Improved Layout of Phylogenetic Networks

Philippe Gambette, Daniel Huson

Improved Layout of Phylogenetic Networks

Philippe Gambette and Daniel H. Huson

Abstract—Split networks are increasingly being used in phylogenetic analysis. Usually, a simple equal-angle algorithm is used to draw such networks, producing layouts that leave much room for improvement. Addressing the problem of producing better layouts of split networks, this paper presents an algorithm for maximizing the area covered by the network, describes an extension of the equal-daylight algorithm to networks, looks into using a spring embedder, and discusses how to construct rooted split networks.

Index Terms—Phylogenetics, phylogenetic networks, graph drawing, algorithms.

✦

1 INTRODUCTION

PHYLOGENETIC networks are playing an increasingly important role in evolutionary studies, being employed either to represent the conflicting signals inherent to phylogenetic data or to explicitly model reticulate evolutionary events. One popular type of phylogenetic networks is split networks, introduced in [1] and subsequently studied in numerous papers. The aim of this paper is to introduce a number of new algorithms for computing better layouts of split networks and for drawing a rooted split network.

Algorithmically, split networks play an important role. On one hand, they provide a direct generalization of phylogenetic trees. On the other hand, we have recently shown that there are close relationships between split networks and reticulate networks [12], [13]. Based on this, we have developed algorithms that infer recombination and hybridization networks, and these are drawn using the construction and layout of a split network as an intermediate computational step.

In [5], we present the *equal-angle* algorithm for computing a split network that represents a given set of *circular* splits, which can optionally be followed by the *convex hull* algorithm to take care of any noncircular splits [2]. Although the equal-angle algorithm is guaranteed to produce a planar network for a set of circular splits, when applied to large data sets, the resulting networks can be unsatisfactory (see Fig. 8a) and may require a lot of interactive manipulation by the user to obtain a useful layout. In particular, parallelograms sometimes have very acute angles and small areas, which make them difficult to see.

We address this problem in a number of ways. First, we present a modification of the equal-angle algorithm that, by dropping the equal-angle constraint, can produce better layouts, at least for small networks. Second, we present a new *box-opening* algorithm that operates by locally modifying the angles of splits. Both approaches aim at maximizing the total area covered by the parallelograms in the network. Third, we describe an algorithm that extends the *equal-daylight* heuristic from phylogenetic trees to split networks. Fourth, we describe how to adapt a standard spring embedder approach to the task of embedding split networks.

The equal-angle algorithm for split networks produces an unrooted network. However, for the evolutionary interpretation of phylogenetic trees and networks, it is important that the position of the root in the tree or network is apparent. Additionally, in the application mentioned above in which the layout of a reticulate network is generated from a corresponding split network, a rooted split network is required. To address this, the final contribution of this paper is an algorithm for drawing a rooted split network, when given an outgroup.

Graph drawing is a well-studied problem [3], [4]; however, existing approaches do not appear to cover the goals pursued in this paper, which include that labeled nodes should (whenever possible) appear on the outside of the graph, edges representing the same split must be parallel and of the same length, and parallelograms in the graph should be as "open" as possible.

Although there exist a number of algorithms for drawing different types of phylogenetic networks, there seem to be only three programs that address the problem of drawing split networks, namely, SplitsTree [9], SplitsTree4 [10], and SpectroNet [8]. The latter program addresses the problem only indirectly and uses an implementation of the convex hull algorithm.

In Section 2, we briefly summarize some definitions and results related to splits and split networks. We describe two layout optimization techniques, the *optimized-angle* algorithm in Section 3 and the *box-opening* algorithm in Section 4. In Section 5, we discuss a spring embedder approach. We present the *equal-daylight* algorithm for split networks in Section 6. Finally, in Section 7, we discuss how to draw a *rooted* split network.

Our implementations of these algorithms are freely available as an integrated part of SplitsTree4 [10], a program that provides a wide range of different algorithms for computing phylogenetic trees and networks from

• P. Gambette is with Departement Informatique, Ecole Normale Superieure de Cachan, 61, avenue du President Wilson, 94235 Cachan Cedex, France. E-mail: philippe.gambette@dptinfo.ens-cachan.fr.
• D.H. Hurson is with the Center for Bioinformatics, Sand 14, Tubingen University, 72076 Tubingen, Germany. E-mail: huson@informatik.uni-tuebingen.de.

biological data sets. Our current implementations are aimed at data sets of up to 400 taxa, for example, which is the beyond the size of data set typically used with this type of software.

