Drawing Phylogenetic Trees*
(Extended Abstract)

Christian Bachmaier, Ulrik Brandes, and Barbara Schlieper

Department of Computer & Information Science, University of Konstanz, Germany
{christian.bachmaier,ulrik.brandes,barbara.schlieper}@uni-konstanz.de

Abstract. We present linear-time algorithms for drawing phylogenetic trees in radial and circular representations. In radial drawings given edge lengths (representing evolutionary distances) are preserved, but labels (names of taxons represented in the leaves) need to be adjusted, whereas in circular drawings labels are perfectly spread out, but edge lengths adjusted. Our algorithms produce drawings that are unique solutions to reasonable criteria and assign to each subtree a wedge of its own. The linear running time is particularly interesting in the circular case, because our approach is a special case of Tutte's barycentric layout algorithm involving the solution of a system of linear equations.

1 Introduction

Phylogeny is the study of the evolutionary relationships within a group of organisms. A phylogenetic tree represents the evolutionary distances among the organisms represented by its leaves. Due to the increasing size of data sets, drawings are essential for exploration and analysis. In addition to the usual requirements for arbitrary tree structures, drawings of phylogenetic trees should also depict given edge lengths and leaf names. Standard approaches [3,12,18] do not take these criteria into account (see [2,7] for an overview of tree drawing algorithms). Popular software tools in computational biology such as TreeView [10], PAUP* [14], or PHYLIP [11] also provide drawings of phylogenetic trees, but the underlying algorithms are not documented.

There are essentially two forms of representation for phylogenetic trees. For an overview see, e. g., [1]. Both are variations of dendrograms, since many algorithms for the construction of phylogenetic trees are based on clustering (see, e. g., [15]). They differ in that leaf labels are either placed monotonically along one axis or around the tree structure. While the first class of representations is very similar to standard dendrograms and easy to layout, it is somewhat difficult to understand the nesting of subtrees from the resulting drawings. We focus on the algorithmically more challenging and graphically more appealing second class of representations.

In radial tree drawings edges extend radially monotonic away from the root, and we give a linear-time algorithm that preserves all edge lengths exactly. In

* Partially supported by DFG under grant Br 2158/1-2.

circular tree drawings leaves are placed equidistantly on the perimeter of a circle and the tree is confined to the inside of the circle. Note that it may not be possible to preserve edge lengths in this representation. We give an algorithm that heuristically minimizes length deviations and, even though based on solving a system of linear equations, runs in linear time as well. Both algorithms yield drawings that are unique in a well-defined sense up to scaling, rotation, and translation. Since each subtree is confined to a wedge rather than an interval of its own, their nesting structure is more apparent than in vertical or horizontal representations. While, typically, phylogenetic trees are cases extended binary trees, our algorithms apply to general trees.

This paper is organized as follows. Basic notation and some background is provided in Sect. 2. In Sects. 3 and 4 we present our algorithms for radial and circular representations, and conclude in Sect. 5.

2 Preliminaries

Throughout the paper let $T = (V, E, \delta)$ denote a phylogenetic tree with $n = |V|$ vertices, $m = |E|$ edges, and positive edge lengths $\delta\colon E \to \mathbb{R}^+$. The leaves of a phylogenetic tree represent the species, molecules, or DNA sequences (*taxons*) under study and its inner vertices represent virtual or hypothetical ancestors. The length of an edge represents the evolutionary divergence between its incident vertices, and the entire tree represents a tree metric fitted to a (potentially noisy and incomplete) dissimilarity matrix defined over all taxons. Since we want the length of an edge $e \in E$ to resemble $\delta(e)$ as closely as possible, only positive values are allowed. If a method for tree construction, e. g., [5, 6, 9, 13], assigns negative or zero length to an edge, we set it to a small positive constant, e. g., to a fraction of the smallest positive edge length in the tree.

