &= \mathrm{CF}(\mathrm{IP}_2(\mathcal{T},k),k)\ , \\
\mathrm{CF}(\mathrm{OP}_3(\mathcal{T},r),k) &= \frac{r}{k}(\mathrm{CF}(\mathrm{OP}_3(\mathcal{T},k),k) + \mathrm{VA}_3(\mathcal{T})[k]) \ \text{ and },\\
\mathrm{CF}(\mathrm{IP}_3(\mathcal{T},r),k) &= \frac{r}{k}\mathrm{CF}(\mathrm{IP}_4(\mathcal{T},k),k)\ , \qquad (54)
\end{aligned}
$$

where $\mathrm{VA}_1(\mathcal{T})$, $\mathrm{VA}_2(\mathcal{T})$, and $\mathrm{VA}_3(\mathcal{T})$ are n-dimensional vectors with elements:

$$
\begin{aligned}
\mathrm{VA}_1(\mathcal{T})[i] = \sum_e & 2\cdot w(e)\cdot i\cdot \mathrm{CF}(\mathrm{RAP}(e,i),i)\cdot \mathrm{CF}(S\setminus S(e),0) + \\
& w^2(e)\cdot i^2\cdot \mathrm{CF}(S(e),i)\cdot \mathrm{CF}(S\setminus S(e),0)\ , \\
\mathrm{VA}_2(\mathcal{T})[i] = \sum_e & 2\cdot w(e)\cdot i^2\cdot \mathrm{CF}(\mathrm{NAP}(e,i),i)\cdot \mathrm{CF}(S\setminus S(e),0) + \\
& w^2(e)\cdot i^4\cdot \mathrm{CF}(S(e),i)\cdot \mathrm{CF}(S\setminus S(e),0) \ \text{ and },\\
\mathrm{VA}_3(\mathcal{T})[i] = 2\sum_e & w(e)\cdot i\cdot \mathrm{CF}(\mathrm{NAP}(e,i),i)\cdot \mathrm{CF}(S\setminus S(e),0) + \\
& w(e)\cdot i^2\cdot \mathrm{CF}(\mathrm{RAP}(e,i),i)\cdot \mathrm{CF}(S\setminus S(e),0) + \\
& w^2(e)\cdot i^3\cdot \mathrm{CF}(S(e),i)\cdot \mathrm{CF}(S\setminus S(e),0)\ . \qquad (55)
\end{aligned}
$$

All the above polynomials and vectors can be computed in $O((d+\log n)n\log n)$ time and $O(n)$ space using similar methods to the ones we described for computing $\mathrm{IP}(\mathcal{T},r)$, $\mathrm{OP}(\mathcal{T},r)$ and $\mathrm{VI}(\mathcal{T})$. Combining (52) with expressions (53)-(55), we calculate the values for the square of MPD for all subset sizes $k < r$. □

The bounds for the time complexities in lemmas 3.3 and 3.4, and in Theorem 3.5 are achieved by using the FFT for multiplying polynomials. However, in practice FFT is numerically unstable [20, 18]. For this reason, one may prefer to use another polynomial multiplication method as a building block for our algorithms; for example, Karatsuba's method which requires $O(k^{1.59})$ time for multiplying two polynomials of maximum degree k [13], or the naive $O(k^2)$ multiplication method. Thus, we can adjust the above results, and redefine the time complexity of our algorithms based on the polynomial multiplication method that we choose to use. We get the next theorem.

Theorem 3.6. *Let $\mathcal{T}$ be a phylogenetic tree that has n nodes and depth d. Let $A(k)$ be the time complexity of a polynomial multiplication algorithm when applied on two polynomials of maximum degree k. For a single leaf-subset size, we can compute the mean and variance of an edge-decomposable measure f according to the* RPB *distribution in $O((d + \log n)A(n))$ time, using $O(n)$ space. We can compute the mean and the variance of the* PD *and* MPD *on $\mathcal{T}$ in the* RPB *model for all possible leaf-subset sizes in $O((d + \log n)A(n))$ time in total. The space required for these computations is $O(n)$.*

