PLOS COMPUTATIONAL BIOLOGY

RESEARCH ARTICLE

# Computing generalized cophenetic distances under all $L_p$ norms: A near-linear time algorithmic framework

**Paweł Górecki**[1], **Alexey Markin**[2], **Sriram Vijendran**[3], **Oliver Eulenstein**[3]*

**1** Faculty of Mathematics, Informatics and Mechanics, University of Warsaw, Warsaw, Mazowieckie, Poland, **2** Virus and Prion Research Unit, National Animal Disease Center, USDA-ARS, Ames, Iowa, United States of America, **3** Department of Computer Science, Iowa State University, Ames, Iowa, United States of America

* oeulenst@iastate.edu





**Citation:** Górecki P, Markin A, Vijendran S, Eulenstein O (2025) Computing generalized cophenetic distances under all $L_p$ norms: A near-linear time algorithmic framework. PLoS Comput Biol 21(6): e1013069. https://doi.org/10.1371/journal.pcbi.1013069

**Editor:** Joëlle Barido-Sottani, Ecole Normale Superieure, FRANCE

**Received:** February 5, 2025

**Accepted:** April 17, 2025

**Published:** June 10, 2025



**Data availability statement:** The source code and data used to produce the results and analyses presented in this manuscript are available from the GitHub repository "near-linear-cophenetic-distance": https://github.com/sriram98v/near-linear-Cophenetic-distance.

**Funding:** The support to PG was provided by National Science Centre grant #2019/33/B/ST6/00737 (https://www.ncn.gov.pl/en). This project was

## Abstract

The cophenetic distance is a well-established metric in biology used to compare pairs of trees represented in a vector format. This distance was introduced by Cardona and his co-authors, building on the foundational work of Sokal and Rohlf, which dates back over 60 years. It is widely recognized for its versatility since it can analyze trees with edge weights using various vector norms. However, when comparing large-scale trees, the quadratic runtime of the current best-known (i.e., naïve) algorithm for computing the cophenetic distance can become prohibitive. Recently, a new algorithmic framework with near-linear time complexity has been developed to calculate the distances of a generalized class of cophenetic distances, which are derived from the work of Sokal and Rohlf. This improvement not only allows the cophenetic distance to be utilized in large-scale studies but also enhances the versatility of these studies by incorporating generalized variants of the cophenetic distance. However, the framework is limited to applying only the $L_1$ and $L_2$ vector norms, which significantly restricts the versatility of generalized cophenetic distances in large-scale applications. To address this limitation, we present a near-linear time algorithmic framework for computing the generalized cophenetic distances across all $L_p$ vector norms. In our scalability study, we showcase the practical performance of our unrestricted algorithmic framework. Furthermore, we investigate the applicability of the generalized cophenetic distances by analyzing the distributions of key components of these distances under various vector norms.

## Author summary

Biological research often relies on large-scale comparisons of evolutionary trees to extract valuable insights across different subfields of biology. To effectively compare trees on a large scale, sensitive metrics are needed to assess differences in topology and branch lengths alongside efficient computational algorithms. In this study, we focus on the

funded in part by the United States Department of Agriculture (USDA), Agricultural Research Service (ARS, https://ars.usda.gov) project numbers 3022-32000-018-017-S, 5030-32000-231-095-S and with federal funds from the USDA Agricultural Research Service (5030-32000-231-095-S, and 3022-32000-018-017-S)to OE. The funding sources had no role in study design, data collection, and interpretation, or the decision to submit the work for publication. Mention of trade names or commercial products in this article is solely to provide specific information and does not imply recommendation or endorsement by the USDA. USDA is an equal opportunity provider and employer.

**Competing interests:** The authors have declared that no competing interests exist.

classic cophenetic distance, a metric that compares pairs of trees represented in vector format. The cophenetic distance can analyze trees with edge weights using various vector norms, making it highly versatile. Recently, a fast algorithm was developed to calculate generalized cophenetic distances, including the cophenetic distance itself, which has enabled large-scale studies and expanded the types of cophenetic distances applicable for tree pair comparisons. However, this algorithm is limited to $L_1$ and $L_2$ norms, which greatly reduces the versatility of the cophenetic distance. To overcome this limitation, we introduce a fast algorithm that computes the generalized cophenetic distance for all vector norms, making it suitable for large-scale studies. We also conduct a scalability study to demonstrate the effectiveness of our algorithm in practice, and we analyze the distributions of key representatives of cophenetic distances across various vector norms.

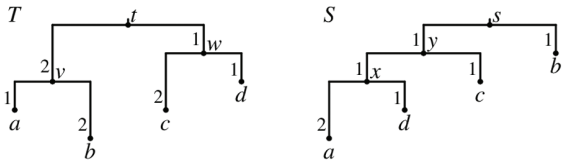
## Introduction

The cophenetic distance, introduced by Cardona et al. [1], originates from the pioneering work of Sokal and Rohlf over 60 years ago [2] and has gained substantial recognition in biology due to its reputation for reliability in analysis. This distance is a vector-based metric that is more versatile than many other commonly used tree metrics [3–5], as it can be applied to tree pairs with edge weights and analyzed using various vector norms. This capability allows for a more in-depth analysis, particularly when comparing similar tree topologies. Consequently, the cophenetic distance has broad applicability across various fields, such as phylogenetics [6–8], genomics [9], ecology [10–12], epidemiology [13,14], and conservation biology [15].

The cophenetic distance between a pair of rooted binary trees is defined based on their representation as cophenetic vectors. A *cophenetic vector* for a rooted tree assigns a value to each pair of taxa within the tree, representing the depth of their least common ancestor in that particular tree. Cophenetic vectors encode their corresponding trees equivalently [1], allowing the distance between a pair of trees to be measured as a $L_p$ norm of the vector difference of the corresponding trees. Vector norms are beneficial not only for biological data analysis [16] but also applicable in various fields, such as statistical analyses [17] and machine learning [18]. An example of various vector norms is depicted in Fig 1.

With the advent of genome-scale data, the established reputation of the cophenetic distance and its analysis using various norms has led to increased interest among researchers in utilizing this metric for large-scale biological analyses. Such analyses include phylogenetics [19], median tree estimation [20,21], comparison of evolutionary process models [22], balancing of trees and networks [6,7], studies on theoretical properties of cophenetic $L_\infty$ norm [23], and representative sampling [10]. Calculating cophenetic distances using the naïve algorithm, which has a quadratic time complexity, is slow and impractical for large datasets. Fortunately, an efficient algorithmic framework now enables near-linear time computations for a *generalized class of cophenetic distances* for the $L_1$ and $L_2$ norms [24]. This advancement makes it feasible to conduct large-scale analyses. However, despite these advancements, algorithms for calculating cophenetic distances in sub-quadratic time for other $L_p$ norms are still unknown.

In this paper, we present a near-linear time algorithmic framework for calculating the $L_p$ norms for the generalized class of cophenetic distances. We demonstrate the performance of our algorithmic framework through a scalability study. Furthermore, we analyze the distributions of key representatives from the generalized class of cophenetic distances across various $L_p$ norms. An implementation of the algorithmic framework can be found on GitHub [25].




| Contribution mapping w.=weighted | Cophenetic vectors for $T$ and $S$ $\langle aa, \underline{ab}, ac, ad, bb, bc, bd, cc, cd, dd\rangle$ $T$ | $S$ | $L_1$ | $L_2$ | $L_3$ | $L_4$ | $L_\infty$ |
|---|---|---|---|---|---|---|---|
| depth | 2 1 0 0 2 0 0 2 1 2 | 3 0 1 2 1 0 0 2 1 3 | 7 | 3.00 | 2.35 | 2.14 | 2 |
| w. depth | 3 2 0 0 4 0 0 3 1 2 | 4 0 1 2 1 0 0 2 1 3 | 11 | 4.58 | 3.61 | 3.29 | 3 |
| subtree height | 0 1 2 2 0 2 2 0 1 0 | 0 3 2 1 0 3 3 0 2 0 | 6 | 2.83 | 2.29 | 2.11 | 2 |
| w. subtree height | 0 2 4 4 0 4 4 0 2 0 | 0 4 3 2 0 4 4 0 3 0 | 6 | 3.16 | 2.62 | 2.41 | 2 |
| subtree size | 1 2 4 4 1 4 4 1 2 1 | 1 4 3 2 1 4 4 1 3 1 | 6 | 3.16 | 2.62 | 2.41 | 2 |
| lca | $a\ \underline{v}\ t\ t\ b\ t\ t\ c\ w\ d$ | $a\ x\ y\ s\ b\ s\ s\ c\ y\ d$ | | | | | |

**Fig 1. Cophenetic distances under various $L_p$ norms and contribution functions.** The vectors have values computed for the least common ancestor (lca, see the bottom row) of the corresponding taxon pairs. For example, $\mathsf{lca}_T(a, b)$ is $v$, and the second position in cophenetic vectors for $T$ reflects the contribution of $v$, that is, the depth of $v$ is 1.

https://doi.org/10.1371/journal.pcbi.1013069.g001

**Related work.** The cophenetic distance is frequently used in phylogenetics to measure the similarity between two trees; both trees are encoded as vectors and are then compared in the corresponding vector space using different norms, such as the $L_1$, $L_2$, or $L_\infty$ norm. Another popular metric that utilizes vector encodings is the path-difference distance, and both metrics can be naïvely computed in $O(pn^2)$ time for a pair of trees with $n$ taxa under the $L_p$ norm. This quadratic runtime is highly restrictive in the realm of large-scale phylogenetic analysis. [26,27] addressed the quadratic barrier for the path-difference metric under the $L_p$ norm by proposing near-linear time algorithms. For the cophenetic distances, [24] proposed a novel algorithmic framework to efficiently compute the distance between a pair of trees under the $L_1$ and $L_2$ norms in $O(n\log^2 n)$ time and $O(n\log n)$ time, respectively [24]. Furthermore, [24] demonstrated that the framework can be applied to a broad class of cophenetic metrics constructed using path-monotonic mappings, which are monotonic on the paths of the input trees. For instance, the original cophenetic distance is derived from the depth of a node, which is a path-monotonic mapping. This condition enables the development of hybrid approaches, resulting in new cophenetic costs rather than metrics, by allowing distinct path-monotonic mappings for each input tree. Such an approach may be particularly suitable when the input trees originate from different sources.

**Contribution.** We introduce a near-linear time algorithmic framework for computing the $L_p$ norms of a broad class of cophenetic distances. Our framework achieves the following time complexities: $O(pn\log n)$ for even values of $p$, $O(n\log^2 n + pn\log n)$ for odd values of $p$, and $O(n\log^2 n)$ under the $L_\infty$ norm.