2   SPLITS AND SPLIT NETWORKS

We need the following basic definitions and facts concerning trees, splits, and split networks.

Let X denote a set of taxa. A *phylogenetic tree for X*, or *X-tree*, consists of a tree $T = (V, E)$ in which every node v is either a *leaf* of degree 1 or an *internal* node of degree ≥ 3, together with a leaf labeling $\nu : X \to V$ such that every leaf of T obtains a unique label [16]. Additionally, we may designate one of the taxa $o \in X$ to be an *outgroup* and then consider the tree to be "rooted" at the midpoint ρ of the pendant edge leading to $\nu(o)$ in the usual sense.

Suppose we are given a set of taxa X. A *split* (or, more precisely, *X-split*) is a bipartitioning of X into two nonempty sets A and B, denoted by $S = \frac{A}{B}(= \frac{B}{A})$.

For a given X-tree T, the deletion of any single edge e will produce a graph with exactly two connected components, and this defines a split $\sigma_T(e) = \frac{A}{B}$ given by the two sets of taxa labeling the two components [1]. The set of all splits obtainable in this way is called the *split encoding* $\Sigma(T)$ of T.

Two distinct X-splits $S = \frac{A}{B}$ and $S' = \frac{A'}{B'}$ are called *compatible* if one is a refinement of the other, that is, if one of the four following inclusions holds: $A \subset A'$, $A \subset B'$, $B \subset A'$, or $B \subset B'$. If S and S' are not compatible, then we call them *incompatible* and write $S \parallel S'$. A set of X-splits Σ is called *compatible* if all pairs of splits in Σ are compatible. The *incompatibility graph* $IG(\Sigma) = (V, E)$ has node set $V = \Sigma$ and edge set $E \subseteq \binom{V}{2}$, in which any two nodes S and S' are connected if and only if they are incompatible.

A basic result in mathematical phylogenetics [16] states that a set of X-splits Σ is compatible if and only if there exists a unique X-tree T with $\Sigma = \Sigma(T)$. In this case, we say that T *represents* Σ. Moreover, an arbitrary set of splits Σ, not necessarily compatible, can also be represented by a graph. Such a *split network* (also called a *split graph*) $SN(\Sigma)$ consists of a connected graph (V, E) together with a node labeling $\nu : X \to V$ and an edge coloring $\sigma : \Sigma \to E$, whose essential property is that deleting all edges colored by a given split $S = \frac{A}{B} \in \Sigma$ will produce precisely two connected components: one labeled by the taxa in A and the other labeled by the taxa in B (see [5] for details).

Let X be a set of taxa and assume that we are given a cyclic ordering $(x_1, \ldots, x_n)$. We say that an X-split S is *circular* (with respect to the given ordering) if there exist numbers p and q, with $1 < p \leq q \leq n$, such that $S = \frac{\{x_p, \ldots, x_q\}}{X \setminus \{x_p, \ldots, x_q\}}$.

The following result is simple but useful:

Lemma 1. *Let G be a directed graph and let v be some fixed node in the graph. Consider two different assignments of coordinates ω and ω' to the nodes of G. If $\omega(v) = \omega'(v)$ and if all edges have the same angles and lengths under both assignments of coordinates, then $\omega = \omega'$.*

In other words, if we fix the position of some node v, then an embedding of a graph is completely defined by specifying the angles and lengths of all edges. In particular, the layout of

a split network is uniquely defined (up to translation) by specifying an angle $\alpha(S_i)$ and length for each split S_i.

In many applications, the edges of a phylogenetic tree $T = (V, E)$ are "weighted," that is, a map $\omega : E \to \mathbb{R}^{\geq 0}$ is given that assigns a *length* or *weight* to each edge, usually representing some measure of evolutionary change along the edge. Thus, throughout this paper, we will assume that every set of splits Σ is "weighted" by a map $\omega : \Sigma \to \mathbb{R}^{\geq 0}$ that assigns a *length* or *weight* to every split $S \in \Sigma$. If the map ω is not explicitly given, then we will assume that $\omega(S) = 1$ for all splits $S \in \Sigma$.

