Sketch, capture and layout phylogenies

Daniel H. Huson

Email: daniel.huson@uni-tuebingen.de

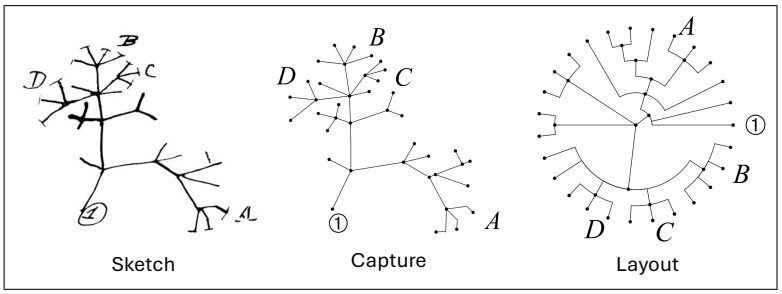
Institute for Bioinformatics and Medical Informatics,
University of Tübingen, 72076 Tübingen, Germany

Abstract

Phylogenetic trees and networks play a central role in biology, bioinformatics, and mathematical biology, and producing clear, informative visualizations of them is an important task. We present new algorithms for visualizing rooted phylogenetic networks as either combining or transfer networks, in both cladogram and phylogram style. In addition, we introduce a layout algorithm that aims to improve clarity by minimizing the total stretch of reticulate edges. To address the common issue that biological publications often omit machine-readable representations of depicted trees and networks, we also provide an image-based algorithm for extracting their topology from figures. All algorithms are implemented in our new PhyloSketch app, which is open source and freely available at: `https://github.com/husonlab/phylosketch2`.

Keywords: phylogenetics, rooted phylogenetic networks, network visualization, algorithm design, bioinformatics software, image-based phylogeny extraction

Graphical abstract



1 Introduction

Rooted phylogenetic networks are used to represent putative evolutionary scenarios in the presence of reticulate events such as horizontal gene transfer, hybrid speciation, or reassortment.

Several algorithms exist for computing such networks from biological data [Stolzer et al., 2012, Solís-Lemus and Ané, 2016, Huson and Linz, 2018, Wen et al., 2018, Allman et al., 2019, Barrat-Charlaix et al., 2022, Kong et al., 2024, Bernardini et al., 2024, Allman et al., 2025, Zhang et al.], typically producing a description in the extended Newick format [Cardona et al., 2008]. While any phylogenetic tree can be embedded in the plane without edge crossings in linear time, a rooted phylogenetic network may not be planar. In such cases, the computational problem arises of finding an embedding that minimizes the number of edge crossings caused by reticulation. This problem is known to be NP-hard, even for binary networks interpreted as transfer networks [Klawitter and Stumpf, 2020].

While there are many widely-used tools for interactively drawing and editing phylogenetic trees [Rambaut, 2009, Robinson et al., 2016], there are few that also target phylogenetic networks [Huson and Scornavacca, 2016, Vaughan, 2017, Solís-Lemus et al., 2017, Huson and Bryant, 2024].

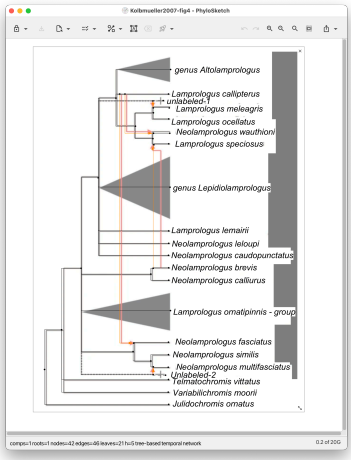
Dedicated algorithms are required to visualize rooted phylogenetic networks. Here, we present backbone-tree-based algorithms for computing embeddings of rooted phylogenetic networks, targeting combined and transfer cladograms and phylograms (see Fig. 1). We also provide a heuristic aimed at minimizing the stretch of reticulation edges – a problem that is NP-hard in general, as it solves the Minimum Linear Arrangement Problem [Garey and Johnson, 1979]. In practice, we address it using a simulated annealing approach [Kirkpatrick et al., 1983].

We introduce the PhyloSketch App, an interactive program for sketching rooted trees and networks, computing their layout using the new algorithms, and importing/exporting in extended Newick format.