Let $\deg\colon V \to \mathbb{N}$ denote the number of edges incident to a vertex and note that typical tree reconstruction methods yield rooted trees in which most inner vertices have two children. We use $\mathsf{root}(T)$ to refer to the root of a tree T. Each vertex $v \in V \setminus \{\mathsf{root}(T)\}$ has an unique parent $\mathsf{parent}(v)$, and we denote the set of children by $\mathsf{children}(v)$. For a vertex $v \in V$ let $T(v)$ be the induced subtree, i. e., the subtree of all descendants of v (including v itself). Clearly, $T(\mathsf{root}(T)) = T$. For a subtree $T(v)$ of T we use $\mathsf{leaves}\,(T(v))$ containing all vertices v in $T(v)$ which have $\deg(v) = 1$ in T to denote the set of its leaves. Tree edges are directed away from the root, i. e., for an edge (u, v) we have $u = \mathsf{parent}(v)$. We sometimes use $\{u, v\}$ to refer to the underlying undirected edge.

Finally, we assume that in ordered trees the outgoing edges are to be drawn in counterclockwise order, i. e., for an inner vertex other than the root the counterclockwise first edge after the incoming edge is that of the first child.

3 Radial Drawings

In this section, we describe a drawing algorithm that yields a planar radial drawing of a phylogenetic tree $T = (V, E, \delta)$. It is $\mathcal{NP}$-complete to decide whether a

graph can be drawn in the plane with prescribed edge lengths, even if the graph is planar and all edges have unit lengths [4], but for trees we can represent any assignment of positive edge lengths exactly.

3.1 Basic Algorithm

The main idea is to assign to each subtree $T(v)$ a wedge of angular width proportional to the number of leaves in $T(v)$. The wedge of an inner vertex is divided among its children, and tree edges are drawn along wedge angle bisectors, so that they can have any length without violating disjointness. See Fig. 1 for illustration. Algorithm 1 therefore traverses the input tree twice:

- in a postorder traversal, the number $l_v = |\mathsf{leaves}(T_v)|$ of leaves in each subtree $T(v)$ is determined, and
- in a subsequent preorder traversal, a child w of an inner vertex v is placed at distance $\delta(v, w)$ on the angular bisector of the wedge reserved for w.

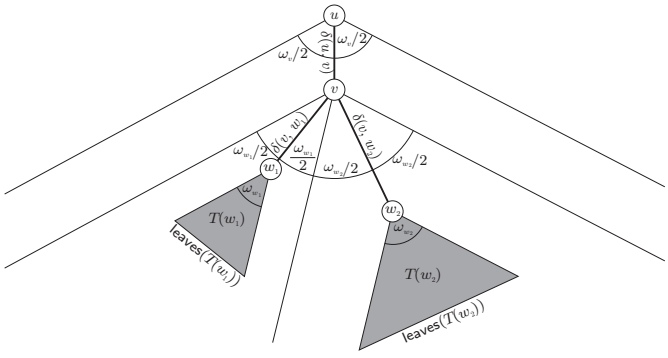


Fig. 1. Wedges of vertex v's neighbors

The following theorem shows that the layouts determined by Algorithm 1 are essentially the only ones that fulfill all natural requirements for radial drawings of trees with given edge lengths.

Theorem 1. *For an unrooted ordered phylogenetic tree $T = (V, E, \delta)$, there is a unique planar radial drawing up to rotation, translation and scaling, that satisfies the following properties:*

1. *Relative edge lengths are preserved exactly.*
2. *Disjoint subtrees are confined to disjoint wedges.*
3. *Subtrees are centered at the bisectors of their wedges.*
4. *The angular width of the wedge of a subtree is proportional to the number of leaves in that subtree.*

Moreover, it can be computed in linear time.