4 Implementations and Experiments

Based on our theoretical results, we implemented algorithms that compute in a batched manner the mean and variance of the PD and MPD. More specifically, for each of these two measures we implemented two algorithms; the first algorithm for each measure computes the mean and the variance for all leaf-subset sizes in $O((d + \log n)n \log n)$ operations and uses the FFT. The second algorithm implemented for each measure computes the same output while performing multiplications of polynomials in a naive manner; the multiplication subroutine that we use here takes $O(k^2)$ time to compute the product of two polynomials of maximum degree k. Therefore, and according to Theorem 3.6, the running time of the latter implementation is $O((d + \log n)n^2)$. We refer to the implementations that make use of the FFT as `fft_pd` and `fft_mpd`. We call these the FFT-*based* implementations. We refer to the other two implementations as `naive_pd` and `naive_mpd`. We call those two the *naive-based* implementations. All described implementations were developed in C++, and for each implementation we also developed an interface to R. For our implementations we used arithmetic with improved precision. We did this because the coefficients of the polynomials handled during the experiments could get very large or very small values, which could not be represented by standard double-precision arithmetic.

We conducted experiments using our four implementations, and measured experimentally how they perform in practice. For our experiments, we used two different phylogenetic tree datasets. The first dataset is a phylogenetic tree that represents the evolution history between all mammal species [1]. This tree has 4,510 leaf nodes, 6,618 nodes in total, and its depth is 39. To speed up the performance of our algorithms, we converted this tree into a binary one by adding extra interior nodes and edges of zero weight. This preprocessing step can be performed in $O(n)$ time, and maintains that the new tree has the same mean and variance as the initial one for any edge-decomposable measure and any leaf subset-size. The resulting binary tree has 9,019 nodes in total, and its depth is 42. We refer to the converted tree as `mammals`. The second dataset is a tree that represents the phylogeny of eukaryotic organisms [11]. It has 71,181 leaves and 83,751 nodes in total. This tree is unrooted, so we picked arbitrarily an internal node and used that as the root. The depth of the rooted tree is 35. We converted this tree into a binary one, and the resulting dataset has 142,361 nodes and depth 86. We refer to the latter tree as `eukaryotes`.

We performed three sets of experiments; in the first set, we compare the performance between the FFT-based and the naive-based implementations. In the second set of experiments we examine how the fastest of the implementations perform on our largest dataset, after boosting their design with parallel processing. Then, in the third set of experiments we show how our implementations perform compared to an inexact sampling method. All experiments were performed on a 64-bit computer with an Intel i7-3770 CPU. This CPU consists physically of four cores where each core is a 3.40 GHz processor. But, due to the use of hyper-threading, this CPU can imitate efficiently operations on eight virtual cores, each with 3.40 GHz capacity. The main memory of the computer is 16 Gigabytes, and the operating system installed on this computer is Ubuntu version 14.04.

In the first set of experiments, we measured the running time of all four implementations on subtrees that we extracted from `mammals`. We extracted eighteen trees from `mammals` such that each tree consisted of $250k + 260$ leaves, with k ranging from zero to seventeen. For each extracted tree of x leaves, we ran our four implementations and computed the mean and the variance for all leaf-subset sizes in the range from one to x. We chose the `mammals` dataset for this experiment because its size is closer to the one usually encountered in ecological case studies.

The running times that we measured for this experiment are illustrated in Figure 1.