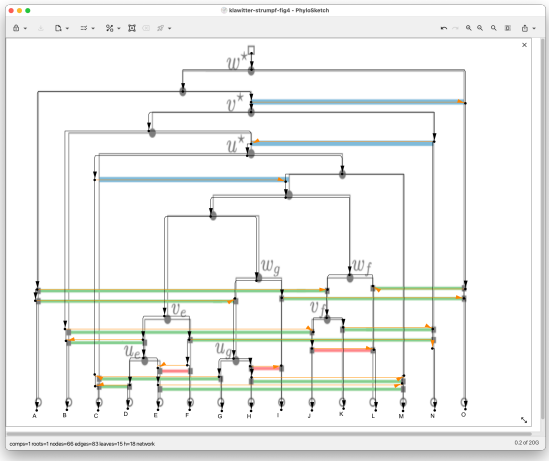
We also describe a simple image-based workflow implemented in PhyloSketch that helps users extract rooted phylogenetic trees and networks from figures. This addresses a common problem in bioinformatics: biology publications often fail to provide a machine-readable description of the topology of depicted trees or networks (see Fig. 1a). Previous approaches [Laubach et al., 2012, Lee et al., 2014] have been limited to phylogenetic trees.

All algorithms are implemented in the PhyloSketch App, which is open source and available at: `https://github.com/husonlab/phylosketch2`.

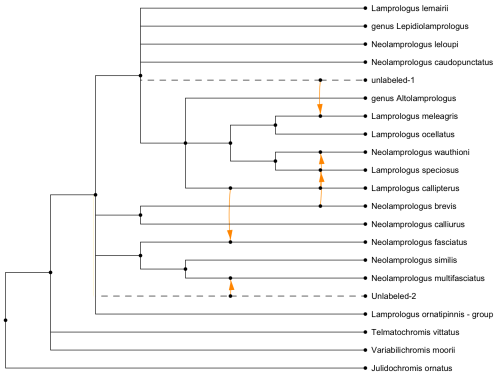
(a) Capture of fish hybridization network



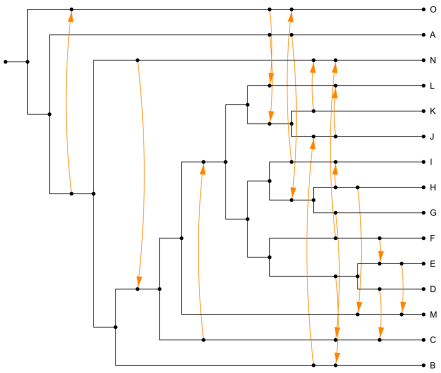
(d) Capture of layout example network



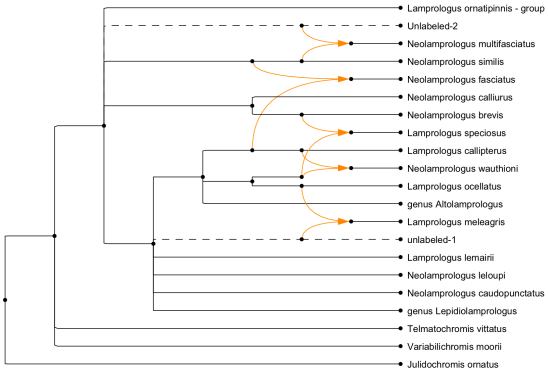
(b) Hybridization network as transfer cladogram



(e) Example network as transfer cladogram



(c) Hybridization network as combining cladogram



(f) Example network as combining cladogram

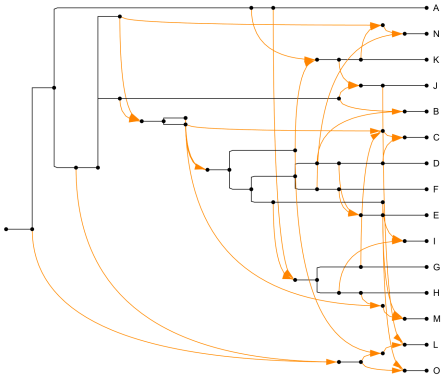


Figure 1: (a) PhyloSketch capture of a putative hybridization network for fish from Figure 4 of [Koblmueller et al., 2007]. The captured network displayed as (b) a transfer cladogram and (c) a combining cladogram; the latter is perhaps more suitable for representing hybridization. (d) PhyloSketch capture of an exemplary transfer network from Figure 4 of [Klawitter and Stumpf, 2020]. The transfer edges in the publication are undirected and so we arbitrarily selected a direction for each such edge. The captured network displayed as (e) a transfer cladogram and (f) a combining cladogram.