Algorithm 1: RADIAL-LAYOUT

Input: Rooted tree $T = (V, E, \delta)$
Data: Vertex arrays l (number of leaves in subtree), ω (wedge size), and τ (angle of right wedge border)
Output: Coordinates x_v for all $v \in V$

```
begin
    postorder_traversal(root(T))
    x_root(T) ← (0, 0)
    ω_root(T) ← 2π
    τ_root(T) ← 0
    preorder_traversal(root(T))
end

procedure postorder_traversal(vertex v)
    if deg(v) = 1 then
        l_v ← 1
    else
        l_v ← 0
        foreach w ∈ children(v) do
            postorder_traversal(w)
            l_v ← l_v + l_w

procedure preorder_traversal(vertex v)
    if v ≠ root(T) then
        u ← parent(v)
        x_v ← x_u + δ(u, v) · (cos(τ_v + ω_v/2), sin(τ_v + ω_v/2))
    η ← τ_v
    foreach w ∈ children(v) do
        ω_w ← l_w / l_root(T) · 2π
        τ_w ← η
        η ← η + ω_w
        preorder_traversal(w)
```

Proof. Choose any vertex as the root. By Property 4, the angular width of the wedge of a subtree $T(v)$ is

$$\omega_v = \alpha \cdot \frac{|\mathsf{leaves}\,(T(v))\,|}{|\mathsf{leaves}(T)|} \quad . \tag{1}$$

If the root is altered such that $u = \mathsf{parent}(v)$ becomes a child of v in the newly rooted tree T', then

$$\frac{|\mathsf{leaves}\,(T'(u))\,|}{|\mathsf{leaves}(T')|} = \frac{|\mathsf{leaves}(T)| - |\mathsf{leaves}\,(T(v))\,|}{|\mathsf{leaves}(T)|} \quad , \tag{2}$$

so that the proportionality factor $\alpha = 2\pi$ and Properties 4, 3 and 2 imply that angles between incident edges are independent of the actual choice of the root. Since relative lengths of edges need to be preserved as well, layouts are unique up to rotation, translation and scaling. Planarity is implied. Clearly, Algorithm 1 determines the desired layouts in linear time. □

3.2 Extensions

Labels of leaves are placed on the angle bisector of the respective wedge. Since the angle of a leaf wedge is $\frac{2\pi}{|\mathsf{leaves}(T)|}$, labels placed close to their leaf may not fit into the wedge. When using a font of height h, non-overlapping labels are guaranteed if they are placed at distance at least

$$\frac{h}{2 \cdot \tan\left(\pi \,/\, |\mathsf{leaves}(T)|\right)} \tag{3}$$

from the parent of their associated leaf.

While the number of leaves is a good indicator of how much angular space is required by a subtree, other scaling schemes can be used to emphasize different aspects such as height, size, or importance of subtrees.

Since child orders are respected by the algorithm, we may sort children according to, say, the size of their subtrees in a preprocessing step. This serves to modify the general appearance of the final layout.

While the confinement of subtrees to wedges of their own nicely separates them, it also results in poor angular resolution and a fair amount of wasted drawing space. We may thus wish to relax this requirement and increase angles between incident edges where possible. This can be achieved during post-processing using a bottom-up traversal, in which the angle between outgoing edges of a vertex v are scaled to the maximum value possible within the wedge of v rooted at $\mathsf{parent}(v)$ (see Fig. 2). Note that labels need to be be taken into account in this angular spreading step, and that the resulting drawings depend on the choice of root. Our experiments suggest that placing the root at the center of the tree yields favorable results.

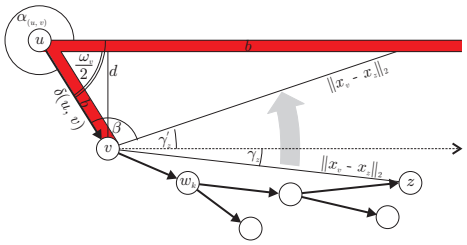


Fig. 2. Increasing angles to fill empty strips around subtrees

4 Circle Drawings

Since it can be difficult to place labels in radial tree drawings, we describe a second method to draw phylogenetic trees. Here, leaves are placed equidistantly along the perimeter of a circle. Again, each leaf thus obtains a wedge of angular width $\frac{2\pi}{|\mathsf{leaves}(T)|}$, but now the radius of the circle determines the maximum height of the font according to (3). It is easy to see that, with this constraint, it might not be possible to draw all edges $e \in E$ with length proportional to $\delta(e)$. Edge length preservation therefore turns into an optimization criterion.