3   THE OPTIMIZED-ANGLE ALGORITHM

The *equal-angle algorithm* described in [5] takes as input a set of X-splits $\Sigma = \{S_1, \ldots, S_k\}$ and a cyclic ordering $(x_1, x_2, \ldots, x_n)$ of the taxon set X and produces as output a split network representing all splits in Σ that are circular with respect to the given ordering.

In this network, each such split $S_i = \frac{\{x_p, \ldots, x_q\}}{X \setminus \{x_p, \ldots, x_q\}}$ (with $1 < p \leq q \leq n$) is represented by a set $E_i = \{e_1, \ldots, e_r\}$ of parallel edges of the same length. If we direct all edges in the graph away from the node labeled x_1, then the angle of every edge $e \in E_i$ in S_i is given by $\alpha_i = \frac{z_p + z_q}{2}$, where $z_p = \frac{p-1}{n} \times 360$ degrees is the angle associated with taxon x_p.

To visualize this, imagine the set of taxa uniformly arranged around the unit circle in the order $x_1, \ldots, x_n$, starting with x_1 at $z_1 = 0$ degree, in a positive direction, as shown in Fig. 2. Then, the angle of S equals the average angle assigned to the taxa $x_p, \ldots, x_q$.

In [5], we show that the resulting network is an *outer-labeled plane* graph, meaning that it is properly embedded in the plane, that is, no edges cross, and all labeled nodes appear around the outside of the graph.

In the equal-angle algorithm, taxa are uniformly spaced around the unit circle. However, for the algorithm to produce a graph that is an *outer-labeled plane*, it is not required that the spacing of the taxa be uniform. Indeed, any assignment of angles $z : X \to [0 \text{ degree}, 360 \text{ degrees})$ will do, as long as the cyclic ordering of the taxa is preserved. (As in [5], this follows from de Bruijn's Dualization Principle).

The main idea is to change the spacing between pairs of taxa around the unit circle in an attempt to optimize the layout of the network. The optimization goal that we propose is to maximize the total area covered by all parallelograms in the network.

We define a *box* as a parallelogram in the split network created by two incompatible splits. We can easily compute the area of a box from the weight of its two splits and one angle. For example, in Fig. 1, the two incompatible splits $S_1 = \frac{\{x_2, x_3, x_4, x_5\}}{\{x_6, x_7, x_8, x_1\}}$ and $S_2 = \frac{\{x_4, x_5, x_6, x_7, x_8\}}{\{x_1, x_2, x_3\}}$ give rise to box 2, with area $\omega(S_1)\omega(S_2)|\sin \alpha|$, where α is the indicated angle.

We propose to optimize the area using the following simple random search:

Algorithm 1 (optimized angle). Compute an initial layout using the equal-angle algorithm. Store the positions of the
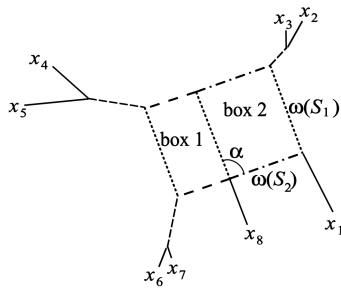
Fig. 1. Pairs of incompatible splits give rise to parallelograms in the network, and the optimization goal is to maximize the total area covered by the parallelograms.

taxa in an array $bestP$, determine the angles of all splits, and compute the total area covered and store it in $bestA$.

Repeat the following steps u times:

For each taxon, consider moving it halfway to either neighbor. In both cases, determine the angles of all splits and compute the total area covered.

If the total area covered is improved, save the new positions, and if it is greater than $bestA$, store the positions in $bestP$ and the area in $bestA$.

If the total area covered is not improved, save the new positions with probability p.

The complexity of this algorithm is $O(ukn)$. In our experience, $p = 0.8$ and $u = 500$ provide good results for small networks. However, the algorithm does not work well on larger networks. This is because the condition that the ordering of the taxa around the unit circle must be preserved is too strict.

The algorithm adjusts the angles of the taxa without taking the angles of the edges representing those splits into account, which are compatible with all other splits (see Fig. 3). The angles associated with such splits can subsequently be optimized using the equal-daylight algorithm, described in Section 6.