2 Results

The main results focus on the layout of rooted phylogenetic trees and networks.

2.1 Rooted phylogenetic networks and their layout

Let X be a set of taxa. A rooted phylogenetic network $N = (V, E, \rho, \lambda)$ on X is a directed graph with node set V, edge set $E \subseteq V \times V$, root node $\rho \in V$, and taxon labeling λ, such that [Huson et al., 2010]:

1. The graph N is a directed acyclic graph (DAG),
2. There is exactly one root node with in-degree 0, namely ρ (and thus, in particular, N is connected),
3. The labeling $\lambda : X \to V$ is a bijection between X and the set of leaves (nodes of out-degree 0),
4. There are no "through nodes" that have both in-degree 1 and out-degree 1, and
5. All leaves have in-degree ≤ 1.

A node v is called a *tree node* if it has in-degree ≤ 1, and a *reticulate node* otherwise. An edge $e = (v, w)$ is called a *tree edge* or *reticulate edge* if its target node w is a tree node or reticulate node, respectively.

We call a network *bifurcating* if all nodes v have out-degree $\deg^+(v) \leq 2$, and *bicombining* if all nodes v have in-degree $\deg^-(v) \leq 2$.

If there are no reticulate edges, then N is a tree, and thus planar—it can be drawn in the plane without edge crossings. In the presence of reticulate edges, N may be non-planar, and any drawing of N may necessarily involve crossings.

Viewing a rooted phylogenetic network as a generalization of the widely-used concept of a rooted phylogenetic tree, our goal when drawing a non-planar rooted phylogenetic network is to do so in such a way that *tree edges never cross each other*, while *reticulate edges may cross any edges*. To enhance visual clarity, we aim to reduce the number and extent of crossings in the drawing.

Klawitter and Stumpf [2020] investigate the problem of minimizing the number of crossings in bifurcating and bicombining rooted phylogenetic networks. They assume that the network N is tree-based (i.e., possesses a spanning tree T, which is assumed to be provided as part of the input), and study theoretical properties of different ways to draw the network in a "transfer view" (as defined below).

Here, we consider the more general case in which networks are not necessarily bifurcating or bicombining, and reticulations can be displayed either in a *combining view*, a *transfer view*, or using a mixture of both.

Rather than explicitly trying to minimize the number of crossings, we instead aim to minimize the total vertical *stretch* of reticulate edges in a left-to-right drawing of the network.

2.2 Combining view

There are two distinct ways to interpret reticulate nodes in a rooted phylogenetic network. In the *combining view*, a reticulate node represents a *combining event*, such as *hybridization-by-speciation*, where all incoming edges are treated equally and rendered in a similar fashion (see Fig. 1c,f). In contrast, the *transfer view* represents a *transfer event*, such as *horizontal gene transfer*. In this case, one incoming edge –the *transfer-acceptor edge*– represents the main lineage and is drawn like a regular tree edge, while all other incoming edges represent transferred material and are drawn as reticulate edges (see Fig. 1d,e).

While the visualization of a rooted phylogenetic network can easily accommodate the simultaneous occurrence of both combining and transfer nodes, when drawing cladograms, two different strategies appear to lead to better results depending on whether the network mainly contains combining or transfer nodes.

In this section, we focus on the *combining view*. A later section will describe how to adapt the algorithm to support the *transfer view*. We consider drawing rooted networks in a *left-to-right layout*—from the root on the left to the leaves on the right—and how to optimize such drawings.

We begin with the case of a *cladogram*, where edge lengths are ignored, and later extend the approach to *phylograms*, which incorporate edge weights. Finally, we discuss how to adapt the algorithm to obtain a *circular layout* of the network. All algorithms described here use the following concept to calculate y-coordinates.

Backbone tree For any reticulate node v in N, we define the *lowest stable ancestor* $LSA(v)$ to be the last node on all paths from the root ρ to v. Consider the following operation on N: for each reticulate node v, set $u = LSA(v)$, delete all edges entering v, and create a new edge from u to v. Note that application of this operation to all reticulate nodes gives rise to a rooted tree $B(N)$ whose root is ρ and whose leaf set contains the leaf set of N, possibly along with other unlabeled leaves. Also note that the tree may contain through nodes. We call $B(N)$ the *backbone tree* of N.