4.1 Basic Algorithm

We use a variant of the weighted version of Tutte's barycentric layout algorithm [16, 17]. The general idea is to fix some vertices to the boundary of a convex polygon and place all other vertices in the weighted barycenter of their neighbors, i. e. v_i is positioned at $x_i = \sum_{v_j \in V}(a_{ij}x_j)$ where a_{ij} is the relative influence of v_j on v_i.

For circular drawings, the leaves of a tree T are fixed to a circle and we define weights a_{ij} by (4). These weights reflect the desired edge lengths, i. e., the shorter an edge $\{v_i, v_j\}$ should be, the more influence has x_j on the resulting coordinate x_i. In the original Tutte algorithm $a_{ij} = \frac{1}{\deg(v_i)}$. Note that the weights a_{ij} sum up to 1 for each v_i, so that v_i is placed inside of the convex hull of its neighbors.

Since we fix leaves to the perimeter of a circle, we can expect in general that an inner vertex of a tree is placed between its children on the one side and its parent on the other side. To counterbalance the accumulated radial influence of the children, their weight is scaled down.

$$
\begin{aligned}
s_{ij} &= \begin{cases} \dfrac{1}{\delta(v_i, v_j) \cdot (\deg(v_i) - 1)} & \text{if } v_i = \mathsf{parent}(v_j), \\ \dfrac{1}{\delta(v_i, v_j)} & \text{if } v_j = \mathsf{parent}(v_i) \end{cases} \\
a_{ij} &= \begin{cases} \dfrac{s_{ij}}{\sum_{\{v_i, v_{j'}\} \in E} s_{ij'}} & \text{if } \{v_i, v_j\} \in E, \\ 0 & \text{otherwise} \end{cases}
\end{aligned}
\tag{4}
$$

In the following, let $V = \{v_1, \ldots, v_n\}$ with $\mathsf{leaves}(T) = \{v_1, \ldots, v_k\}$ for $k = \lceil \frac{n}{2} \rceil$. After fixing the leaves equidistantly around the circle, we need to solve

$$
\begin{pmatrix} a_{k+1,1} & \ldots & a_{k+1,n} \\ \vdots & & \vdots \\ a_{n,1} & \ldots & a_{n,n} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ \\ \vdots \\ \\ x_n \end{pmatrix} = \begin{pmatrix} x_{k+1} \\ \vdots \\ x_n \end{pmatrix} . \tag{5}
$$

By the following lemma, this can be done in linear time by traversing the tree first in postorder to resolve the influence of leaves and then in preorder passing down positions of parents.

Lemma 1. *For $v_i \in V$ and $v_p = \mathsf{parent}(v_i)$ define coefficients*

$$c_i = \begin{cases} 0 & \text{if } v_i \in \mathsf{leaves}(T) \cup \{\mathsf{root}(T)\}, \\ \dfrac{a_{pi}}{1 - \sum_{(v_i,v_j)\in E}(a_{ij}c_j)} & \text{otherwise} \end{cases} \tag{6a}$$

and offsets

$$d_i = \begin{cases} x_i & \text{if } v_i \in \mathsf{leaves}(T), \\ \dfrac{\sum_{(v_i,v_j)\in E}(a_{ij}d_j)}{1 - \sum_{(v_i,v_j)\in E}(a_{ij}c_j)} & \text{otherwise .} \end{cases} \tag{6b}$$

Then, (5) has a unique solution with

$$x_i = \begin{cases} d_i & \text{if } v_i = \mathsf{root}(T), \\ c_i x_p + d_i & \text{otherwise} \end{cases} \tag{7}$$

for all inner vertices $v_i \in V \setminus \mathsf{leaves}(T)$.