Our approach begins by partitioning the input trees into four approximately equal-sized subtrees. Based on the placement of taxon pairs within these subtrees, we identify several categories of taxon pair locations. For each category, we develop a specific method to compute their contributions to the cophenetic distance. The first category requires four recursive calls on controlled (smaller) subtrees extracted from the input trees. Other categories can be computed in linear time. For the final category, the computation depends on the parity of $p$, resulting in time complexities of $O(p\log n)$ and $O(pn)$ for odd and even $p$, respectively. By

integrating these methods, we develop a unified divide-and-conquer algorithm for computing cophenetic distances under $L_p$ norms with finite $p$. Furthermore, we extend this approach to efficiently compute the $L_\infty$ norm of the cophenetic distance.

We demonstrate the efficiency of our algorithmic frameworks through experimental evaluations. We analyze the runtime of the proposed algorithm for tree pairs with varying numbers of taxa in a scalability study under multiple $L_p$ norms. Our findings indicate that our implementation of the divide-and-conquer strategy significantly outperforms the runtime of the best-known quadratic algorithm for pairs of trees with more than 400 taxa across all tested norms $p \leq 100$.

Lastly, we investigated various types of cophenetic distances under different $L_p$ norms, treating them as distributions of pairwise distances. The trees for each distribution were randomly generated using the uniform model and the Yule model [28]. Our analyses reveal that the class of cophenetic distances provides a high degree of diversity under larger $L_p$ norms, which can benefit numerous applications in comparative phylogenetics.

## Methods

This section presents algorithms for computing the $L_p$ norms of cophenetic distances. We begin with the necessary definitions and outline the method for identifying a median vertex in a rooted tree. Next, we describe how input trees are partitioned into four approximately equal subtrees using median vertices. Based on the positions of taxon pairs within these subtrees, we classify them into several categories. For each category, we propose distinct algorithms to calculate their contributions to the cophenetic distance. Finally, we introduce a unified algorithm for computing cophenetic distances for all finite norms and the $L_\infty$ norm. We also analyze and present the time and space complexities of the proposed algorithms.

### Definitions

We introduce needed notation and terminology partially following [24]. Let $T = \langle V_T, E_T \rangle$ be a rooted binary tree, and let $v$ and $w$ be vertices in $T$. The root of $T$ is denoted by $\mathsf{root}(T)$. The least common ancestor of $v$ and $w$ in $T$ is denoted by $\mathsf{lca}_T(v, w)$. We use $v \preceq w$ to denote that $w$ lies on the path between $v$ and the root of $T$. Note that $v \prec w$ means $v \preceq w$ and $v \neq w$. A vertex $v$ is called *strictly internal* in $T$, if $v$ is neither a leaf nor the root. For any non-root vertex $v$, $v.\mathsf{parent}$ and $v.\mathsf{sibling}$ denote the parent and sibling of $v$, respectively. The set of leaves in $T$ is denoted by $L_T$, and the number of leaves by $|T|$. Similarly, $L_T(v)$ denotes the set of all leaves reachable from $v$, and $|v|$ represents the size of $L_T(v)$. A *weighted* tree $T$ is a rooted binary tree with an edge weight function $w_T: E_T \to \mathbb{R}_+$.

We write that a function $\xi: V_T \to \mathbb{R}$ is *path-monotonic* if, for every $v$ and $w$ such that $v \preceq w$, either $\xi(v) \geq \xi(w)$ (descending) or for every $v$ and $w$ such that $v \preceq w$, $\xi(v) \leq \xi(w)$ (ascending). In this article, we distinguish the following three *contribution functions*, defined for unweighted trees, where for a vertex $v$ from $T$ its contribution is:

- the depth of a $v$, i.e., the number of edges on the path from $v$ to the root of $T$ [1],
- the height of the subtree rooted at $v$, defined as the maximum number of edges in any path from $v$ to a leaf in the subtree,
- the number of leaves in the subtree of $T$ rooted at $v$.

Note that the first contribution function is descending, whereas the other two are ascending.

A similar definition applies to weighted depth and height in weighted trees, but instead of counting the number of edges, we sum the weights of edges. See examples in Fig 1. Note

that our definition allows for the contribution of the root. Therefore, a tree may also include an additional rooting edge (with an associated weight), whose bottom vertex is the root, if necessary.

Let $T$ and $T'$ be trees with the same set of leaves $L(T) = L(T') = \{x_1, x_2, \ldots, x_n\}$, where the ordering of leaves is fixed. Let $\xi: V_T \to \mathbb{R}$ and $\xi': V_{T'} \to \mathbb{R}$ (both either ascending or descending) be two path-monotonic contribution functions.

The *cophenetic vector* of $T$ is defined as $\phi(T, \xi) := [\xi(\mathsf{lca}_T(x_i, x_j))]_{i \le j}$, and similarly for $T'$. The $L_p$*-cophenetic distance* with respect to the contribution functions $\xi$ and $\xi'$ is defined for $1 \le p \le \infty$ as:

$$d_p(T, T') := \|\phi(T, \xi) - \phi(T', \xi')\|_p,$$

where $\|...\|_p$ is the $L_p$ norm. That is, $d_p$ is the $L_p$ norm of the difference between the two cophenetic vectors induced by $\xi$ and $\xi'$.

Formally, in our case, if $p$ is finite,

$$d_p(T, T') = \Big( \sum_{1 \le i \le j \le n} |\xi(\mathsf{lca}_T(x_i, x_j)) - \xi'(\mathsf{lca}_{T'}(x_i, x_j))|^p \Big)^{\frac{1}{p}},$$

otherwise

$$d_\infty(T, T') = \max_{i,j} |\xi(\mathsf{lca}_T(x_i, x_j)) - \xi'(\mathsf{lca}_{T'}(x_i, x_j))|.$$

It should be clear that $d_p$ is a metric [1] as long as both $\xi$ and $\xi'$ are based on the same type of contribution functions (e.g., depth).

## Median vertex in a rooted tree

A vertex $t$ of a rooted tree $T$ divides the tree into two parts: the subtree of $T$ rooted at $t$, denoted by $T_t$ and referred to as the *lower tree* with respect to $t$, and the tree $T^t$, called the *upper tree*, which is obtained by replacing $T_t$ with a leaf. The vertex $t$, as introduced in the next lemma, is called a *median vertex* and can be computed in $O(n)$ time.

**Lemma 1** (The existence of a median vertex; [24])**.** *For every rooted binary tree $T$ of size $n \ge 2$ there is a vertex $t$ such that $\frac{n}{2} \le |T^t| \le \frac{3n}{4} + 1$ and $\frac{n}{4} \le |T_t| \le \frac{n}{2}$.*

We will present the proof for Lemma 1, as it was not included in [24].

*Proof*: Let $t$ be a vertex computed as follows: initialize $t$ as the root of $T$, and repeatedly update $t$ to its child with the largest subtree size until $|t| \le \frac{n}{2}$. We show that the resulting vertex $t$ satisfies the condition.

Let $t'$ be the sibling of $t$ and $p$ be the parent of $t$. We have $|T_t| = |t|$, $|t'| \le |t| \le \frac{n}{2}$, and $|p| = |t| + |t'| > \frac{n}{2}$. Hence, $|T^t| = n - |t| + 1 \ge \frac{n}{2} + 1$. If $\frac{n}{4} > |T_t| = |t|$ then $\frac{n}{4} > |t'|$. Thus, $|p| = |t| + |t'| < \frac{n}{2}$, which is a contradiction. Thus, $|T_t| \ge \frac{n}{4}$. Finally, $|T^t| = n - |t| + 1 \le n - \frac{n}{4} + 1 = \frac{3n}{4} + 1$. □

A median vertex is usually non-unique, e.g., a rooted tree $(a, (b, c))$ has two median vertices $b$ and $c$. Note that the concept of a median vertex is similar to the centroid of a tree, which partitions an unrooted tree into subtrees, each with a size of at most $\frac{n}{2}$. However, in our case, the median vertex divides a rooted binary tree into two subtrees, with the size of one subtree being between $\frac{n}{2}$ and $\frac{3n}{4} + 1$, as shown in Lemma 1.

## Classification of taxon pairs

In the remaining part of the article, we assume that $T$ and $T'$ are two trees that share the same fixed set of leaves. Let $t$ and $t'$ represent fixed median vertices of $T$ and $T'$, respectively. Additionally, let $\xi$ and $\xi'$ denote the contribution functions of $T$ and $T'$, respectively. Without loss of generality, we further assume that all contribution functions $\xi$ and $\xi'$ are descending. That is, for any pair of vertices $v$ and $w$ such that $v \preceq w$, it holds that $\xi(v) \geq \xi(w)$.

Given two trees $T$ and $T'$, we fix arbitrarily one median vertex in each tree. Then, the path connecting the median vertex with the root will be called a *median path*. We denote by $A$ and $B$ the sets of leaves excluding the median vertex in the upper and lower trees of $T$, respectively. Similarly, we denote $A'$ and $B'$ for $T'$. Note that $|A| + |B| = |A'| + |B'| = n$.

We now have four possible classes for a pair of leaves $\langle x, y\rangle$, where $x$ can be equal to $y$, from $T$, depending on their location:

- $AA$ — if both leaves are located in the upper tree (i.e., $x, y \in A$),
- $BB$ — if $x, y \in B$,
- $AB$ — if $x \in A$ and $y \in B$,
- $BA$ — if $x \in B$ and $y \in A$.

When considering the pair $\langle x, y\rangle$ in both trees, there are 16 possible *types* of locations, denoted in the form $XY|X'Y'$, where $X, Y \in \{A, B\}$ and $X', Y' \in \{A', B'\}$. We say that the pair $\langle x, y\rangle \in L_T \times L_{T'}$ has type $XY|X'Y'$ if $x \in X \cap X'$ and $y \in Y \cap Y'$.

Since some of the types due to symmetry represent the same sets of taxon pairs, we introduce categories for the joint representation of types as indicated in Table 1. There are 4 non-mixed categories $N_1, N_2, \ldots, N_4$ that correspond uniquely to the types $AA|A'A'$, $AA|B'B'$, $BB|A'A'$ and $BB|B'B'$. Taking symmetry into account, there are 4 single-mixed categories $S_1 - S_4$, e.g., in $S_1$ the type $AB|A'A'$ is equivalent to the $BA|A'A'$ type, since they involve the same taxon pairs, and 2 double-mixed categories: $D_1$ (which includes $AB|A'B'$ and $BA|B'A'$) and $D_2$ (which includes $AB|B'A'$ and $BA|A'B'$).

To compute $d_p(T, T')$, it is sufficient to demonstrate how to compute it for one representative type $XY|X'Y'$ from each category, where $X, Y \in \{A, B\}$ and $X', Y' \in \{A', B'\}$. Specifically, we calculate the *partial distances* as:

$$d^p_{XY|X'Y'} = \sum_{x \in X \cap X'} \sum_{y \in Y \cap Y'} |\xi(\mathsf{lca}_T(x, y)) - \xi'(\mathsf{lca}_{T'}(x, y))|^p .$$

Then, the cophenetic distance is the $\frac{1}{p}$-th power of the sum of the partial distances computed for each representative of the category. In the next sections, we show algorithms to compute partial distances for each representative type.