We say that a node u in $B(N)$ is *open* if and only if there is a reticulate edge in N that connects a descendant of u in $B(N)$ with some other node that is not a descendant of u in $B(N)$.

Combining cladogram algorithm We compute a left-to-right layout of the backbone tree $B(N)$ using a post-order traversal. If the node v that we are visiting is a leaf, we assign the coordinates $(x(v), y(v)) = (0, n)$,

where n is the number of leaves visited so far. Otherwise, v is an internal node, and we assign the coordinates $(x(v), y(v)) = (k-1, a)$, where $k = x(w)$ is the smallest x-coordinate assigned to any child w of v, both in $B(N)$ and in N. (Taking the children in N into account ensures that reticulate edges always go from left to right in the drawing.)

Once the coordinates have been computed, we obtain a left-to-right rooted network cladogram K for N by drawing a path for each edge $e = (v, w)$ from $(x(v), y(v))$ to $(x(w), y(w))$, via $(x(v), y(w))$. We use a rectangular path for a tree edge and a quadratic curve for a reticulate edge, for example.

2.3 Layout optimization

We define the cost of a left-to-right drawing as

$$\text{cost}(K) = \sum_{e=(v,w)\in R} |y(v) - y(w)|,$$

where $R \subseteq E$ is the set of reticulate edges (but excluding transfer-acceptor edges, when introduced later). The cost corresponds to the total *vertical stretch* $|y(v) - y(w)|$ of all reticulate edges in the drawing.

This cost depends solely on the y-coordinates assigned to nodes during the post-order traversal of the backbone tree $B(N)$, which in turn depend on the order in which the children of each *open* node v are visited. To optimize the layout, we must therefore consider all possible permutations of the children below each open node in $B(N)$.

NP-hardness We now argue that this layout optimization problem is NP-hard by reducing an instance of the *Minimum Linear Arrangement* (MinLA) problem to it.

An instance of MinLA consists of an undirected graph $G = (V, E)$ with $n = |V|$ vertices, and the goal is to find a bijective assignment $f : V \to \{1, \ldots, n\}$ that minimizes the total edge cost

$$\text{cost}(f) = \sum_{(u,v)\in E} |f(u) - f(v)|.$$

To reduce this to our layout optimization problem, we extend G to a rooted phylogenetic network N as follows: We introduce a root node ρ and create an edge from ρ to every original node $v \in V$. The original edges are considered reticulate edges. The new edges makeup the backbone tree $B(N)$.

In this construction, the total vertical stretch of the reticulate edges corresponds exactly to the MinLA cost function applied to the layout determined by the order of children below ρ. Thus, any solution to our

layout optimization problem for this network instance provides a solution to the MinLA instance, and vice versa. Since MinLA is NP-hard [Garey and Johnson, 1979], it follows that minimizing total vertical stretch in our problem is also NP-hard.

Heuristic optimization. To address this optimization in practice, we use a pre-order traversal of $B(N)$. For each node v that is the lowest stable ancestor (LSA) of some reticulate node, we determine a permutation of the children of v that minimizes the total vertical stretch.

If the number of children is at most eight, we exhaustively enumerate all permutations. Otherwise, we perform a heuristic search by iteratively swapping pairs of children, using simulated annealing [Kirkpatrick et al., 1983] to escape local minima.

Note: The optimal orderings at different nodes in the backbone tree are not independent when one node is an ancestor of another. However, our current implementation ignores this dependency and treats each node independently during optimization.

2.4 Transfer view

Let N be a rooted phylogenetic network on X. In the absence of additional information, we assume by default that every reticulate node is a *combining node*, to be displayed in the combining view.

A reticulate node v is called a *transfer node* if exactly one of its incoming edges is designated as the *transfer-acceptor edge*, and the remaining incoming edges are called *transfer edges*. (For example, in the extended Newick format, the transfer-acceptor edge is indicated using `##`, rather than `#`, for the corresponding incoming edge.)