Proof. We use induction over the vertices. For the base case let v_i be an inner vertex having only leaves as children. Then in case $v_i \neq \mathrm{root}(T)$ let v_p be the parent of v_i and thus

$$\begin{aligned} x_i &= a_{pi}x_p + \sum_{(v_i,v_j)\in E}(a_{ij}\underbrace{x_j}_{=d_j}) = \frac{a_{pi}}{1-0}x_p + \frac{\sum_{(v_i,v_j)\in E}(a_{ij}d_j)}{1-0} = \\ &= \frac{a_{pi}}{1 - \sum_{(v_i,v_j)\in E}(a_{ij}c_j)}x_p + \frac{\sum_{(v_i,v_j)\in E}(a_{ij}d_j)}{1 - \sum_{(v_i,v_j)\in E}(a_{ij}c_j)} = c_i x_p + d_i \ . \end{aligned} \tag{8}$$

The case $v_i = \mathrm{root}(T)$ is a special case of (8) which uses only the second addend. For the inductive step let $v_i \neq \mathrm{root}(T)$ be an inner vertex and v_p the parent of v_i. Then

$$\begin{aligned} x_i &= a_{pi}x_p + \sum_{(v_i,v_j)\in E}(a_{ij}x_j) \overset{\text{i. h.}}{=} a_{pi}x_p + \sum_{(v_i,v_j)\in E}(a_{ij}(c_j x_i + d_j)) = \\ &= a_{pi}x_p + x_i \sum_{(v_i,v_j)\in E}(a_{ij}c_j) + \sum_{(v_i,v_j)\in E}(a_{ij}d_j) = \\ &= \frac{a_{pi}}{1 - \sum_{(v_i,v_j)\in E}(a_{ij}c_j)}x_p + \frac{\sum_{(v_i,v_j)\in E}(a_{ij}d_j)}{1 - \sum_{(v_i,v_j)\in E}(a_{ij}c_j)} = c_i x_p + d_i \ . \end{aligned} \tag{9}$$

The proof for $v_i = \mathrm{root}(T)$ is a special case of (9) which uses only the second addend. □

Algorithm 2: CIRCLE-LAYOUT

Input: Ordered rooted tree $T = (V, E, \delta)$
Data: Vertex arrays c (coefficient), d (offset), and edge array s (weighting)
Output: Coordinates x_v in/on the unit circle for each vertex $v \in V$

begin
  $i \leftarrow 0$
  $k \leftarrow 0$
  foreach $v \in V$ **do**
    if $\deg(v) = 1$ **then** $k \leftarrow k + 1$
  postorder_traversal (root(T))
  preorder_traversal (root(T))
end

procedure postorder_traversal(vertex v)
  foreach $w \in$ *children*(v) **do**
    postorder_traversal(w) *// opt. ordered by $h(w) + \delta(v, w)$*
  if *is_leaf*(v) *or* $(v =$ *root*(T) *and* $\deg($*root*$(T)) = 1)$ **then**
    $c_v \leftarrow 0$
    $d_v \leftarrow \left(\cos\left(\frac{2\pi i}{k}\right), \sin\left(\frac{2\pi i}{k}\right)\right)$ *// fix vertex on circle*
    $i \leftarrow i + 1$
  else
    $S \leftarrow 0$
    foreach *adjacent edge* $e \leftarrow \{v, w\}$ **do**
      if $v =$ *root*(T) *or* $w =$ *parent*(v) **then**
        $s_e \leftarrow \frac{1}{\delta(e)}$
      else
        $s_e \leftarrow \frac{1}{\delta(e) \cdot (\deg(v) - 1)}$
      $S \leftarrow S + s_e$
    $t \leftarrow t' \leftarrow 0$
    foreach *outgoing edge* $e \leftarrow (v, w)$ **do**
      $t \leftarrow t + \frac{s_e}{S} \cdot c_w$
      $t' \leftarrow t' + \frac{s_e}{S} \cdot d_w$
    if $v \neq$ *root*(T) **then**
      $e \leftarrow (\text{parent}(v), v)$
      $c_v \leftarrow \frac{s_e}{S \cdot (1 - t)}$
    $d_v \leftarrow \frac{t'}{1 - t}$

procedure preorder_traversal(vertex v)
  if $v =$ *root*(T) **then**
    $x_v \leftarrow d_v$
  else
    $u \leftarrow \text{parent}(v)$
    $x_v \leftarrow c_v \cdot x_u + d_v$
  foreach $w \in$ *children*(v) **do**
    preorder_traversal(w)