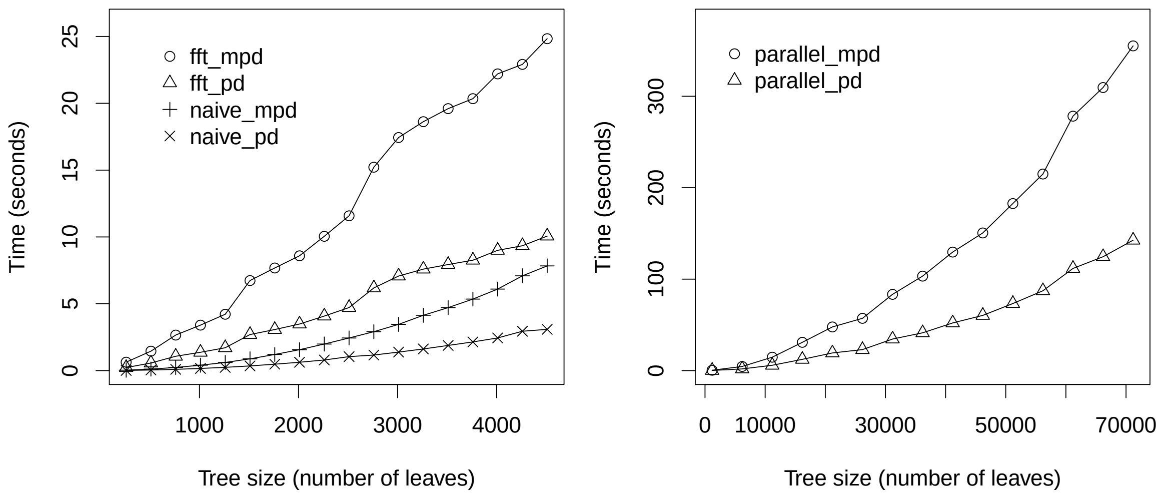

Figure 1: **Left:** the measured running times of the FFT-based and naive-based implementations on trees extracted from `mammals` dataset. **Right:** the running times of the two parallelized implementations on trees extracted from `eukaryotes`.

We see that the naive-based implementations outperform the FFT-based ones. Also, the implementations for the PD are faster than the ones for MPD (this is expected since for the MPD computations we need to construct more polynomials than for PD). For the largest tree that we used (the complete `mammals` tree with 4,510 leaves) implementation `fft_pd` took 10.6 seconds to compute the required moments, while `naive_pd` took 3.08. For the same tree, program `fft_mpd` took 24.83 seconds and `naive_mpd` took 7.83 seconds. To showcase the advantages of our implementations, we compared the running times of our algorithms with the performance of an inexact method. More specifically, for a given subset-size r we select at random from `mammals` one thousand leaf-subsets of this size. For this we used function `sample` in software platform R; this function samples a subset from a weighted set of elements by sequentially picking an element t with probability $p(t)$ divided by the sum of probabilities of all remaining items. For each sample, the PD and MPD values where computed using the functions of package `PhyloMeasures`, and the mean and the variance of each measure was calculated from these values. We refer to this inexact method as the *heuristic*. The heuristic method took more than an hour to execute for this tree, both for the PD and the MPD. Even for the smallest tree that we considered in this experiment (260 nodes), the heuristic method took 19.22 and 26.92 seconds for the PD and MPD respectively.

This difference between the naive-based and FFT-based programs seems to contradict the theoretical complexity of the corresponding algorithms. One explanation is that the FFT has a larger constant hidden in its asymptotical time complexity than the naive multiplication algorithm. It is the case that the gap in performance becomes smaller as the tree size increases. Even so, preliminary experiments showed that also for much larger trees (such as the complete `eukaryotes`, with $> 70{,}000$ leaves) the FFT-based algorithms were slower.

Also, as already mentioned, the FFT is a numerically unstable method. The robustness of this method depends on the precision of evaluating trigonometric functions, to generate the so-called complex roots of unity (also known as "twindle" factors) [20, 18]. As a result, for our FFT-based implementations and for all the datasets that we considered, almost the entire output was wrong. In the first stages of development, we experimented with different available implementations of the FFT ([10, 15]), and with an implementation of our own that allowed trigonometric calculations with high precision floats [9]. Despite all these attempts, it was not possible to get rid of the numerical errors in the output of the FFT-based implementations. For instance, when computing the PD variance for `mammals`, the relative error in the variance values computed by the FFT-based methods is for most coefficients larger than 10^{200}, while the absolute error is close to 10^{-14}. This is probably because of the following reason. We observed that the correct coefficients of the polynomials that we want to compute occupy a large range of values; from 10^{-1937} to roughly 500 for `mammals`, with most coefficients having a value smaller than 10^{-200}. On the contrary, in the FFT-based methods, the coefficients computed for `mammals` fall entirely within the range from 10^{-17} to approximately 500, with the vast majority appearing in the range 10^{-12} to 10^{-15}. In the best case, even for state-of-the-art FFT libraries, the absolute error observed in FFT computations is somewhere in the range between 10^{-15} and 10^{-17} [8]. This error that appears from using the FFT is apparently a major obstacle for the exact computation, at least for the smallest values. For this reason, and because of the difference in performance, we focused on improving the efficiency of the naive-based implementations. We describe this in more detail with the next set of experiments.