Transfer cladogram algorithm To account for transfer nodes and their edges, we modify the construction of the backbone tree $B(N)$ as follows: Tree nodes and combining nodes are handled as in the standard approach. For each transfer node v, we retain only its designated transfer-acceptor edge and remove all other incoming edges. This adjustment ensures that any drawing of $B(N)$ avoids edge crossings among tree edges and transfer-acceptor edges.

In a left-to-right layout, we aim to draw transfer edges vertically whenever possible. To compute a transfer cladogram, we traverse the network to calculate the maximum number $\mu(v)$ of tree edges and transfer-acceptor edges on any path from the root to node v in N, after initially setting $\mu(v) = 0$ for all nodes. We use $\mu(v)$ as the x-coordinate of node v. The y-coordinates are computed using the post-order

approach described in Section 2.2, applied to the modified backbone tree $B(N)$, ignoring transfer-acceptor edges in the calculation of cost.

Because the traversal proceeds only in the forward direction, an additional refinement is performed to ensure that the source node v of a transfer edge $e = (v, w)$ is assigned the same x-coordinate as the target node w, whenever possible. For each such edge, we set

$$\mu(v) = \max\{\mu(v), \mu(w)\}$$

and then rerun the traversal starting with the updated values of μ. This adjustment can be repeated a few times to improve alignment.

2.5 Phylograms

So far, we have discussed drawing a cladogram, in which all leaves are placed on the right side of the drawing, all with the same horizontal coordinate 0, and any edge lengths, if given, are ignored.

To compute a *phylogram* of N, we assume that every tree or transfer-acceptor edge $e \in E$ is assigned a non-negative weight $\omega(e)$. The algorithm described here can be applied to both the combined view and the transfer view [Huson et al., 2010].

Combining and transfer phylogram algorithm We first use the approach described in Section 2.2 to compute all y-coordinates of the nodes, followed by the layout optimization strategy from Section 2.3.

To compute the x-coordinates, we assign the root ρ the coordinate $x(\rho) = 0$ and then perform a breadth-first traversal as follows:

Let v be a node for which all parent nodes in N have already been processed. There are two cases to consider:

- Tree or transfer node: Then v has an incoming edge $e = (u, v)$ that is either a tree edge or a transfer-acceptor edge and we set $x(v) = x(u) + \omega(e)$.
- Combining reticulation node: Then v is a combining node with $k \geq 2$ incoming reticulate edges $e_1 = (u_1, v)$, ..., $e_k = (u_k, v)$, and we set $x(v) = \max\{x(u_1), \ldots, x(u_k)\} + \delta$, where δ is a small, fixed value that is used to ensure that reticulate edge always go from left-to-right.

2.6 Circular layout

A circular cladogram or phylogram can be derived from a left-to-right layout by converting y-coordinates, which range from 0 to $n-1$, into angles ranging from 0° to $\frac{n-1}{360}^\circ$, and by computing radii instead of x-coordinates.

Optimization of the circular drawing can be achieved by optimizing the underlying left-to-right layout, as described above, using a slightly modified cost function: For each reticulate edge $e = (v, w)$, the cost is defined as

$$\min\{|y(v) - y(w)|,\ H - |y(v) - y(w)|\},$$

where H is the number of leaves in $B(N)$, plus 1. This accounts for the fact that in a circular layout, the shortest path between two nodes may go around the circle in either direction.

The impact of this adjustment is shown in Figure 2: the optimal orderings for the left-to-right and circular cladograms differ. However, the ordering used for the circular layout would also be acceptable for the linear layout.

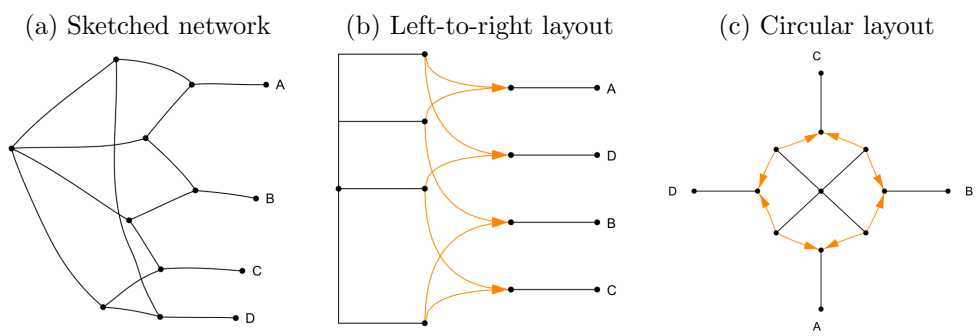


Figure 2: (a) A rooted phylogenetic network interactively constructed in PhyloSketch, shown (b) as a left-to-right cladogram and (c) as a circular cladogram.