**Table 1. Categories of the types of taxon pair sets.** The representative type of each category is marked by a star.

| $XY\|X'Y'$ | $A'A'$ | $A'B'$ | $B'A'$ | $B'B'$ |
|---|---|---|---|---|
| $AA$ | $N_1^*$ | $S_3*$ | $S_3$ | $N_2*$ |
| $AB$ | $S_1*$ | $D_1*$ | $D_2*$ | $S_2*$ |
| $BA$ | $S_1$ | $D_2$ | $D_1$ | $S_2$ |
| $BB$ | $N_3*$ | $S_4*$ | $S_4$ | $N_4*$ |

https://doi.org/10.1371/journal.pcbi.1013069.t001

## The partial distance of non-mixed types

According to Table 1, there are four types of non-mixed taxon pairs: $AA|A'A'$, $AA|B'B'$, $BB|A'A'$, and $BB|B'B'$. To calculate the partial distance for each non-mixed type, we begin by contracting the trees $T$ and $T'$ to the set of leaves defined by the corresponding pair. For instance, for the type $BB|A'A'$, the trees $T$ and $T'$ are contracted to the set $B \cap A'$.

We then compute the partial distance $d_p$ for these contracted trees recursively. See also lines 14-15 in Algorithm 5.

If $f(n)$ represents the complexity of the algorithm for computing the distance between trees of size $n$, the total time to calculate these four partial distances is $\sum_{i=1}^{4} f(c_i)$, where $c_1 = |A \cap A'|$, $c_2 = |A \cap B'|$, $c_3 = |B \cap A'|$, and $c_4 = |B \cap B'|$ (note that $\sum_i c_i = n$.), plus the linear cost of performing the contractions.

## The partial distance of double-mixed types

Here, we present algorithms for computing partial distances for the categories $D_1$ and $D_2$, represented by the two double-mixed types $AB|A'B'$ and $AB|B'A'$, respectively.

**Double-mixed types $AB|A'B'$.** The type $AB|A'B'$ denotes pairs $\langle x, y\rangle$, where $x$ belongs to the upper trees and $y$ to the lower trees of both $T$ and $T'$. In this case, the lca of $\langle x, y\rangle$ is positioned along the median path in both trees. Furthermore, $\mathsf{lca}_T(x, y)$ is the same as $\mathsf{lca}_T(x, t)$. Similarly, we have $\mathsf{lca}_{T'}(x, y) = \mathsf{lca}_{T'}(x, t')$.

Having this, the partial distance is

$$d^p_{AB|A'B'} = \sum_{x \in A \cap A'} |\xi(\mathsf{lca}_T(x,t)) - \xi'(\mathsf{lca}_{T'}(x,t'))|^p \cdot |B \cap B'|, \tag{1}$$

which can be computed in $O(pn)$ time.

**Double-mixed types $AB|B'A'$.** For $AB|B'A'$ the naïve approach requires $\Theta(pn^2)$ steps. Below, we show an $O(pn)$ time solution. We begin with the following problem.

**Problem 1.** *Given two sequences of numbers:* $\alpha_1 \le \alpha_2 \le \ldots \le \alpha_k$ *and* $\beta_1 \le \beta_2 \le \ldots \le \beta_m$. *Compute:* $\sum_{i,j} |\alpha_i - \beta_j|^p$.

**Algorithm 1 Function GetCntr (partially adapted from [24]).**

```
1: Function GetCntr(G, g, X), where g is a median vertex of G
2:       Set v.c := 0 for every vertex v on the median path of G # Init
   counters
3:       For every leaf l in X: lca_G(g, l).c = lca_G(g, l).c + 1
4:       Initialize an empty sequence γ (a list)
5:       For every vertex v on the median path: append ξ(v) to γ,
   repeated v.c times
6:       Return γ
```

**Lemma 2.** SeqPrd *from Algorithm 2 computes* $\sum_{i,j} |\alpha_i - \beta_j|^p$ *in* $O(p(m+k))$ *time and space.*

*Proof*: *Correctness:* Note that the swap in the 5-th line ensures that $\alpha_k \le \beta_m$. Let $\lambda_j$ be the number of elements from $\alpha$ that are smaller or equal to $\beta_j$. In particular we have $\lambda_m = k$. The algorithm of SeqPrd in the main loop computes $\sigma_{j,l}$ as $\sum_{i=1}^{\lambda_j} \alpha_i^l$, i.e., it is the sum of the $l$-th powers of all elements from $\alpha$ that are smaller or equal to $\beta_j$. In particular, $\sigma_{m,l} = \sum_{i=1}^{k} \alpha_i^l$. Then, for a fixed $j$,

**Algorithm 2 Partial distances: type $AB|B'A'$.**

```
1: Input: T and T' with median vertices t and t', resp.
2: Output: d^p_{AB|B'A'}(T, T')
3: Function SeqPrd(p, α_1, α_2, ..., α_k, β_1, β_2, ..., β_m):
4:         i := j := 1
5:         If α_k > β_m Then swap α and β
6:         Let σ_{j,l} := 0 for all l ∈ {0, ..., p} and j ∈ {0, ..., m}
7:         While j ≤ m:
8:                 If i ≤ k and α_i ≤ β_j
9:                 Then i = i + 1; For l ∈ {0, ..., p} : σ_{j,l} = σ_{j,l} + α^l_{i-1}
10:                Else j = j + 1; For l ∈ {0, ..., p} : σ_{j,l} := σ_{j-1,l}
11:        Return Σ_{j=1}^{m} Σ_{l=0}^{p} C(p,l)(-1)^{p-l}(β_j^l σ_{j,p-l} + β_j^{p-l}(σ_{m,l} - σ_{j,l}))
12: Let α = GetCntr(T, t, A ∩ B') and β = GetCntr(T', t', B ∩ A')
13: Return SeqPrd(p, α, β)
```

$$\begin{aligned}\sum_{i=1}^{k}|\alpha_i-\beta_j|^p &= \sum_{i=1}^{\lambda_j}(\beta_j-\alpha_i)^p+\sum_{i=\lambda_j+1}^{k}(\alpha_i-\beta_j)^p\\ &=\sum_{i=1}^{\lambda_j}\sum_{l=0}^{p}\binom{p}{l}\beta_j^l(-1)^{p-l}\alpha_i^{p-l}+\sum_{i=\lambda_j+1}^{k}\sum_{l=0}^{p}\binom{p}{l}\alpha_i^l(-1)^{p-l}\beta_j^{p-l}\\ &=\sum_{l=0}^{p}\binom{p}{l}\beta_j^l(-1)^{p-l}\Big(\sum_{i=1}^{\lambda_j}\alpha_i^{p-l}\Big)+\sum_{l=0}^{p}\binom{p}{l}(-1)^{p-l}\beta_j^{p-l}\Big(\sum_{i=\lambda_j+1}^{k}\alpha_i^l\Big)\\ &=\sum_{l=0}^{p}\binom{p}{l}(-1)^{p-l}\Big(\beta_j^l\sum_{i=1}^{\lambda_j}\alpha_i^{p-l}+\beta_j^{p-l}\sum_{i=\lambda_j+1}^{k}\alpha_i^l\Big).\end{aligned}$$

Now, the formula in the 11-th line is derived by applying above the following identities $\sum_{i=1}^{\lambda_j}\alpha_i^{p-l}=\sigma_{j,p-l}$ and $\sum_{i=\lambda_j+1}^{k}\alpha_i^l=\sum_{i=1}^{k}\alpha_i^l-\sum_{i=1}^{\lambda_j}\alpha_i^l=\sigma_{m,l}-\sigma_{j,l}$.

*Time and space complexity:* The main loop requires $O(m+k)$ steps, while computing the formula in the 11-th line requires $O(pm)$ steps plus precomputing all values of binomial coeficients $\binom{p}{l}$ for $l\in\{0\dots p\}$, which can be done once in $O(p)$ time. In total, the overall time complexity is $O(p(m+k))$, the same applies to space complexity. □

Now, $d^p_{AB|B'A'}$ and similarly, $d^p_{BA|A'B'}$ is computed by calling $\mathsf{SeqPrd}(p,\alpha,\beta)$, where $\alpha$ is the sequence of contributions of $\mathsf{lca}_T(x,t)$'s for $x$ in $A\cap B'$ and $\beta$ is the sequence of contributions of $\mathsf{lca}_{T'}(y,t')$'s for $y$ in $B\cap A'$. Such sequences are inferred in $O(n)$ steps by the function GetCntr in Algorithm 1.

**Lemma 3.** *Algorithm 2 computes $d^p_{AB|B'A'}$ in $O(pn)$ time.*

*Proof*: The proof is similar to the proof of Lemma 3 from [24]. The difference is in the more general computation of the sum from Lemma 2. We leave out simple details. □

## The partial distance of single-mixed types

There are four single-mixed categories represented by the types $AA|A'B'$, $AB|A'A'$, $BB|A'B'$, and $AB|B'B'$ (see Table 1). Each of these variants can be solved similarly. Thus, we present only the algorithm for computing the partial distance for $AA|A'B'$.

Assume that the pair of taxa $\langle x,y\rangle$ has the type $AA|A'B'$, meaning $x\in A\cap A'$ and $y\in A\cap B'$. In this case, $\mathsf{lca}_T(x,y)$ is a vertex from the upper part of tree $T$, while $\mathsf{lca}_{T'}(x,y)$ lies on the

median path of $T'$. To apply our algorithm, we consider the tree to be ordered. For non-leaf vertices, the right child of a vertex $v$ is denoted $v.\mathsf{rgh}$, and the left child is denoted $v.\mathsf{lft}$.

Let $x.\beta = \xi'(\mathsf{lca}_{T'}(x,t'))$ for $x \in A \cap A'$. Our solution is divided depending on the parity of $p$. We start with the solution to odd norms.

**Single-mixed types under odd $L_p$ norms.** We start with the following definitions.

- $L_v^+ := \begin{cases} \varnothing & \text{if } v = \mathsf{root}(T), \\ \{x : x \in A \cap A' \cap L_v \text{ and } x.\beta \le \xi(v.parent)\} & \text{otherwise,} \end{cases}$
- and $L_v^- := (A \cap A' \cap L_v) \setminus L_v^+$.

In the next two lemmas, we prove several properties of vertex attributes $\delta$ and $\sigma$ from Algorithm 3.