4 THE BOX-OPENING ALGORITHM

The algorithm described above suffers from the constraint that the layout of the taxa around the unit circle must be

Fig. 2. A split network constructed using the equal-angle algorithm. (a) Each split is represented by a cut-set of edges, consisting of either a single edge if the split is compatible with all other splits or a band of parallel edges otherwise. (b) The same set of splits is represented by chords of the unit circle, each separating the two split parts. The angles used in (a) are orthogonal to the ones used in (b).
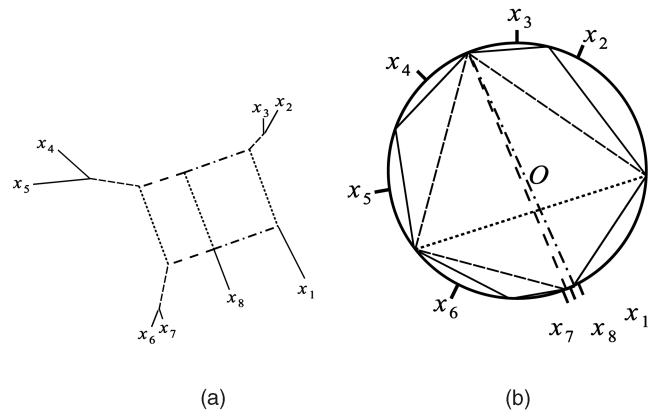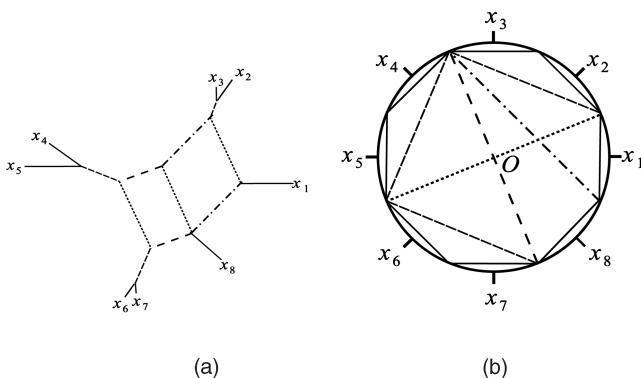

Fig. 3. (a) Network obtained by the optimized-angle algorithm from the example shown in Fig. 2. (b) The corresponding taxon layout, in which the circular ordering of taxa but not their angular spacing is preserved.

strictly preserved. In this section, we describe an algorithm that is not hampered by this constraint and produces better results in practice.

The algorithm considers each split $S \in \Sigma$ in turn. Assume that S is incompatible with splits $S_1, \ldots, S_t \in \Sigma$, then each pair S_i, S gives rise to a single box b_i in the network. The goal is to compute an angle $\alpha(S)$ such that the total area of all b_i is maximized. To ensure that the resulting network is planar, two types of collisions must be avoided.

The first type of collisions, which we will call *local* collisions, involves the nodes and edges of the boxes $b_1, \ldots, b_t$. The second type, *global* collisions, involves nodes or edges that lie in regions of the network that are not directly involved in the representation of the split S. In the following, we discuss how to avoid both types of collisions.

4.1 Preventing Local Collisions

A local collision happens when one of the boxes b_i is squashed flat. To avoid a local collision, note that there are two critical angles $\alpha_{max}(S)$ and $\alpha_{min}(S)$ that constrain the possible values of $\alpha(S)$:

$$\alpha_{max}(S) = \alpha(S) + \min_{\text{box } b_i}\{(\alpha_i - \alpha(S) - \pi) \bmod 2\pi\},$$
$$\alpha_{min}(S) = \alpha(S) - \min_{\text{box } b_i}\{(\alpha(S) - \alpha_i) \bmod 2\pi\},$$

where $\alpha(S)$ is the angle currently associated with split S, and α_i is the angle currently associated with split S_i ($i = 1, \ldots, k$) (see Fig. 4). Any choice of angle between α_{min} and α_{max} will give rise to a layout without local collisions.

4.2 Preventing Global Collisions

When modifying the angle $\alpha(S)$ of a split S, we can assume that one part of the split network, the "bottom part," stays fixed in the same place, whereas the "top part" of the graph is translated, as indicated in Fig. 4.