2.7 Equal spacing of leaves

The cladogram and phylogram layout algorithms presented above suffer from the deficiency that leaves may not appear equally spaced, due to the presence of unlabeled leaves in the backbone tree $B(N)$.

To address this issue, once an optimal layout has been determined, we perform the following post-processing step: We repeat the post-order traversal of the backbone tree $B(N)$ to recompute y-coordinates. This time, we assign integer values $0, 1, \ldots$ to the labeled leaves and assign intermediate fractional, equally spaced coordinates to the unlabeled leaves.

Other approaches also face this problem, as is evident for IcyTree in Fig. 3c.

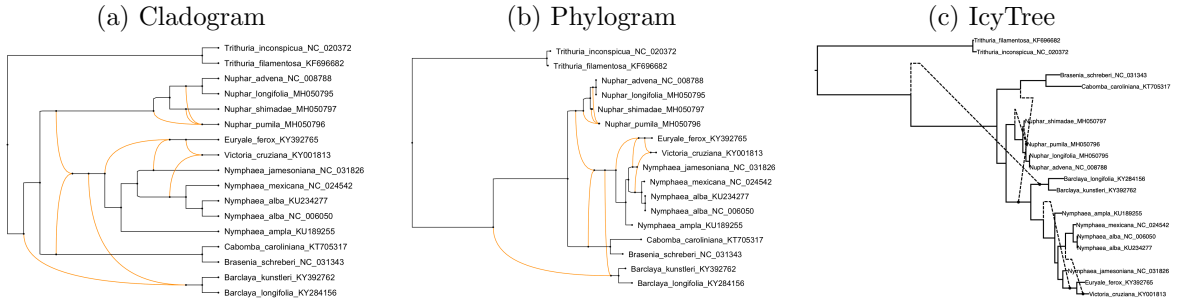


Figure 3: Using a rooted phylogenetic network computed by applying PhyloFusion [Zhang et al.] to 11 NADH dehydrogenase-associated gene trees of water lilies [Gruenstaeudl, 2019], we show (a) a cladogram and (b) a phylogram, computed using the described algorithms, and (c) a "time tree" visualization computed using IcyTree [Vaughan, 2017].

3 Materials and Methods

We provide implementations of all of the algorithms described above, supporting both left-to-right and circular layouts of cladograms and phylograms, including layout optimization.

3.1 PhyloSketch App

The *PhyloSketch* App is an interactive tool for working with phylogenetic trees and rooted networks. Trees and networks can be imported and exported using the extended Newick format. By default, reticulate nodes are interpreted as combining nodes; however, users can interactively declare transfer-acceptor edges, thereby reinterpreting the corresponding reticulate nodes as transfer nodes.

PhyloSketch also supports the interactive construction and editing of trees and networks. When a user draws a path using a mouse or touch screen, the software attempts to interpret the path as an edge, connecting its start point to a nearby node or an interior point of an existing edge, or its end point to another nearby node or interior point, if possible. A proposed path is accepted as a new edge only if it will not introduce a directed cycle into the network.

The program additionally provides a range of operations for modifying the topology of the tree or network, as well as tools for styling and labeling nodes and edges.

3.2 Network capture

The *PhyloSketch* App includes a workflow that assists users in capturing a rooted phylogenetic tree or network directly from an image (see Fig. 1a,d).

To capture a phylogenetic tree or network, the user loads an image into the program and selects the location of the root. The workflow then proceeds as follows:

1. The Tesseract library [Smith, 2007] is used to detect labels in the image.
2. A skeletonization algorithm [Zhang and Suen, 1984] reduces all lines to single-pixel width.
3. All nodes (i.e., branching points and endpoints of lines) are detected using a pixel mask.
4. A flood-fill algorithm is applied to identify all paths connecting pairs of nodes.
5. Paths that lie entirely within the bounding box of a detected label are removed.
6. A root node is created at the chosen root location, splitting any nearby path if necessary.
7. The layout orientation, such as left-to-right, top-to-bottom, or radial, is determined by comparing the indicated root location with the positions of detected nodes.
8. Detected labels are associated with nearby nodes, taking the root location into account.
9. In order of increasing distance from nodes added so far, each remaining path is considered a potential new edge, and an attempt is made to incorporate it, as described in the previous section.