In the second set of experiments, we boosted the naive-based implementations by introducing parallelism. We did this by re-designing the naive operator that performs polynomial multiplications. More specifically, let P_α and P_β be two polynomials of maximum degree m, and let g denote the number of available processors. When computing the product $P_\gamma = P_\alpha \otimes P_\beta$, we split the computation of the at most $2m$ coefficients of P_γ into $2m/g$ groups, and fed each group to a different processor. This simple adjustment leads to an algorithm whose time complexity is $O((d + \log n)(n^2/g + n \log n))$. We made this adjustment to both of our naive-based methods. We refer to these adjusted implementations as `parallel_pd` and `parallel_mpd`. We evaluated the running time of the parallelized implementations on trees extracted from the `eukaryotes` dataset. These trees consist of $5000k + 1181$ leaves, with k ranging from zero to fourteen. For these experiments, at any point during the executions, the maximum number of active parallel threads was set to eight, the number of available processors on our computer. The results of the experiments appear in Figure 1.

We see that both implementations perform very fast even for datasets with many thousands of leaves. For the complete `eukaryotes` tree, `parallel_pd` took 143 seconds to execute, and `parallel_mpd` took 355 seconds. Recall that, during these executions, each program calculated the mean and the variance for as many leaf-subset sizes as the number of leaves in the tree. We conducted experiments with different numbers of parallel threads. Yet, our programs performed faster when we set the number of active threads to the total number of available processors. Using more threads than processors can only provide better results if, on average, each thread remains idle for a considerable amount of time due to cache misses. However, this was not the case in our experiments; when using eight threads, during most of the execution we observed a CPU utilization which was close to 90% for each core .

In the third set of experiments we demonstrate how our implementations can speedup the computations for standard case studies in Ecology. To do this, we used the `mammals` tree and a dataset that represents mammal communities around the world. The community dataset that we used is a presence-absence matrix which is structured as follows; each column of the matrix corresponds to a mammal species, and each row corresponds to a geographical region. Each entry M_{ij} in the matrix has value one or zero, based on whether the i-th species resides in the region represented by the j-th row.

To construct the matrix, we produced a raster of the world with a resolution of 193km. Then, we used polygons that represent the geographical areas where mammal species reside. These polygons were acquired from the International Union for Conservation of Nature (IUCN) [12]. We overlayed the polygons on the world raster, and extracted for each cell the community of species whose polygons overlap with the cell. We then represented each extracted community as a separate row in the presence-absence matrix. We refer to the resulting matrix as `communities`. The `communities` matrix consists of 4,971 rows and 4,173 columns. The number of columns in this matrix is smaller than the number of leaves in `mammals` because we excluded marine mammals, and because data was absent for a few of the other species. To each species v in `mammals` we assigned a probability value $p(v)$ equal to the sum of entries in the corresponding column of `communities`, divided by the total number of rows in the matrix. The species that were not represented by a column in `communities` were assigned a probability value equal to zero.