**Algorithm 3 Partial distances: type $AA|A'B'$ for odd $p$'s.**

```
1: Input: T and T' with median verticest and t', respectively; p is odd
2: Output: d^p_{AA|A'B'}(T, T')
3: For every v in the upper tree of T:
4:         v.κ := 0;  v.δ := v.σ^+ := v.σ^- := 0 ∈ R^{p+1}  # the zero vector
5: For x ∈ A ∩ A': # The preprocessing loop
6:         x.β = ξ'(lca_{T'}(x, t'));  β* = ⟨1, x.β^1, x.β^2, ..., x.β^p⟩
7:         If x.β ≤ ξ(x)
8:                 Then ω_x := arg min_w{ξ(w) | x ≼ w and x.β ≤ ξ(w)};  ω_x.δ = ω_x.δ + β*
9:         If x.β ≤ ξ(x.parent) Then x.σ^+ := β* Else x.σ^- := β*
10: For every non-root v in the upper tree of T in postfix order: #
    The main loop
11:        v.σ^+ := v.lft.σ^+ + v.rgh.σ^+ − v.δ;        v.σ^- := v.lft.σ^- + v.rgh.σ^- + v.δ
12:        Y_v = A ∩ B' ∩ L(v.sibling)
13:        v.κ := |Y_v| Σ_{l=0}^{p} C(p,l)(−1)^{p−l}(ξ(v.parent)^l v.σ^+_{p−l} + ξ(v.parent)^{p−l} v.σ^-_l)
14: Return Σ_{v∈T} v.κ
```

**Lemma 4.** *If $v$ is strictly internal, then after the preprocessing loop of Algorithm 3, we have*

$$v.\delta = \langle \sum_{x \in \Omega_v} (x.\beta)^l : l \in \{0,1,\dots,p\} \rangle,$$

*where* $\Omega_v = \{x \in L_v \cap A \cap A' : \xi(v) \ge x.\beta > \xi(v.\mathsf{parent})\}$.

*Proof*: In line 8 of Algorithm 2, $\omega_x$ is defined as the vertex such that $x.\beta \le \xi(\omega_x)$, and this inequality is not satisfied by the parent of $\omega_x$; that is, $x.\beta > \xi(\omega_x.\mathsf{parent})$. Combining these observations, we can conclude that $\omega_v$ consists of vertices $x \in A \cap A'$ for which $v = \omega_x$ as specified in line 8. Consequently, from the assignment in line 8, we have
$v.\delta = \langle \sum_{x \in \Omega_v} (x.\beta)^l : l \in \{0,1,\dots,p\} \rangle$. □

**Lemma 5.** *If $v$ is strictly internal, then after the main loop of Algorithm 3,*

- $v.\sigma^+ = \langle \sum_{x \in L_v^+} x.\beta^l : l \in \{0,1,\dots,p\} \rangle$,
- and $v.\sigma^- = \langle \sum_{x \in L_v^-} x.\beta^l : l \in \{0,1,\dots,p\} \rangle$.

*Proof*: We present an inductive proof for a fixed $l \in \{0,1,\dots,p\}$ and for $v.\sigma^+_{j+1}$, which denotes the $(j+1)$-th element of the vector $v.\sigma^+$. If $v$ is a leaf, then the equality follows directly from line 9 of Algorithm 3 and the definition of $L_v^+$. Now, let us consider the case where $v$ is

strictly internal. Note that $\xi(v) \geq \xi(v.\mathsf{parent})$.

$$\begin{aligned} v.\sigma^+_{l+1} &= v.\mathsf{lft}.\sigma^+_{l+1} + v.\mathsf{rgh}.\sigma^+_{l+1} - v.\delta_{l+1} \\ &= \sum_{x \in L^+_{v.\mathsf{lft}}} x.\beta^l + \sum_{x \in L^+_{v.\mathsf{rgh}}} x.\beta^l - v.\delta_{l+1} \\ &= \sum_{\substack{x \in A \cap A' \cap L_{v.\mathsf{lft}} \\ x.\beta \leq \xi(v.\mathsf{lft}.\mathsf{parent})}} x.\beta^l + \sum_{\substack{x \in A \cap A' \cap L_{v.\mathsf{rgh}} \\ x.\beta \leq \xi(v.\mathsf{rgh}.\mathsf{parent})}} x.\beta^l - v.\delta_{l+1} \\ &= \sum_{\substack{x \in A \cap A' \cap L_v \\ x.\beta \leq \xi(v)}} x.\beta^l - \sum_{\substack{x \in A \cap A' \cap L_v \\ \xi(v) \geq x.\beta > \xi(v.\mathsf{parent})}} x.\beta^l \\ &= \sum_{\substack{x \in A \cap A' \cap L_v \\ x.\beta \leq \xi(v.\mathsf{parent})}} x.\beta^l = \sum_{x \in L^+_v} x.\beta^l. \end{aligned}$$

The proof for $v.\sigma^-$ is similar. We omit details. □

**Lemma 6.** *If $v$ is strictly internal in $T$, then after the main loop of Algorithm 3 we have*

$$v.\varkappa = \sum_{x \in A \cap A' \cap L_v} \sum_{y \in A \cap B' \cap L(v.\mathsf{sibling})} |\xi(\mathsf{lca}_T(x,y)) - \xi'(\mathsf{lca}_{T'}(x,y)|^p. \tag{2}$$

*Proof*: We can prove (2) by using Lemmas 4 and 5. Let $v$ be strictly internal in the upper tree of $T$. Then, for a leaf $x \in A \cap A' \cap L_v$ and a leaf $y$ from $Y_v$ (i.e., from $A \cap B' \cap L(v.\mathsf{sibling})$, see line 12), we have $\mathsf{lca}_T(x,y) = v.\mathsf{parent}$ and $\xi'(\mathsf{lca}_{T'}(x,y)) = \xi'(\mathsf{lca}_{T'}(x,t')) = x.\beta$. Let $R$ be the right side of (2). Then,

$$\begin{aligned} R &= \sum_{x \in A \cap A' \cap L_v} \sum_{y \in Y_v} |\xi(v.\mathsf{parent}) - x.\beta|^p \\ &= \sum_{y \in Y} \Big( \sum_{\substack{x \in A \cap A' \cap L_v \\ x.\beta \leq \xi(v.\mathsf{parent})}} (\xi(v.\mathsf{parent}) - x.\beta)^p - \sum_{\substack{x \in A \cap A' \cap L_v \\ x.\beta > \xi(v.\mathsf{parent})}} (\xi(v.\mathsf{parent}) - x.\beta)^p \Big) \\ &= |Y_v| \Big( \sum_{x \in L^+_v} (\xi(v.\mathsf{parent}) - x.\beta)^p - \sum_{x \in L^-_v} (\xi(v.\mathsf{parent}) - x.\beta)^p \Big) \\ &= |Y_v| \sum_{l=0}^{p} \binom{p}{l} (-1)^{p-l} \big( \xi(v.\mathsf{parent})^l v.\sigma^+_{p-l+1} - \xi(v.\mathsf{parent})^{p-l} v.\sigma^-_{l+1} \big) \end{aligned}$$

The last equation is obtained by identities from Lemma 5 and binomial expansions. $R$ equals the right side of the assignment from line 13. □

**Lemma 7.** *Algorithm 3 computes $d^p_{AA|A'B'}$ in $O(n \log n + pn)$ time.*

*Proof*: We show the algorithm's correctness followed by the stated time complexity.

*Correctness:* Let $I$ be the set of non-root vertices from the upper tree of $T$. Then, every pair of leaves $\langle x, y \rangle$ of type $AA|A'B'$ uniquely determines $v \in I$ such that $\mathsf{lca}_T(x,y) = v.\mathsf{parent}$ and $x \in L_v$. It also follows that $y \in L(v.\mathsf{sibling})$. Let $x \oplus y$ denote such vertex $v$. For a given $v$, $A \cap A' \cap L_v \times A \cap B' \cap L(v.\mathsf{sibling})$ is the set of all pairs $\langle x, y \rangle$ such that $x \oplus y = v$. Hence,

$$d^p_{AA|B'A'} = \sum_{v \in I} \sum_{x \in A \cap A' \cap L_v} \sum_{y \in A \cap B' \cap L(v.\mathsf{sibling})} |\xi(\mathsf{lca}_T(x,y)) - \xi'(\mathsf{lca}_{T'}(x,y))|^p.$$

By Lemma 6 the above sum equals the value returned in the last line of Algorithm 3.

*Time complexity:* The key aspect of the time complexity is found in line 8. We show that $\omega$ can be found by a binary search in $O(\log n)$ time that seeks the value in an ordered array composed of vertices on the path connecting a given leaf $x$ with the root of $T$. An infix traversal of $T$ can construct such an array. Then, a vertex is inserted into the array when visited for the first time. When a vertex is visited for the last time, it is removed from the array. Due to the monotonic ordering of paths, the array is always sorted, and its size is limited by $n$. The time complexity of the remaining loops is $O(pn)$, which gives $O(n \log n + pn)$ time of Algorithm 3. □

**Algorithm 4 Partial distances: type $AA|A'B'$ for even $p$.**

```
1: Input: T and T' with median vertices t and t', resp.
2: Output: d^p_{AA|A'B'}(T, T')
3: For every v in the upper tree of T: v.κ := 0. v.σ := 0 ∈ R^{p+1} (the zero vector)
4: For x ∈ A ∩ A': # Preprocessing
5:         x.β = ξ'(lca_{T'}(x, t')); x.σ := ⟨x.β^l : l ∈ {0, 1, ..., p}⟩
6: For every non-root v in the upper tree of T in postfix order:
7:         v.σ := v.lft.σ + v.rgh.σ
8:         v.κ := |A ∩ B' ∩ L(v.sibling)| · Σ_{l=0}^{p} C(p,l) (−1)^{p−l} ξ(v.parent)^{p−l} v.σ_l
9: Return Σ_{v∈T} v.κ
```

**Single-mixed types under even $L_p$ norms.** The advantage of even $L_p$ norms is given by the relation $(|a - b|)^p = (a - b)^p$ for $p$ even. This fact allows us to circumvent the extra complexity of Algorithm 3. Algorithm 4 outlines the computation of $d^p_{AA|A'B'}$ when $p$ is even.

**Lemma 8.** *For each non-root vertex v in the upper tree of $T$ after Algorithm 4 we have*

- $v.\sigma = \langle \sum_{x \in A \cap A' \cap L_v} (x.\beta)^l : l \in 0, \ldots, p \rangle$,
- $v.\kappa = \sum_{x \in A \cap A' \cap L_v} \sum_{y \in A \cap B' \cap L(v.\text{sibling})} |\xi(\text{lca}_T(x,y)) - \xi'(\text{lca}_{T'}(x,y))|^p$.

*Proof*: The relation for $v.\sigma$ follows directly from the bottom-up nature of the algorithm and the relation for $v.\kappa$ can be observed using the following equation:

$$\begin{aligned}
&\sum_{x \in A \cap A' \cap L_v} \sum_{y \in A \cap B' \cap L(v.\text{sibling})} |\xi(\text{lca}_T(x,y)) - \xi'(\text{lca}_{T'}(x,y))|^p \\
&= \sum_{x \in A \cap A' \cap L_v} \sum_{y \in A \cap B' \cap L(v.\text{sibling})} (x.\beta - \xi(v.\text{parent}))^p \\
&= |A \cap B' \cap L(v.\text{sibling})| \sum_{l=0}^{p} \binom{p}{l} (-1)^{p-l} \xi(v.\text{parent})^{p-l} \sum_{x \in A \cap A' \cap L_v} (x.\beta)^l.
\end{aligned}$$

□

**Lemma 9.** *Algorithm 4 computes $d^p_{AA|A'B'}(T, T')$ in $O(pn)$ time for even $p$.*

*Proof*: The correctness of the algorithm follows from Lemmas 8 and 7. Finally, the time complexity of Algorithm 4 is $O(pn)$ since each line within the loops requires $O(p)$ time and is executed at most $2n$ times. □

**Partial distances of the remaining single-mixed types.** As previously mentioned, Algorithm 3 and Algorithm 4 for computing the partial distances of single mixed type $AA|A'B'$ can be adapted to solve the other types as follows. For the type $BB|A'B'$, replace the term "upper" with "lower" and $A$ with $B$ in both algorithms. For the type $AB|A'A'$, swap

the input trees and execute both algorithms. For the type $AB|B'B'$, swap the input trees and run the algorithm designed for the type $BB|A'B'$. These modifications are detailed in lines 14 through 16 of Algorithm 5.