We have already discussed how to prevent local collisions. In a large or complicated graph, it may happen that modifying $\alpha(S)$ will cause a global collision between the bottom part and top part of the graph.

The collisions can occur on the left side or the right side of the split S, independent of whether we increase or decrease the angle of the split. Therefore, we perform similar calculations for all four points that represent the
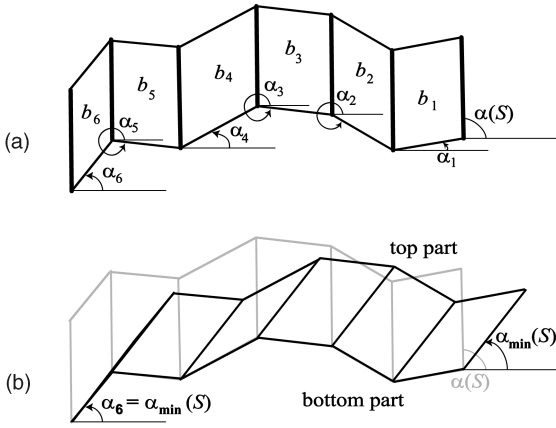
Fig. 4. (a) Edges corresponding to a split S are shown in bold; other edges represent splits $S_1, S_2, \ldots$ that are incompatible with S. Angle α_i is associated with split S_i. (b) Here, the angle $\alpha(S)$ has been modified to the critical value $\alpha_{min}(S) = \alpha_6$, for which the leftmost box collapses to a straight line.

"extreme nodes" of the split. Consider the rightmost edge representing split S in Fig. 4a. Assume that this edge is represented by the line (v, w) in Fig. 5. By definition, the set of edges representing split S splits the graph into two connected components, the bottom part containing all nodes $V_S(v)$ that can be reached from v and the top part containing all nodes $V_S(w)$ that can be reached from w, without using any edge representing S.

To determine p_1, we visit all nodes in $V_S(v)$ to find the one whose location p_1 minimizes the angle $(\overrightarrow{vp_1}, \overrightarrow{vw})$. Similarly, p_2 is obtained as the location of the node in $V_S(w)$ for which the angle $(\overrightarrow{vp_2}, \overrightarrow{vw})$ is maximal (see Fig. 5a).

As we assume that the bottom part of the network $V_S(v)$ will remain stationary, whereas the top part $V_S(w)$ is moved, we call p_1 the *defender* and p_2 the *striker*. If we decrease the angle $\alpha(S)$ by less than $\theta = (\overrightarrow{vp_1}, \overrightarrow{vp_2})$, then, usually, there will be no global collisions, and thus, we call this the *safe angle*, and for each of the four safe angles found, we check whether they place further restrictions on the interval permitted by the critical angles $\alpha_{min}(S)$ and $\alpha_{max}(S)$.

When the angle $\alpha(S)$ of the split S is changed, then the strikers and defenders may change: for example, in Fig. 5, the striker is p_2 in Fig. 5a but w', not p'_2, in Fig. 5b. Hence, after optimizing the angle of each split in the graph, a new round of optimization may be possible.
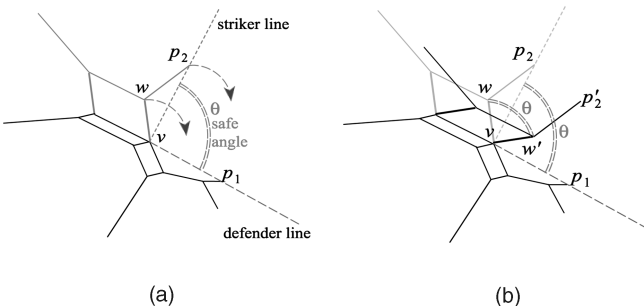

Fig. 5. Example of a critical situation where we try to avoid a collision on the right of the split. The situation (a) before and (b) after modification of the split angle.
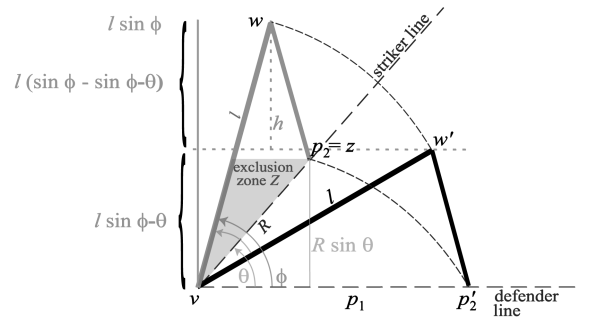

Fig. 6. Determining the coordinates of a node z such that decreasing the angle of the split by the safe angle θ will put z exactly on the defender line. Note that $h = R \sin \theta$ implies $R \sin \theta = l \sin \phi - l \sin(\phi - \theta) = l(\sin \phi - \sin(\phi - \theta))$.