For each species set R in `communities` we used our implementations to compute an index based on the MPD. This index is called the reverse-Net Relatedness Index (rNRI), the reverse of the NRI index [24]. For a species set R of r species this is equal to:

$$\text{rNRI}(\mathcal{T}, R) = \frac{\text{MPD}(\mathcal{T}, R) - \mu_r(\mathcal{T})}{\sqrt{\text{var}_r(\mathcal{T})}} ,$$

where $\mathcal{T}$ is the phylogenetic tree, $\mu_r(\text{MPD}, \mathcal{T})$ is the mean value of MPD among all leaf subsets of r elements, and $\text{var}_r(\text{MPD}, \mathcal{T})$ is the variance of MPD among these subsets. The rNRI values were computed using the interface of our implementations in R. To calculate $\text{MPD}(\mathcal{T}, R)$ for a single subset R we used the efficient implementation of this measure that appears in the R package `PhyloMeasures`. Using `parallel_mpd`, the time taken to compute all the rNRI values is 2.6 seconds. The time taken to compute these values with `naive_mpd` is 8.5 seconds. Figure 2 shows the world grid colored according to the computed rNRI values.

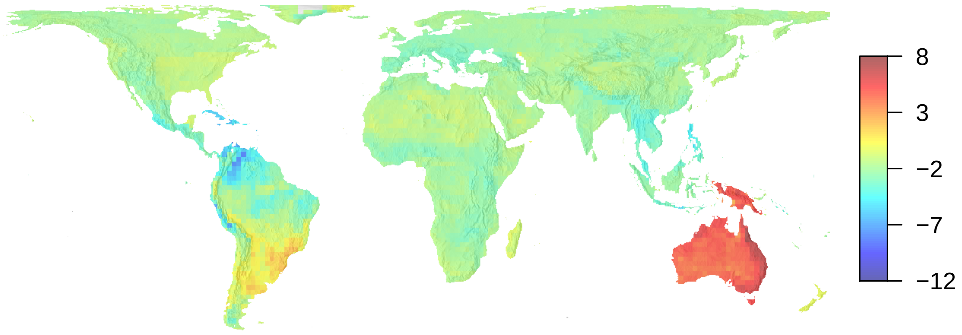

Figure 2: The world raster of 193 km resolution, colored according to the rNRI values of mammal communities. Red regions represent areas of higher relative phylogenetic biodiversity, while blue regions indicate lower diversity.

Observe that Australia shows remarkably high rNRI values, due to mixing of many marsupial and placental mammal lineages, leading to high pairwise distances. In contrast, northern South America shows particularly low rNRI values, indicating a concentration of closely related lineages.

Using the heuristic method, it took more than 65 minutes and 58 seconds to compute the rNRI values for all the species sets in `communities`. Comparing this with the performance of our methods, we conclude that our algorithms provide a huge speedup for standard applications in Ecology. This allows to process much larger datasets than it was possible before. We intend to incorporate our parallelized implementations to the R package `PhyloMeasures`.

5 Future Work

In this paper, we presented efficient algorithms for computing the weighted moments of a well-defined category of phylogenetic measures. The random model that we considered is similar to the Poisson binomial distribution restricted to leaf subsets of fixed size. We also presented improved results for two of the most popular measures, namely the PD and MPD. An interesting problem would be to derive improved algorithms for other phylogenetic measures, as well as for different random models. Another possible challenge would be to derive more efficient algorithms for the specific case where the probability values at the leaf nodes of the tree are distributed according to a specific pattern, for example the so-called dust bunny model [3].

References

[1] O.R.P. Bininda-Emonds, M. Cardillo, K.E. Jones, R.D.E MacPhee, R.M.D. Beck, R. Grenyer, S.A. Price, R.A. Vos, J.L. Gittleman and A. Purvis. The Delayed Rise of Present-Day Mammals. *Nature*, 446: 507–512, 2007.

[2] S.X. Chen and J.S. Liu. Statistical Applications of the Poisson-Binomial and Conditional Bernoulli Distributions. *Statistica Sinica*, 7:875–892, 1997.

[3] B. McCune, H.T. Root. Origin of the Dust Bunny Distribution in Ecological Community Data. *Plant Ecology*, 216(5):645–656, 2015.

[4] V. Devictor, D. Mouillot, C. Meynard, F. Jiguet, , W.Thuiller and N. Mouquet. Spatial Mismatch and Congruence between Taxonomic, Phylogenetic and Functional Diversity: the Need for Integrative Conservation Strategies in a Changing World. *Ecology Letters*, 13:1030–1040, 2010.