## Algorithm to compute the cophenetic distance under the $L_p$ norm

The pseudo-code in Algorithm 5 summarizes the complete procedure for computing the cophenetic distance for finite $p$. The correctness of the algorithm follows from the results presented in the previous section. Below, we analyze the time complexity in two scenarios: when $p$ is constant, and when $p$ is considered as a parameter in the asymptotic analysis.

**Algorithm 5 Computing cophenetic distance.**

1: **Input:** $T$ and $T'$ with the same set of leaves of size $n$, two ascending contribution functions $\xi: V_T \to \mathbb{R}$ and $\xi': V_{T'} \to \mathbb{R}$, and an integer $p \geq 1$
2: **Output:** The $L_p$-cophenetic distance between $T$ and $T'$ with respect to $\xi$ and $\xi'$
3: Compute all binomial coeficients $a_i = \binom{p}{i}$, for all $i$, by $a_i = \frac{p-i+1}{i} a_{i-1}$ and $a_0 = 1$
4: **Function** PartDist$(T, T')$
5: Compute $t$ and $t'$ the median vertices of $T$ and $T'$, resp.
6: Let $A$ and $A'$ be the set of all leaves in the upper tree of $T$ and $T'$, resp.
7: $s := 0$; $B := L_T \setminus A$; $B' := L_{T'} \setminus B$
8: For $X$, in $\{A, B\}$: For $X'$ in $\{A', B'\}$
9: Let $S :=$ contract$(T, X \cap X')$ and $S' :=$ contract$(T', X \cap X')$
10: $s = s +$ PartDist$(S, S')$ # *Compute partial distances for type* $XX|X'X'$
11: $s = s + d^p_{AB|A'B'}$ see Eq. (1) # *Double-mixed* $AB|A'B'$
12: $s = s + d^p_{AB|B'A'}$ by Alg. 2 # *Double-mixed* $AB|B'A'$
13: # *All single-mixed types*
14: $s = s + d^p_{AA|A'B'}$ using Alg. 3 if $p$ is odd, or Alg. 4 otherwise
15: Similarly to line 14 , $s = s + d^p_{BB|A'B'}$, but replace "upper" with "lower" and $A$ with $B$ in Alg. 3/4
16: For $AB|A'A'$ and $AB|B'B'$ swap the input trees and repeat lines 14 and 15
17: **Return** $s$ # *Return the sum of partial distances*
18: **Return** PartDist$(T, T')^{\frac{1}{p}}$ # *Return the cophenetic distance*

First, consider the case where $p$ is constant. Since $pn + n\log n \in O(n\log n)$, it follows from Lemmas 3 and 7 that computing the partial distances of mixed types requires $O(n\log n)$ time when $p$ is odd, and $O(n)$ time when $p$ is even. Consequently, the overall time complexity of Algorithm 5, as derived in [24], is $O(n\log^2 n)$ for odd $p$, and $O(n\log n)$ for even $p$.

If $p$ is a parameter, $f(n,p)$ is the worst-case time complexity of the complete algorithm. Then, computation of partial distances of non-mixed types (see lines 8-10 in Algoritm 5) requires $f(c_1,p)+f(c_2,p)+f(c_3,p)+f(c_4,p)+O(n)$ time where $\sum c_i = n$ and $0 \leq c_i \leq .75n+1$ for each $i$ (by Lemma 1), while the computation of mixed types requires $O(n\log n + pn)$ time if $p$ is odd or $O(pn)$ time if $p$ is even, (Lemmas 3 and 7). Therefore, for some $k \geq 1$ we can write that, $f(n,p) = 1$ if $n \leq 5$, $f(n,p) = k(n\log n + pn) + \max_{0\leq c_i \leq .75n+1} f(c_1,p)+f(c_2,p)+f(c_3,p)+f(c_4,p)$, if $n > 5$ and $p$ is odd, and $f(n,p) = kpn + \max_{0\leq c_i \leq .75n+1} f(c_1,p)+f(c_2,p)+f(c_3,p)+f(c_4,p)$, otherwise.

**Theorem 10.** *The time complexity of the algorithm is* $O(n\log^2 n + pn\log n)$, *when* $p$ *is odd and* $O(pn\log n)$ *when* $p$ *is even.*

*Proof*: *The proof for odd norms*. Assume that $p$ is odd. We show that there are constants $b \geq 1$ and $d > 0$ such that for every $n > 0$ and $p > 0$, $f(n,p) \leq d(n\log^2 n + pn\log n) + b$. The proof is by induction on $n$. For $n \leq 5$ we have $f(n,p) = 1$ and the inequality is satisfied. For $n > 5$, $f(n,p) = kn\log n + kpn + \max\sum_i f(c_i) \leq kn\log n + kpn + 4b + d\max\sum_i (c_i\log^2 c_i + pc_i\log c_i) \leq kn\log n + kpn + 4b + dn\log^2(.75n+1) + dpn\log(.75n+1)$.

Let $d = -bk/\log\frac{11}{12} \approx bk7.966$. Then, for $n > 5$, $\log(.75+\frac{1}{n}) \leq \log(.75+\frac{1}{6}) = \log\frac{11}{12}$. Also, $d(\log(.75n+1) - \log n) = d\log(.75+\frac{1}{n}) \leq d\log\frac{11}{12} = -bk$ and $d(\log^2(.75n+1) - \log^2 n) = d\log(.75+\frac{1}{n})\log((.75n+1)n) \leq -bk\log((.75n+1)n) \leq -bk\log n - bk$. Finally, for $n > 5$, $f(n) - dn\log^2 n - dnp\log n - b \leq kn\log n + 3b + dn(\log^2(.75n+1) - \log^2 n) + dpn(\log(.75n+1) - \log n) \leq (1-b)kn\log n + 3b - nbk + pn(-bk) < 0$.

*Even norms*. If $p$ is even, it suffices to show that $f(n,p) \leq dpn\log n + b$, for some constants $d > 0$ and $b \geq 1$. The proof is similar and simpler compared to the odd case and follows the same reasoning. We omit details. □

The last theorem shows that, despite the higher complexity of odd norms, the factor $p$ only appears in the asymptotically minor term of $O(n\log^2 n + pn\log n)$. This suggests that the computational effort for cophenetic distances is more influenced by the size of the input trees than by the chosen norm level $p$. Additionally, this theoretical result highlights that the binary search algorithm - required for computing double-mixed types and representing the main distinction between the two algorithmic variants - introduces $O(n\log^2 n)$ more steps in the worst-case than the algorithm for even norms. See the experimental section for a detailed discussion on scalability and a comparison of the algorithms for odd and even norms.

**Cophenetic distances under $L_\infty$ norm.** Algorithm 5 can be easily adapted to solve the last remaining case of cophenetic distances. The algorithm for the $L_\infty$ norm is similar to the case $p = 1$. The only difference is taking the maximum value instead of adding partial distances as follows. First, fix $p := 1$. Then in line 7, replace $s := 0$ with $s := -\infty$. In lines 10-16 instead of assignments $s = s + r$, where $r$ is the right-hand expression, write $s := \max\{s, r\}$. Now, the time complexity is $O(n\log^2 n)$, which follows from Theorem 10 for the odd case $p = 1$.

## Results

In the following sections, we present the scalability study and distribution analysis results. The near-linear time algorithm, i.e., Algorithm 5, for computing the $L_p$ norm cophenetic distance and the naïve (quadratic time) algorithm were implemented in Rust 1.79.0. The implementation of both algorithms, along with instructions on reproducing the simulation results below, is available on GitHub [25]. Note that the distances computed by the naïve and our algorithm are the same.

### Scalability analysis

We investigate the scalability of our algorithm by comparing its runtime to that of the best-known quadratic-time algorithm for computing cophenetic distance.

**Datasets.** We generated three datasets consisting of pairs of random binary trees with $n$ leaves using the classic Yule model, where for each $n$ we generated $q$ pairs. The first dataset was generated using $n = 25, 50, \ldots, 600$ and $q = 1000$, the second with $n = 200, 400, \ldots, 2600$ and $q = 1000$, and the last one with $n = 1000, 2000, \ldots, 6000$ and $q = 100$. Algorithms were executed on each pair of trees for various norms $L_p$ with $p$ values ranging from 1 to 100. The comparative runtime analysis for each dataset is illustrated in the diagrams of Figs 2, 3, and 4.

**Results.** Our algorithm shows a significant improvement in efficiency compared to the quadratic solution for trees with more than 250 taxa and norms below 20, as illustrated in

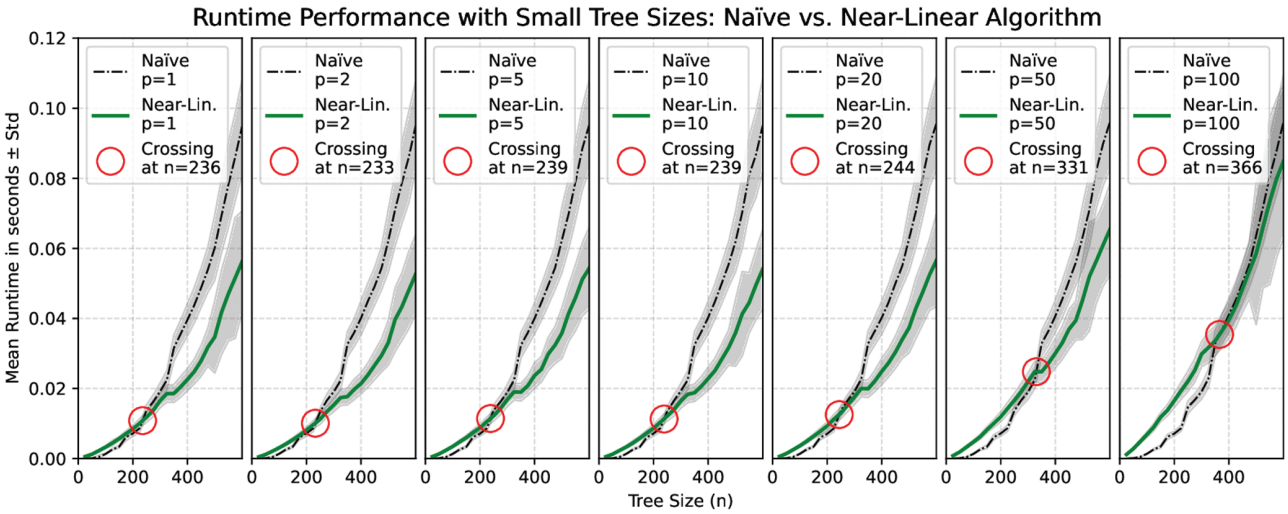



**Fig 2. Scalability analysis comparing the near-linear algorithm and the naïve quadratic-time algorithm.** The diagrams show the average runtime with standard deviation bands over 1000 runs for each tree size ($n = 25, 50, \ldots, 600$) on pairs of random trees, evaluated for $L_p$-norms ($p = 1, 2, 5, 10, 20, 50, 100$). Crossing points indicate the tree sizes where the near-linear algorithm outperforms the naïve one.