Unfortunately, even if we respect the safe angle, collisions still may occur, because the transformation performed on the top part of the network is a translation, not a rotation. To address this, for all four "extremal" nodes, we will identify four "exclusion zones" such that if they are empty, then we are guaranteed to obtain a plane graph.

The following theorem gives the location of the exclusion zone associated with the rightmost node in the bottom part of the network for the split S:

Theorem 1. *Let p_1 be the defender node, p_2 be the striker node, ϕ be the angle between the defender line and the rightmost edge of the split, $\theta > 0$ be the safe angle, $l = \omega(S)$, $\theta' = -\min(\theta, \alpha(S) - \alpha_{min}(S))$, and z be the point such that $vz = \frac{l(\sin \phi - \sin(\phi - \theta))}{\sin \theta}$ and $(\overrightarrow{vz}, \overrightarrow{vw}) = \theta$. The exclusion zone Z is the triangle formed by the straight line L parallel to the defender line and containing z, the split edge (v, w), and the striker line (v, p_2) (see Fig. 6). If Z is empty, then no node will collide with the defender line on the right if we decrease the split angle $\alpha(S)$ by angle θ'.*

Proof. We first identify the location of striker nodes z such that if the split is rotated by the safe angle θ, then $z \in (v, p_1)$. As shown in Fig. 6, for such nodes z, with $l = vw$ and $R = zv$, we have $R \sin \theta = l(\sin \phi - \sin(\phi - \theta))$, so

$$R = \frac{l(\sin \phi - \sin(\phi - \theta))}{\sin \theta}.$$

This is the equation of the zone, depending on ϕ such that if the striker node is inside it and if we move the split angle by θ, it will collide with the defender line. Some examples of such zones are illustrated in Fig. 7.

Therefore, knowing the position of the striker node p_2, we can find the point z that lies on the striker line and satisfies the above equation. As the movement of the top part of the network is a translation, no node above z can collide with the defender line unless z does, and thus, the exclusion zone need not contain any node whose distance to the defender line (v, p_1) is strictly greater than the distance between (v, p_1) and z. By the definition of the striker node, there is no node below the striker line. Finally, as the angle θ' avoids local collisions, any node n below z and over the split edge, that is, such that $(\overrightarrow{vp_1}, \overrightarrow{vn}) > (\overrightarrow{vp_1}, \overrightarrow{vw})$ and $d(n, (v, p_1)) < d(z, (v, p_1))$, remains over the split edge after the change of the split angle and therefore will not collide with the striker line.
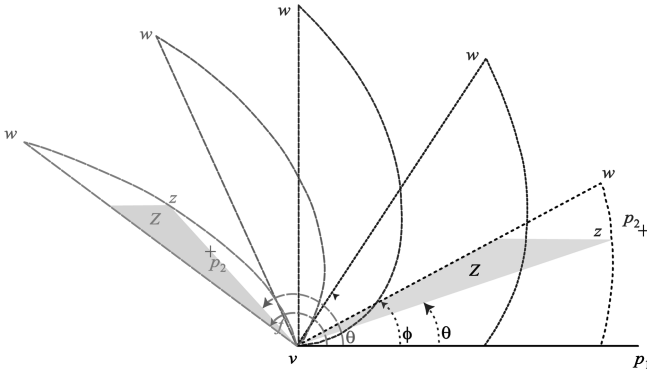
Fig. 7. For different values of the angle ϕ between the split edge and the defender line, the arcs indicate the boundaries of the regions that should not contain the striker to avoid a collision between the striker and the defender line. Then, by knowing the position of the striker, the exclusion zone Z can be drawn: it should not contain any node of the top part if we want to move the split angle by the safe angle θ without collisions.