This approach assumes that the tree or network is drawn in dark colors on a light background, that edges are rendered as reasonably thin lines, and that nodes are not represented by large disks, for example.

Several post-processing steps are available to clean up the captured phylogeny: through nodes can be removed; nodes of in-degree two and out-degree two can be replaced by crossing edges; edge directions can be reversed; and sets of nodes can be merged into a single node while maintaining labels and connectivity.

4 Discussion

While many programs and packages exist for computing, constructing, and editing phylogenetic trees, and to a lesser extent, phylogenetic networks, to the best of our knowledge, *PhyloSketch* is the first program that

allows users to interactively create a phylogenetic tree or network from scratch, or to capture a phylogenetic network from an image using image processing algorithms.

The algorithm proposed for computing a left-to-right cladogram builds on the approach that described in [Huson et al., 2010] and is implemented in *Dendroscope* [Huson and Scornavacca, 2016], but it differs in how the layout cost is computed and in the use of simulated annealing for optimization. Our new algorithm for computing a transfer cladogram improves on the earlier version implemented in *Dendroscope*, as it explicitly aims to place the source and target of transfer edges on the same level whenever possible.

Using the backbone tree to compute and optimize network layouts provides a useful simplification. However, this approach can miss "obvious" improvements that would require rearrangements not represented in the backbone tree. For example, in Figure 3a, switching the subtree containing the two lowest taxa (*B. kunstleri* and *B. longifolia*) with the one above it (containing *C. caroliniana* and *B. schreberi*) would significantly reduce the stretch of the two associated reticulation edges. However, this rearrangement is never considered since the subtrees are not attached to the same node in the backbone tree.

Much effort was invested in designing a simple yet powerful user interface for *PhyloSketch*, which should also work well on mobile devices. Future work may include extending the layout and capture algorithms to support other classes of networks. We also plan to expand the image-based capture workflow to support batch processing and to use AI for post-processing.

Acknowledgments

The author is thankful to Elizabeth Allman and Mike Steel for many helpful discussions.

This material is based upon work supported by the National Science Foundation under Grant No. DMS-1929284 while the author was in residence at the Institute for Computational and Experimental Research in Mathematics in Providence, RI, during the "Theory, Methods, and Applications of Quantitative Phylogenomics" program.

This work was also supported by the NZ Marsden Fund (23-UOC-003) during a visit to New Zealand.

Supplementary Materials

The following supplementary materials are available:

1. Extended Newick encodings of all networks shown in the paper.

2. User manual for the *PhyloSketch* App.

References

E.S. Allman, H. Baños, and J.A. Rhodes. NANUQ: A method for inferring species networks from gene trees under the coalescent model. *Algorithms Mol. Biol.*, 14(24):1–25, 2019.

E.S. Allman, H. Baños, J.A. Rhodes, and Kristina Wicke. NANUQ$^+$: A divide-and-conquer approach to network estimation, 2025. Note appear in: Algorithms for Molecular Biology.

P. Barrat-Charlaix, T. Vaughan, and R. Neher. TreeKnit: Inferring ancestral reassortment graphs of influenza viruses. *PLOS Computational Biology*, 18(8):1–19, 08 2022. doi: 10.1371/journal.pcbi.1010394.

G. Bernardini, L. van Iersel, E. Julien, and L. Stougie. Inferring phylogenetic networks from multifurcating trees via cherry picking and machine learning. *Molecular Phylogenetics and Evolution*, 199:108137, 2024. doi: https://doi.org/10.1016/j.ympev.2024.108137.

G. Cardona, F. Rosselló, and G. Valiente. Extended Newick: it is time for a standard representation of phylogenetic networks. *BMC Bioinformatics*, 9:532, 2008. doi: 10.1186/1471-2105-9-532.

M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* A Series of Books in the Mathematical Sciences. W. H. Freeman, 1979.

M. Gruenstaeudl. Why the monophyly of nymphaeaceae currently remains indeterminate: an assessment based on gene-wise plastid phylogenomics. *Plant Systematics and Evolution*, 305:827–836, 2019.