https://doi.org/10.1371/journal.pcbi.1013069.g002

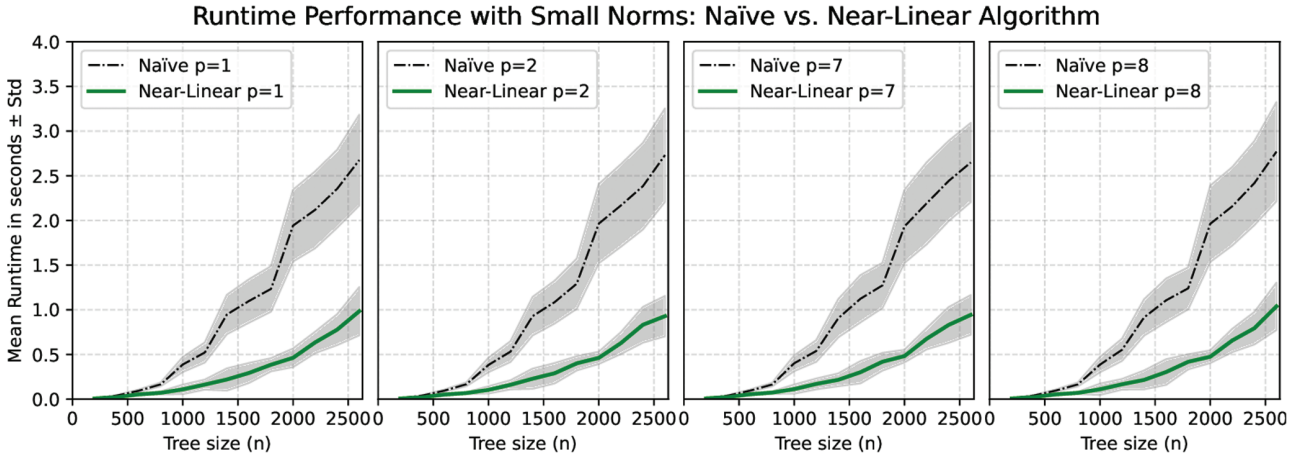



**Fig 3. Scalability analysis comparing the near-linear algorithm and the naïve quadratic-time algorithm for small norms.** The diagrams show the average runtime with standard deviation bands over 1,000 runs for each tree size ($n = 200, 400, \ldots, 2600$) on pairs of random trees, evaluated for $L_p$-norms ($p = 1, 2, 7, 8$).

https://doi.org/10.1371/journal.pcbi.1013069.g003

Fig 2. Furthermore, the crossing point at which our algorithm outperforms the quadratic solution increases gradually as $p$ increases, reaching 366 at $p = 100$. This finding indicates the potential for developing a hybrid algorithm that utilizes the divide-and-conquer method for larger trees while reverting to the quadratic time algorithm for trees with fewer than 230–370 leaves during recursion.

Despite the differences in asymptotic worst-case time complexity between odd and even norms, these differences are not evident in the average runtime diagrams presented in Figs 3 and 4. Many of these differences can be attributed to the binary search step in Algorithm 3, as explained in Lemma 7, where the runtime is relatively straightforward to estimate. Notably, the computation of odd norms accounted for only 0.01% of the total average runtime in the

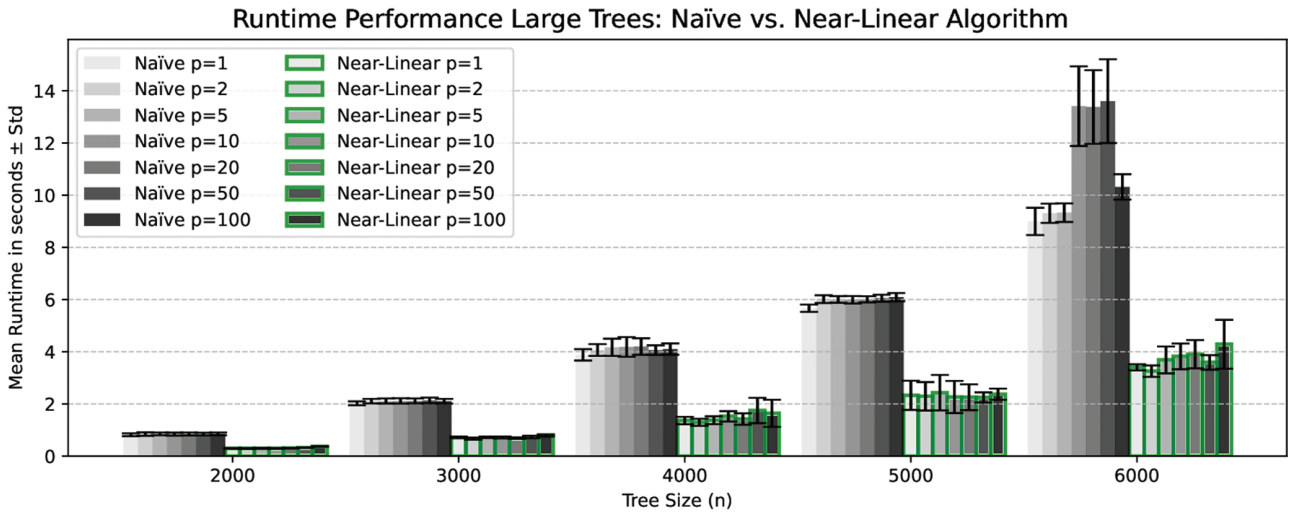



**Fig 4. Scalability analysis comparing the near-linear algorithm (outlined bars) and the naïve quadratic-time algorithm on large trees.** The diagrams show the average runtime with standard deviation error over 100 runs for each tree size ($n = 1000, 2000, \ldots, 6000$) on pairs of random trees, evaluated for $L_p$-norms ($p = 1, 2, 5, 10, 20, 50, 100$).

https://doi.org/10.1371/journal.pcbi.1013069.g004

binary search steps. In practical terms, this indicates that the time complexity of Algorithm 3 is closer to $O(pn)$ rather than the more conservative estimate of $O(n \log n + pn)$ provided in Lemma 7. Consequently, the overall runtime complexity for calculating the cophenetic distance can be approximated as $O(pn \log n)$ for any given norm. Since the trees were generated randomly, we propose that $O(pn \log n)$ is also a valid estimator of the average time complexity of our algorithm. However, as shown in Fig 4, there are no significant differences in runtime across various norms for a fixed tree size $n$. This indicates that the runtime is primarily affected by the overhead associated with maintaining supporting data structures, rather than by the computation of values, which involves loops over the range from 0 to $p$. We conjecture that for extremely large values of $p$, the computational cost of these loops will become increasingly noticeable in the runtime; however, such scenarios were not tested in this study.

## Cophenetic distance distributions

We investigate the distributions of three key representatives from the generalized class of cophenetic distances. These representatives are (i) the original cophenetic distance metric defined by depths, (ii) the metric defined by the heights of the subtrees, and (iii) the metric defined by the number of taxa in the subtrees. For ease of reference, we refer to these metrics as the *depth*, *subtree-height*, and *subtree-size* cophenetic metrics, respectively.

To the best of our knowledge, the specific distributions for any selected metrics remain unpublished. Therefore, we present here the sampled distributions of cophenetic distances based on two standard models of phylogenetic tree sampling: the uniform model and the Yule model [28,29]. The sampled distributions for the classical depth-based cophenetic distance were previously discussed in [1] for both the $L_1$ norm and the $L_2$ norm under the uniform model.

**Data.** We generated a dataset containing $10^6$ pairs of trees, each with 100 leaves. Each tree was independently generated under the uniform model. Similarly, we generated another dataset based on the Yule model. It is important to note that the generated trees do not

include edge lengths; therefore, both depths and subtree heights are measured in terms of the number of edges.

**Results.** The sampled distributions are illustrated in Figs 5 and 6, with the corresponding mean and standard deviation statistics depicted in Fig 7.

The distribution of subtree-height and depth cophenetic metrics exhibit similar shapes under the uniform model for smaller values of $p$ as illustrated in Fig 5. A similar trend is observed for the depth under the Yule model. Furthermore, these distributions are positively skewed, which is consistent with the findings for the $L_1$ and $L_2$ norms of the depth cophenetic metric reported in [1]. On the contrary, the subtree-height distributions exhibit positive skewness. It is also worth noting that our study involved a significantly larger number of pairs for sampling. As a result, the diagrams in Fig 5 appear smoother compared to the sampled distributions of depth metric from [1].

Since cophenetic vectors are finite, it is straightforward to prove that $\lim_{p\to\infty} d_p(T, T') = d_\infty(T, T')$ under any contribution function. Consequently, $L_p$ cophenetic distributions converge to $L_\infty$ distributions as $p$ tends to infinity. Furthermore, in our case, all three contribution functions return integer values, which implies that the $L_\infty$ distributions have an integer domain, as illustrated in Fig 6. This property is evident in diagrams, where the distributions become increasingly irregular as $p$ grows and more and more similar to the corresponding discrete distributions under $L_\infty$. This effect is particularly noticeable for the depth and subtree-height metrics. By contrast, the distributions for the subtree-size metric remain relatively smooth. This smoothness arises from the fact that, under the $L_\infty$ norm, almost all frequencies under uniform and Yule models are concentrated at 98 (see low standard deviation in Fig 7). This value is derived from the pair of leaf labels that form a cherry in one tree but are separated by the root in the other tree, yielding subtree sizes of 2 and 100, respectively, and a corresponding distance of 98.

The sampled distributions for the subtree-size cophenetic metric are notably distinct from the analysis mentioned above. The histograms for this metric are *negatively*-skewed, and the mean value under the Yule model is, in this case, larger than the mean value under the uniform model.

In comparing the depth and subtree-height cophenetic metrics, we find that the mean values under the Yule model are significantly lower than those under the uniform model. A similar bias towards the Yule model was previously noted in the sampled distributions for the path-difference distance [30,31].

Additionally, the mean and standard deviation of the depth metric under the $L_1$ and $L_2$ norms align with the exact values reported in [32].