Theorem 2. *For a rooted ordered phylogenetic tree $T = (V, E, \delta)$, there is a unique planar circle drawing up to rotation, translation, and scaling, that satisfies the following properties:*

1. *Leaves are placed equidistantly on the perimeter of a circle.*
2. *Disjoint subtrees are confined to disjoint wedges.*
3. *Inner vertices are placed in the weighted barycenter of their neighbors with weights defined by Eq. (4).*

Moreover, it can be computed in linear time.

The most general version of Tutte's algorithm for arbitrary graphs fixes at least one vertex of each component and simply places each vertex in the barycenter of its neighbors, which yields a unique solution. The running time corresponds to solving n symmetric equations, which can be done in $\mathcal{O}(n^3)$ time. For planar graphs $\mathcal{O}(n \log n)$ time can be achieved [8], but it is not known whether this is also a lower bound. We showed that for trees with all leaves in convex position, the running time is in $\mathcal{O}(n)$. Giving up planarity this result can be generalized to trees having arbitrary fixed vertices (at least one). Consider a fixed inner vertex v and let each other vertex w in the subtree $T(v)$ be free. Then $T(v)$ collapses to the position of v. On the other hand, if v has a fixed ancestor u, then all positions of vertices in $T(v)$ are computed and for the rest a restart on $T \backslash T(v) \cup \{v\}$ computes all remaining positions. It follows by induction that our algorithm computes in $\mathcal{O}(n)$ time the unique solution.

Note that the desirable property of disjoint subtree wedges together with the circular leaves constraint further restricts the class of edge lengths that can are represented exactly. Nevertheless, our experiments indicate that typical phylogenetic tree metrics are represented fairly accurately.

4.2 Extensions

Our weights depend on the choice of the root, since re-rooting the tree changes weights along the path between the previous and the new root. The rationale behind reducing the influence weights of children suggests that the tree should be rooted at its center (minimum eccentricity element). Using weights independent of the parent-child relation the layout can be made independent of the root just like the radial layouts discussed in the previous section.

It is easy to see that by fixing the order of leaves we are also fixing the child order of all inner vertices. If no particular order is given, we can permute the children of inner vertices to improve edge lengths preservation. Ordering the children of each vertex v according to ascending height $h(w)$ of the subtrees $T(w)$ plus $\delta(v, w)$ ensures that shallow and deep subtrees are never placed alternatingly. See Fig. 3. Though sorting the children leads to $\mathcal{O}(n \log n)$ preprocessing time in general, most phylogenetic trees have bounded degree, so that sorting can be performed in linear time.

Since correct edge lengths cannot be guaranteed in circle drawings, we use the following coloring scheme to depict the error: Let $\sigma = \frac{\sum_{e=(u,v)\in E} \|x_v - x_u\|_2}{\sum_{e \in E} \delta(e)}$

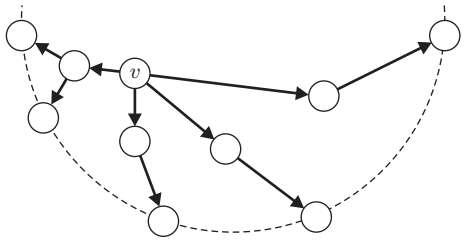


Fig. 3. Ordering children according to subtree hight supports postulated edge lengths

be the mean resolution of the drawing, i. e., the scaling factor between drawn units and length units of δ. Then we obtain the (multiplicative) error of an edge $e = (u, v)$ by $f_e = \frac{\|x_u - x_v\|_2}{\sigma \cdot \delta(e)}$, which we encode into a color $\mathrm{rgb}\colon \mathbb{R}^+ \to [0;1]^3$ by