[5] D.P. Faith. Conservation Evaluation and Phylogenetic Diversity. *Biol. Conserv.*, 61:1–10, 1992.

[6] B. Faller, F. Pardi and M. Steel. Distribution of Phylogenetic Diversity Under Random Extinction. *Journal of Theoretical Biology*, 251: 286–296, 2008.

[7] M.Fernández and S. Williams. Closed-Form Expression for the Poisson-Binomial Probability Density Function. *IEEE Transactions On Aerospace And Electronic Systems*, 46(2): 803–817, 2010.

[8] Benchmark Page for Numerical Errors for Several FFT Libraries. `http://www.fftw.org/accuracy/method.html`

[9] L. Fousse, G. Hanrot, V. Lefèvre, P. Pélissier, and P. Zimmermann. MPFR: A Multiple-Precision Binary Floating-Point Library with Correct Rounding. *ACM Transactions on Mathematical Software*, 33(2):13, 2007.

[10] M. Frigo and S.G. Johnson. FFTW: An Adaptive Software Architecture for the FFT. In *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 3, pages 1381–1384, 1998.

[11] P.A. Goloboff, S.A. Catalano, J.M. Mirandeb, C.A. Szumika, J.S. Ariasa, M. Kallersjoc and J.S. Farris. Phylogenetic Analysis of 73 060 Taxa Corroborates Major Eukaryotic Groups. *Cladistics*, 25:211–230, 2009.

[12] The Webpage of the International Union for Conservation of Nature. `http://www.iucn.org/` .

[13] A. Karatsuba and Y. Ofman. Multiplication of Many-Digital Numbers by Automatic Computers. *Proceedings of the USSR Academy of Sciences*, 145: 293–294, 1962.

[14] S.W. Kembel. Disentangling Niche and Neutral Influences on Community Assembly: Assessing the Performance of Community Phylogenetic Structure Tests. *Ecology Letters*, 12:949–960, 2009.

[15] The Kiss FFT Software Library. `http://sourceforge.net/projects/kissfft/` .

[16] N.J.B. Kraft, W.K. Cornwell, C.O. Webb and D.A. Ackerly. Trait Evolution, Community Assembly, and the Phylogenetic Structure of Ecological Communities. *The American Naturalist*, 170:271–283, 2007.

[17] C. Van Loan. *Computational Frameworks for the Fast Fourier Transform*, Vol. 10, Siam, 1992.

[18] J. C. Schatzman. Accuracy of The Discrete Fourier Transform and the Fast Fourier Transform. *SIAM J. Sci. Comput.*, 17(5): 1150–1166, 1996.

[19] M. Steel. Tools to Construct and Study Big Trees: A Mathematical Perspective. In T. Hodkinson, J. Parnell and S. Waldren(eds.), *Reconstructing the Tree of Life: Taxonomy and Systematics of Species Rich Taxa*, CRC Press, pages 97–112, 2007.

[20] M. Tasche and H. Zeuner. Improved Roundoff Error Analysis for Precomputed Twiddle Factors. *Journal of Computational Analysis and Applications*, 4(1):1–18, 2002.

[21] C. Tsirogiannis and B. Sandel. PhyloMeasures: a Package for Computing Phylogenetic Biodiversity Measures and their Statistical Moments. *Ecography*, http://dx.doi.org/10.1111/ecog.01814, 2015.

[22] C. Tsirogiannis, B. Sandel and D. Cheliotis. Efficient Computation of Popular Phylogenetic Tree Measures. In *Proc. 12th Workshop on Algorithms in Bioinformatics (WABI)*, pages 30 – 43, 2012.

[23] C. Tsirogiannis, B. Sandel B and A. Kalvisa. New Algorithms for Computing Phylogenetic Biodiversity. *Algorithms in Bioinformatics*, 8701:187-203, 2014.

[24] C.O. Webb, D.D. Ackerly, M.A. McPeek and M.J. Donoghue. Phylogenies and Community Ecology. *Annu. Rev. Ecol. Syst.*, 33:475–505, 2002.