## Results and discussion

We introduced a novel algorithmic framework for computing the $L_p$ norm cophenetic distance, achieving a time complexity of $O(n\log^2 n + pn\log n)$ for odd values of $p$, $O(pn\log n)$ for even values of $p$, and $O(n\log n)$ for $L_\infty$ norm. This represents a substantial advancement compared to the previously best-known naïve algorithm, which requires $\Theta(pn^2)$ time.

Additionally, our scalability studies suggest that the estimated runtime of our algorithm approaches $O(pn\log n)$ time under all $L_p$ norms with finite $p$, contrasting with the larger upper asymptotic bound observed for odd values of $p$. These advancements greatly improve the usability of the cophenetic distance for large-scale phylogenetic studies and the median-tree inference of species trees from gene trees using this metric.

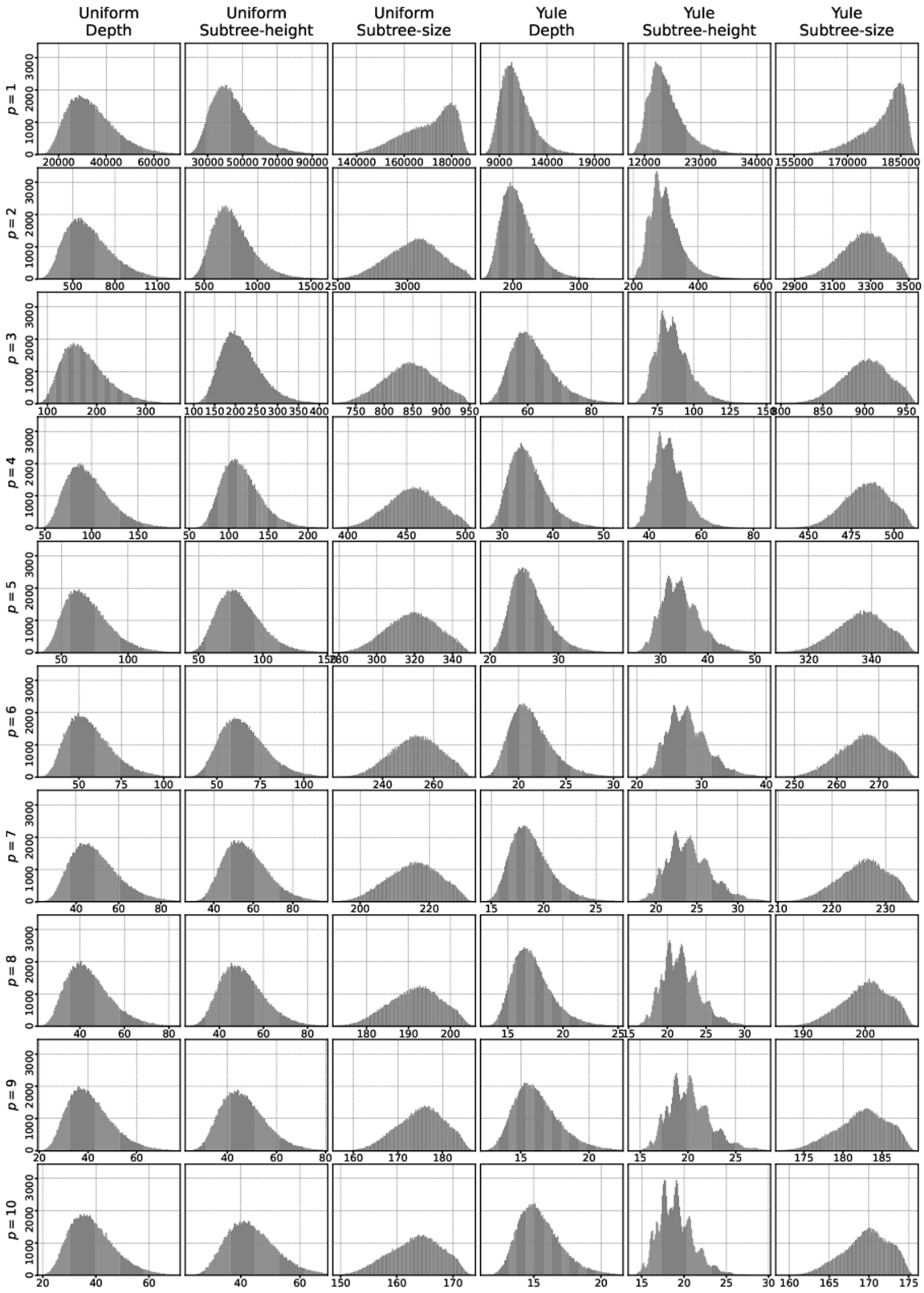



**Fig 5. Sampled distribution of three cophenetic distances under the $L_p$ norms for $p = 1$ to 10 (rows).** Pairs of trees of size 100 were sampled according to the uniform model (left columns) and the Yule model (right columns). The frequencies are grouped into 200 bins. Note that the y-axis scale and range are the same across all diagrams. To enhance visibility, low frequencies (e.g., at low distances close to 0) are omitted and the width of the diagrams is appropriately adjusted.

https://doi.org/10.1371/journal.pcbi.1013069.g005

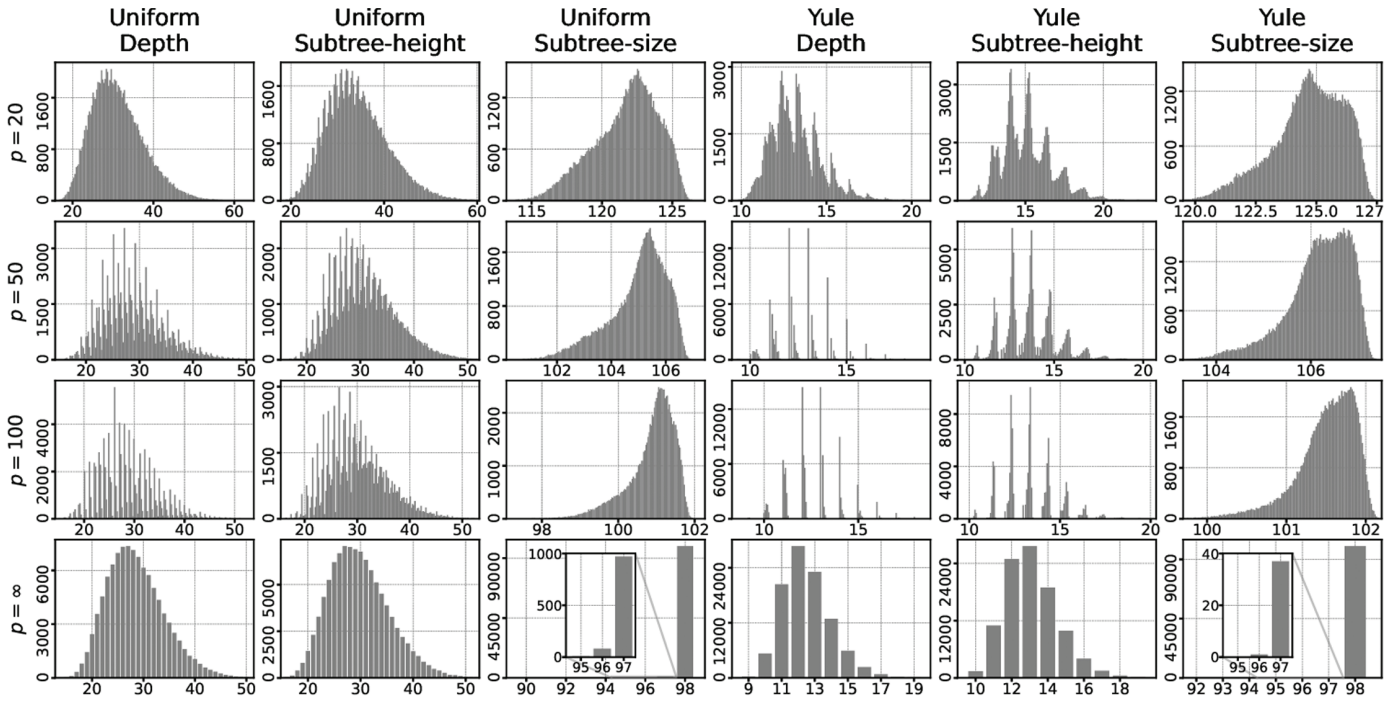



**Fig 6.** (Cont. from Fig 5). Sampled distribution of three cophenetic distances is shown for $L_p$ norms with $p = 20$, 50, 100, and $\infty$ (rows). Similar to Fig 5, the frequencies for finite $p$ are grouped into 200 bins. Since all three contribution functions return integers, the $L_\infty$ norms are also integers. As a result, the frequency values for $L_\infty$ are presented without binning.

https://doi.org/10.1371/journal.pcbi.1013069.g006

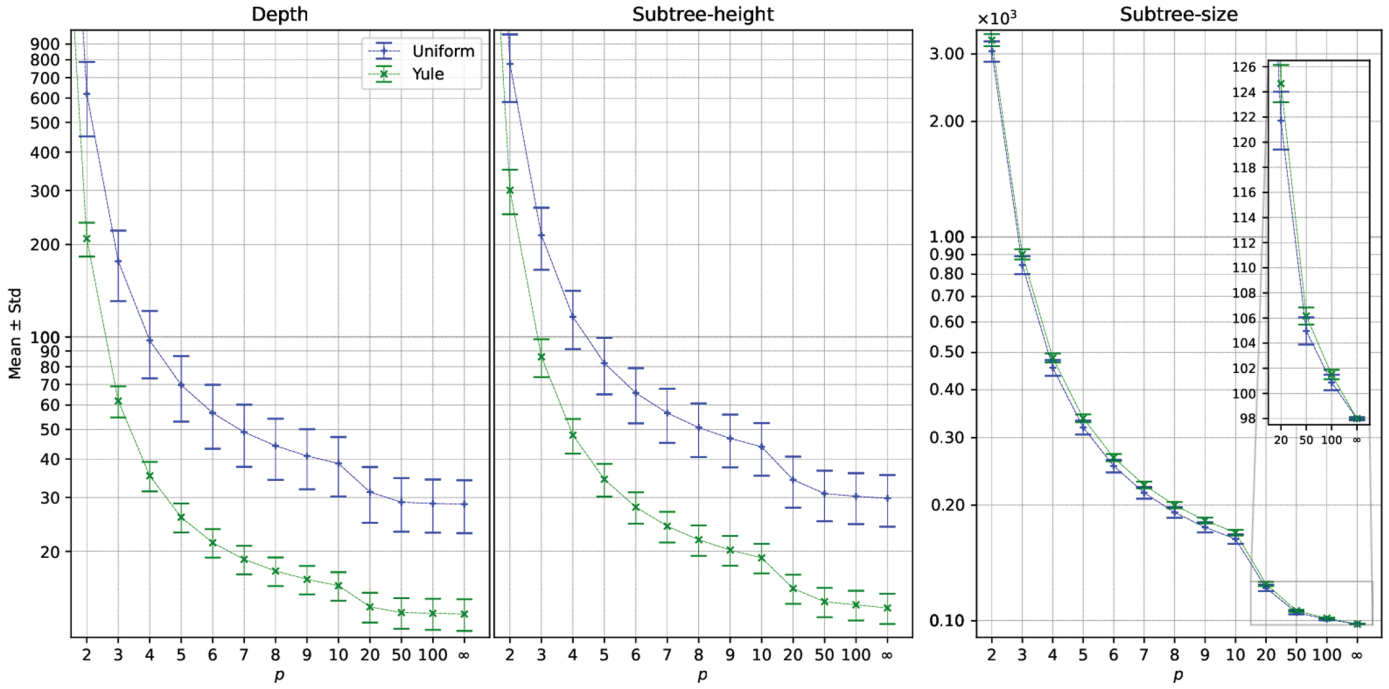



**Fig 7. Mean and standard deviation from the sampled distributions for the uniform and Yule models for $L_p$ norms with $p > 1$.**

https://doi.org/10.1371/journal.pcbi.1013069.g007

Distribution analyses of these three key representative metrics from the cophenetic class further enhance this work, offering practitioners valuable guidance in selecting appropriate metrics for their specific needs.