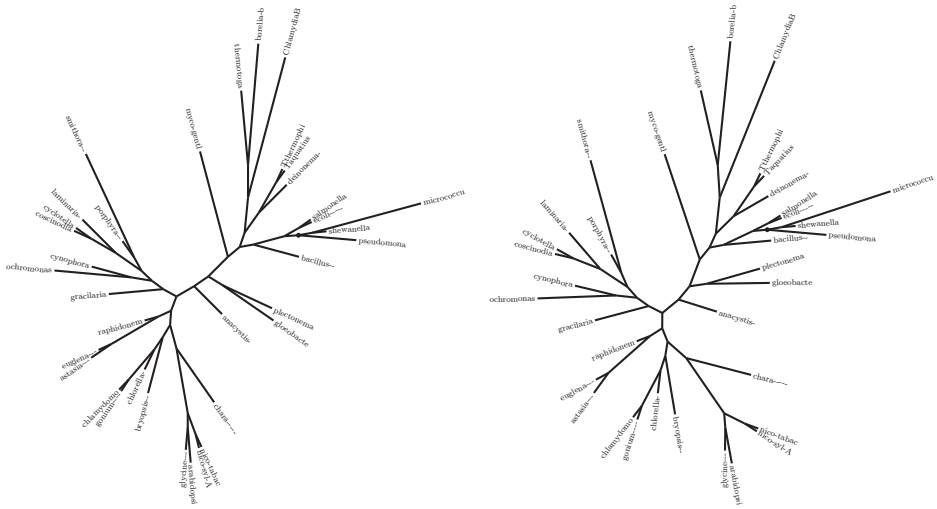
$$\mathrm{rgb}(f_e) = \begin{cases} (0,0,1) & \text{if } f_e \leq \frac{1}{2}, \\ (0,0,-\log_2(f_e)) & \text{if } \frac{1}{2} < f_e < 1, \\ (\log_2(f_e),0,0) & \text{if } 1 \leq f_e < 2, \\ (1,0,0) & \text{if } 2 \leq f_e \ . \end{cases} \tag{10}$$

so that blue and red signify edges that are too short and too long, respectively. 100% red means that the edge is at least twice as long as desired whereas 100% blue means that the edge should be at least twice as long. Weaker saturation reflects intermediate values. Black edges have the correct lengths.

5 Discussion

We have presented two linear-time algorithms for drawing phylogenetic trees. Example drawings are shown in Figs. 4 and 5. Both are easy to implement and scale very well. While the algorithm for radial drawings preserves edge lengths exactly, the algorithm for circle drawings is constrained by having leaves fixed on the perimeter of a circle. Since each inner vertex is positioned in the weighted barycenter of its neighbors, it would be interesting to devise a weighting scheme that, in a sense to be defined, is provably optimal with respect to the prespecified edge lengths. A related open question is the complexity status of deciding whether a circle drawing preserving the edge lengths exists.

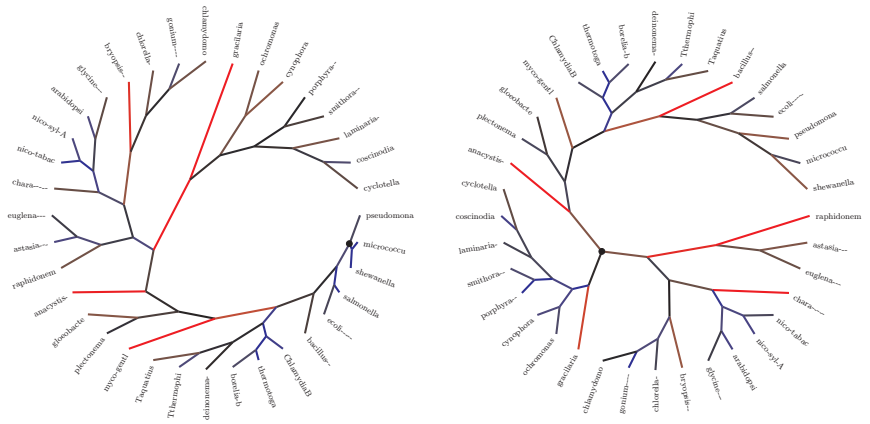
Acknowledgment. We wish to thank Lars Volkhardt for implementing our algorithms in Java using yFiles version 2.3 [19], and Falk Schreiber from the Institute of Plant Genetics and Crop Plant Research in Gatersleben for providing real-world data.