D. H. Huson and D. Bryant. The SplitsTree app: interactive analysis and visualization using phylogenetic trees and networks. *Nature Methods*, 2024.

D. H. Huson and S. Linz. Autumn algorithm—computation of hybridization networks for realistic phylogenetic trees. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 15:398–420, 2018.

D. H. Huson and C. Scornavacca. Dendroscope 3: An interactive tool for rooted phylogenetic trees and networks. *Systematic Biology*, 65(3):543–546, 2016. doi: 10.1093/sysbio/syw020.

D. H. Huson, R. Rupp, and C. Scornavacca. *Phylogenetic Networks.* Cambridge, 2010.

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598): 671–680, 1983. doi: 10.1126/science.220.4598.671.

J. Klawitter and P. Stumpf. Drawing tree-based phylogenetic networks with minimum number of crossings. In *Graph Drawing and Network Visualization (GD 2020)*, volume 12590 of *Lecture Notes in Computer Science*, pages 173–180. Springer, 2020.

S. Koblmueller, N. Duftner, K. M. Sefc, M. Aibara, M. Stipacek, M. Blanc, B. Egger, and C. Sturmbauer. Reticulate phylogeny of gastropod-shell-breeding cichlids from Lake Tanganyika–the result of repeated introgressive hybridization. *BMC Evolutionary Biology*, 7:7, 2007. doi: 10.1186/1471-2148-7-7. URL `https://doi.org/10.1186/1471-2148-7-7`.

Sungsik Kong, David L Swofford, and Laura S Kubatko. Inference of phylogenetic networks from sequence data using composite likelihood. *Systematic Biology*, 74(1):53–69, 10 2024. ISSN 1063-5157. doi: 10.1093/sysbio/syae054. URL `https://doi.org/10.1093/sysbio/syae054`.

T. Laubach, A. von Haeseler, and M. J. Lercher. TreeSnatcher plus: capturing phylogenetic trees from images. *BMC Bioinformatics*, 13(1):110, 2012. doi: 10.1186/1471-2105-13-110.

P. Lee, S. T. Yang, J. D. West, and B. Howe. PhyloParser: A hybrid algorithm for extracting phylogenies from dendrograms. In *2014 IEEE International Conference on Big Data (Big Data)*, pages 49–56. IEEE, 2014. doi: 10.1109/BigData.2014.7004220.

A. Rambaut. FigTree v1.4: Tree figure drawing tool, 2009. URL `http://tree.bio.ed.ac.uk/software/figtree/`.

O. Robinson, D. Dylus, and C. Dessimoz. Phylo.io: interactive viewing and comparison of large phylogenetic trees on the web. *Molecular Biology and Evolution*, 33(8):2163–2166, 2016. URL `https://phylo.io/`.

R. Smith. An overview of the Tesseract OCR engine. In *Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR)*, pages 629–633. IEEE, 2007.

C. Solís-Lemus and C. Ané. Inferring phylogenetic networks with maximum pseudolikelihood under incomplete lineage sorting. *PLoS Genetics*, 12(3):e1005896, 2016. doi: 10.1371/journal.pgen.1005896.

C. Solís-Lemus, P. Bastide, and C. Ané. PhyloNetworks: A package for phylogenetic networks. *Molecular Biology and Evolution*, 34(12):3292–3298, 2017. doi: 10.1093/molbev/msx235.

M. Stolzer, H. Lai, M. Xu, D. Sathaye, B. Vernot, and D. Durand. Inferring duplications, losses, transfers and incomplete lineage sorting with nonbinary species trees. *Bioinformatics*, 28(18):i409–i415, 2012.

T. G. Vaughan. IcyTree: Rapid browser-based visualization for phylogenetic trees and networks. *Bioinformatics*, 2017. doi: 10.1093/bioinformatics/btx155.

D. Wen, Y. Yu, M. W. Hahn, and L. Nakhleh. Inferring phylogenetic networks using PhyloNet. *Systematic Biology*, 67(4):735–740, 2018. doi: 10.1093/sysbio/syy015.

L. Zhang, B. Cetinkaya, and D. H. Huson. PhyloFusion—fast and easy fusion of rooted phylogenetic trees into rooted phylogenetic networks. Manuscript under review.

T. Y. Zhang and C. Y. Suen. A fast parallel algorithm for thinning digital patterns. *Communications of the ACM*, 27(3):236–239, 1984.