The framework demonstrates broad practical applicability by generalizing to all metrics that rely on the path-monotonic property, here referred to as the class of *generalized cophenetic distances*. This generalization can be achieved by either designing contribution functions that satisfy basic monotonicity properties or by forming linear, positively weighted combinations of existing contribution functions.

As a result, the class of generalized cophenetic distances includes the original cophenetic distance based on depth, as well as other metrics, particularly the subtree height and subtree size cophenetic distances. Additionally, this class incorporates the more recently proposed metric from [19], which combines both weighted and unweighted contribution functions. Consequently, this metric can be computed in near-linear time using our algorithm under any $L_p$ norm.

Furthermore, the framework can be applied to mixed scenarios where different types of cophenetic vectors are used for the trees — for example, depth in one tree and height in another. Although these mixed scenarios may not fully satisfy metric properties, they can still be useful for comparing trees from different origins by allowing asymmetry. For instance, this approach may be more suitable when comparing a gene tree, which represents the evolutionary history of a gene, to a species tree that illustrates the evolutionary history of the species from which the genes were sampled.

## Author contributions

**Conceptualization:** Paweł Górecki, Oliver Eulenstein.

**Formal analysis:** Paweł Górecki, Alexey Markin.

**Funding acquisition:** Oliver Eulenstein.

**Investigation:** Paweł Górecki, Alexey Markin, Oliver Eulenstein.

**Methodology:** Paweł Górecki, Alexey Markin.

**Project administration:** Oliver Eulenstein.

**Software:** Alexey Markin, Sriram Vijendran.

**Supervision:** Oliver Eulenstein.

**Validation:** Sriram Vijendran.

**Visualization:** Paweł Górecki, Oliver Eulenstein.

**Writing – original draft:** Paweł Górecki, Alexey Markin, Oliver Eulenstein.

**Writing – review & editing:** Paweł Górecki, Sriram Vijendran, Oliver Eulenstein.

## References

1. Cardona G, Mir A, Rosselló F, Rotger L, Sánchez D. Cophenetic metrics for phylogenetic trees, after Sokal and Rohlf. BMC Bioinformatics. 2013;14:3. https://doi.org/10.1186/1471-2105-14-3 PMID: 23323711
2. Sokal RR, Rohlf FJ. The comparison of dendrograms by objective methods. Taxon. 1962;11(2):33–40. https://doi.org/10.2307/1217208
3. Robinson DF, Foulds LR. Comparison of phylogenetic trees. Math Biosci. 1981;53(1–2):131–47. https://doi.org/10.1016/0025-5564(81)90043-2

4. Sand A, Holt MK, Johansen J, Fagerberg R, Brodal GS, Pedersen CNS, et al. Algorithms for computing the triplet and quartet distances for binary and general trees. Biology (Basel). 2013;2(4):1189–209. https://doi.org/10.3390/biology2041189 PMID: 24833220
5. Li M, Tromp J, Zhang L. On the nearest neighbour interchange distance between evolutionary trees. J Theor Biol. 1996;182(4):463–7. https://doi.org/10.1006/jtbi.1996.0188 PMID: 8944893
6. Knüver L, Fischer M, Hellmuth M, Wicke K. The weighted total cophenetic index: a novel balance index for phylogenetic networks. Discrete Appl Math. 2024;359:89–142. https://doi.org/10.1016/j.dam.2024.07.037
7. Fischer M, Herbst L, Kersting SJ, Kühn AL, Wicke K. Total cophenetic index. Springer. 2023. p. 81–7. https://doi.org/10.1007/978-3-031-39800-1_8
8. Elliott TL, Davies TJ. Jointly modeling niche width and phylogenetic distance to explain species co-occurrence. Ecosphere. 2017;8(8). https://doi.org/10.1002/ecs2.1891
9. Zogopoulos VL, Saxami G, Malatras A, Papadopoulos K, Tsotra I, Iconomidou VA, et al. Approaches in gene coexpression analysis in eukaryotes. Biology (Basel). 2022;11(7):1019. https://doi.org/10.3390/biology11071019 PMID: 36101400
10. Kuramae EE, Robert V, Echavarri-Erasun C, Boekhout T. Cophenetic correlation analysis as a strategy to select phylogenetically informative proteins: an example from the fungal kingdom. BMC Evol Biol. 2007;7:134. https://doi.org/10.1186/1471-2148-7-134 PMID: 17688684
11. Haji D, Vailionis J, Stukel M, Gordon E, Lemmon EM, Lemmon AR, et al. Lack of host phylogenetic structure in the gut bacterial communities of New Zealand cicadas and their interspecific hybrids. Sci Rep. 2022;12(1):20559. https://doi.org/10.1038/s41598-022-24723-3 PMID: 36446872
12. Yang I, Woltemate S, Piazuelo MB, Bravo LE, Yepez MC, Romero-Gallo J, et al. Different gastric microbiota compositions in two human populations with high and low gastric cancer risk in Colombia. Sci Rep. 2016;6:18594. https://doi.org/10.1038/srep18594 PMID: 26729566
13. Kusejko K, Kadelka C, Marzel A, Battegay M, Bernasconi E, Calmy A, et al. Inferring the age difference in HIV transmission pairs by applying phylogenetic methods on the HIV transmission network of the Swiss HIV Cohort Study. Virus Evol. 2018;4(2):vey024. https://doi.org/10.1093/ve/vey024 PMID: 30250751
14. Shaw LP, Wang AD, Dylus D, Meier M, Pogacnik G, Dessimoz C, et al. The phylogenetic range of bacterial and viral pathogens of vertebrates. Mol Ecol. 2020;29(17):3361–79. https://doi.org/10.1111/mec.15463 PMID: 32390272
15. Weisbecker V, Beck RMD, Guillerme T, Harrington AR, Lange-Hodgson L, Lee MSY, et al. Multiple modes of inference reveal less phylogenetic signal in marsupial basicranial shape compared with the rest of the cranium. Philos Trans R Soc Lond B Biol Sci. 2023;378(1880):20220085. https://doi.org/10.1098/rstb.2022.0085 PMID: 37183893
16. Rong Z, Cai J, Qiu J, Xu P, Garmire LX, Lian Q, et al. L2 normalization and geodesic distance for enhanced information preservation in visualizing. High-dimensional single-cell sequencing data. In: ACM-BCB 24. New York, NY, USA: ACM; 2024. https://doi.org/10.1145/3698587.3701361
17. Wooldridge JM. Applications of generalized method of moments estimation. J Econ Perspect. 2001;15(4):87–100. https://doi.org/10.1257/jep.15.4.87
18. Lange M, Zühlke D, Holz O, Villmann T, Mittweida SG. Applications of lp-norms and their smooth approximations for gradient based learning vector quantization. In: ESANN; 2014. p. 271–6.
19. Kendall M, Colijn C. Mapping phylogenetic trees to reveal distinct patterns of evolution. Mol Biol Evol. 2016;33(10):2735–43. https://doi.org/10.1093/molbev/msw124 PMID: 27343287
20. Markin A, Eulenstein O. Cophenetic median trees under the manhattan distance. In: ACM-BCB 17; 2017. p. 194–202. https://doi.org/10.1145/3107411.3107443
21. Markin A, Eulenstein O. Cophenetic median trees. IEEE/ACM Trans Comput Biol Bioinform. 2019;16(5):1459–70. https://doi.org/10.1109/TCBB.2018.2870173 PMID: 30222583
22. Sánchez-Charles D, Muntés-Mulero V, Carmona J, Solé M. Process model comparison based on cophenetic distance. Springer; 2016. p. 141–58. https://doi.org/10.1007/978-3-319-45468-9_9
23. Munch E, Stefanou A. The l∞-cophenetic metric for phylogenetic trees as an interleaving distance. Springer; 2019. p. 109–27. https://doi.org/10.1007/978-3-030-11566-1_5
24. Gorecki P, Markin A, Eulenstein O. Cophenetic distances: a near-linear time algorithmic framework. In: COCOON 2018. 2018. p. 168–79. https://doi.org/10.1007/978-3-319-94776-1_15
25. Vijendran S. Cophenetic distance in near-linear time. 2025. https://github.com/sriram98v/near-linear-cophenetic-distance
26. Wang B-F, Li C-Y. Fast algorithms for computing path-difference distances. IEEE/ACM Trans Comput Biol Bioinform. 2019;16(2):569–82. https://doi.org/10.1109/TCBB.2018.2790957 PMID: 29993953

27. Bryant D, Scornavacca C. An $O(n \log n)$ time algorithm for computing the path-length distance between trees. Algorithmica. 2019;81(9):3692–706. https://doi.org/10.1007/s00453-019-00594-5
28. Yule GU. A mathematical theory of evolution, based on the conclusions of Dr. JC Willis, FRS. Philos Trans Roya Soc Lond Ser B. 1925;213:21–87. https://doi.org/10.1098/rstb.1925.0002
29. Harding EF. The probabilities of rooted tree-shapes generated by random bifurcation. Adv Appl Probab. 1971;3(1):44–77. https://doi.org/10.2307/1426329
30. Markin A, Eulenstein O. Computing manhattan path-difference median trees: a practical local search approach. IEEE/ACM Trans Comput Biol Bioinform. 2019;16(4):1063–76. https://doi.org/10.1109/TCBB.2017.2718507 PMID: 28650824
31. Markin A, Eulenstein O. Efficient local search for Euclidean path-difference median trees. IEEE/ACM Trans Comput Biol Bioinform. 2019;16(4):1374–85. https://doi.org/10.1109/TCBB.2017.2763137 PMID: 29035224
32. Cardona G, Mir A, Rosselló F, Rotger L. The expected value of the squared cophenetic metric under the Yule and the uniform models. Math Biosci. 2018;295:73–85. https://doi.org/10.1016/j.mbs.2017.11.007 PMID: 29155134