(a) Our basic algorithm

(b) Angle-spread extension

Fig. 4. Radial drawing examples



(a) Our basic algorithm

(b) Re-rooted at center

Fig. 5. Circular drawing examples

References

1. S. F. Carrizo. Phylogenetic trees: An information visualization perspective. In Y.-P. Phoebe Chen, editor, *Asia-Pacific Bioinformatics Conference (APBC 2004)*, volume 29 of *CRPIT*, pages 315–320. Australian Compter Science, 2004.
2. G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1999.
3. P. Eades. Drawing free trees. *Bulletin of the Institute of Combinatorics and its Applications*, 5:10–36, 1992.
4. P. Eades and N. C. Wormald. Fixed edge-length graph drawing is $\mathcal{NP}$-hard. *Discrete Applied Mathematics*, 28:111–134, 1990.
5. J. Felsenstein. Maximum likelihood and minimum-steps methods for estimating evolutionary trees from data on discrete characters. *Systematic Zoology*, 22:240–249, 1973.
6. W. M. Fitch. Torward defining the course of evolution: Minimum change for a specified tree topology. *Systematic Zoology*, 20:406–416, 1971.
7. M. Kaufmann and D. Wagner. *Drawing Graphs*, volume 2025 of *LNCS*. Springer, 2001.
8. R. J. Lipton, D. J. Rose, and R. E. Tarjan. Generalized nested dissection. *SIAM Journal on Numerical Analysis*, 16:346–358, 1979.
9. C. D. Michener and R. R. Sokal. A quantitative approach to a problem in classification. *Evolution*, 11:130–162, 1957.
10. R. D. M. Page. TreeView. `http://taxonomy.zoology.gla.ac.uk/rod/treeview.html`. University of Glasgow.
11. PHYLIP. Phylogeny inference package. `http://evolution.genetics.washington.edu/phylip.html`.
12. E. M. Reingold and J. S. Tilford. Tidies drawing of trees. *IEEE Transactions on Software Engineering*, 7(2):223–228, 1981.
13. N. Saitou and M. Nei. The neighbor-joining method: A new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4(4):406–425, 1987.
14. D. L. Swofford. PAUP*. Phylogenetic analysis using parsimony (and other methods). `http://paup.csit.fsu.edu/`. Florida State University.
15. D. L. Swofford, G. J. Olsen, P. J. Waddel, and D. M. Hillis. Phylogenetic inference. In D. Hillis, C. Moritz, and B. Mable, editors, *Molecular Systematics*, pages 407–514. Sinauer Associates, 2nd edition, 1996.
16. W. T. Tutte. Convex representations of graphs. In *Proc. London Mathematical Society, Third Series*, volume 10, pages 304–320, 1960.
17. W. T. Tutte. How to draw a graph. In *Proc. London Mathematical Society, Third Series*, volume 13, pages 743–768, 1963.
18. J. Q. Walker. A node-positioning algorithm for general trees. *Software Practice & Experience*, 20(7):685–705, 1990.
19. R. Wiese, M. Eiglsperger, and M. Kaufmann. yFiles: Visualization and automatic layout of graphs. In P. Mutzel, M. Jünger, and S. Leipert, editors, *Proc. Graph Drawing 2001*, volume 2265 of *LNCS*, pages 453–454. Springer, 2002.