Thus, the nodes that may cause collisions in our problem must indeed lie in the triangular exclusion zone Z. □

In practice, the exclusion zones usually do not contain nodes when the box-opening algorithm is applied to a network that was previously constructed by the equal-angle algorithm.

These considerations lead to the following algorithm:

Algorithm 2 (box opening). Perform the following loop u times:

For each split $S \in \Sigma$:

- Identify the extreme angles $\alpha_{min}(S)$ and $\alpha_{max}(S)$ to avoid local collisions.
- For each of the four extreme nodes of the split, identify the defender and the striker and compute the safe angle.
- If the exclusion zone is empty, use the four safe angles and the local constraints to determine the interval I of possible new angles for S. Else, set $I = \emptyset$.
- If I is nonempty, then compute a new angle that maximizes the total area covered by the boxes associated with S.

For a given split S_0, the angle $\alpha(S_0)$ that maximizes the total area of the set of boxes $\{b_1, \ldots, b_t\}$ associated with the set of splits $\{S_1, \ldots, S_t\}$ that are incompatible with S_0 can be directly determined from the formula for the total area associated with the boxes:

$$
\begin{aligned}
\text{Area} &= \sum_{S_i \not\parallel S_0} \omega_0 \omega_i |\sin(\alpha(S_0) - \alpha(S_i))| \\
&= \sum_{S_i \not\parallel S_0} \omega_0 \omega_i \sin(\alpha(S_0) - \beta_i) \\
&= \sin \alpha(S_0)(\omega_0 \sum_{S_i \not\parallel S_0} \omega_i \cos \beta_i) \\
&\quad - \cos \alpha(S_0) \quad (\omega_0 \sum_{S_i \not\parallel S_0} \omega_i \sin \beta_i),
\end{aligned}
$$

where ω_i is the weight associated with split S_i, and $\beta_i = \alpha(S_i)$ if $\sin(\alpha(S_0) - \alpha(S_i)) > 0$ and $\beta_i = \alpha(S_i) + \pi$ otherwise.

This can be written as

$$
\text{Area} = A \sin \alpha(S_0) + B \cos \alpha(S_0) = C \cos(\alpha(S_0) - D),
$$

with

$$
A = \omega_0 \sum_{S_i \not\parallel S_0} \omega_i \cos \beta_i, B = -\omega_0 \sum_{S_i \not\parallel S_0} \omega_i \sin \beta_i, C = \sqrt{A^2 + B^2},
$$

and $\tan D = \frac{B}{A}$.

Therefore, the optimal angle is obtained by maximizing $\cos(\alpha(S_0) - D)$ over the interval I.

In summary, the box-opening algorithm attempts to globally optimize the total area of the boxes in a split network by locally optimizing the area associated with individual splits. We have implemented this algorithm, and experiments show that the algorithm is fast and efficient in practice (see Fig. 8).

If n is the number of taxa and k is the number of splits, then the complexity of one iteration of the algorithm is

$$
O(number \ of \ splits \ \times \ numbers \ of \ nodes),
$$

that is, $O(nk + k^3)$, as the number of nodes is $O(n + k^2)$ for an outer-planar split network, as shown in [5].

5 A Spring Embedder Approach

One approach often in graph drawing is to use a *spring embedder* algorithm (such as [7]) that simulates a system of mass particles. The vertices of the graph correspond to mass points that repel each other, and the edges are interpreted as springs with attracting forces. The algorithm tries to minimize the energy of this physical system.

This type of algorithm is not directly applicable to split networks, as a spring embedder does not respect the constraint that edges representing the same split must be parallel and have the same length. To address this, the following strategy works well in practice:

Algorithm 3 (modified spring embedder). Obtain an embedding of a split network as follows:

- First, use a spring embedder to compute an "approximate" embedding of the given split network.
- For each split S represented in the network, let $\alpha(S)$ be the average angle of all the edges representing S in the approximate embedding.
- For each edge e, define a new angle $\alpha(e) = \alpha(S)$, where S is the split represented by e.
- Determine an exact embedding of the network using the new angles. This can be done using the algorithms described in [5], using the computed angles.

In our experience, this method is especially useful when the split network is significantly not outer-labeled planar, as depicted in Fig. 9. For complicated outer-labeled planar networks, the box-opening algorithm produces results that are often "superior" in the sense that the boxes are more open while converging just as fast in practice (see Fig. 8).
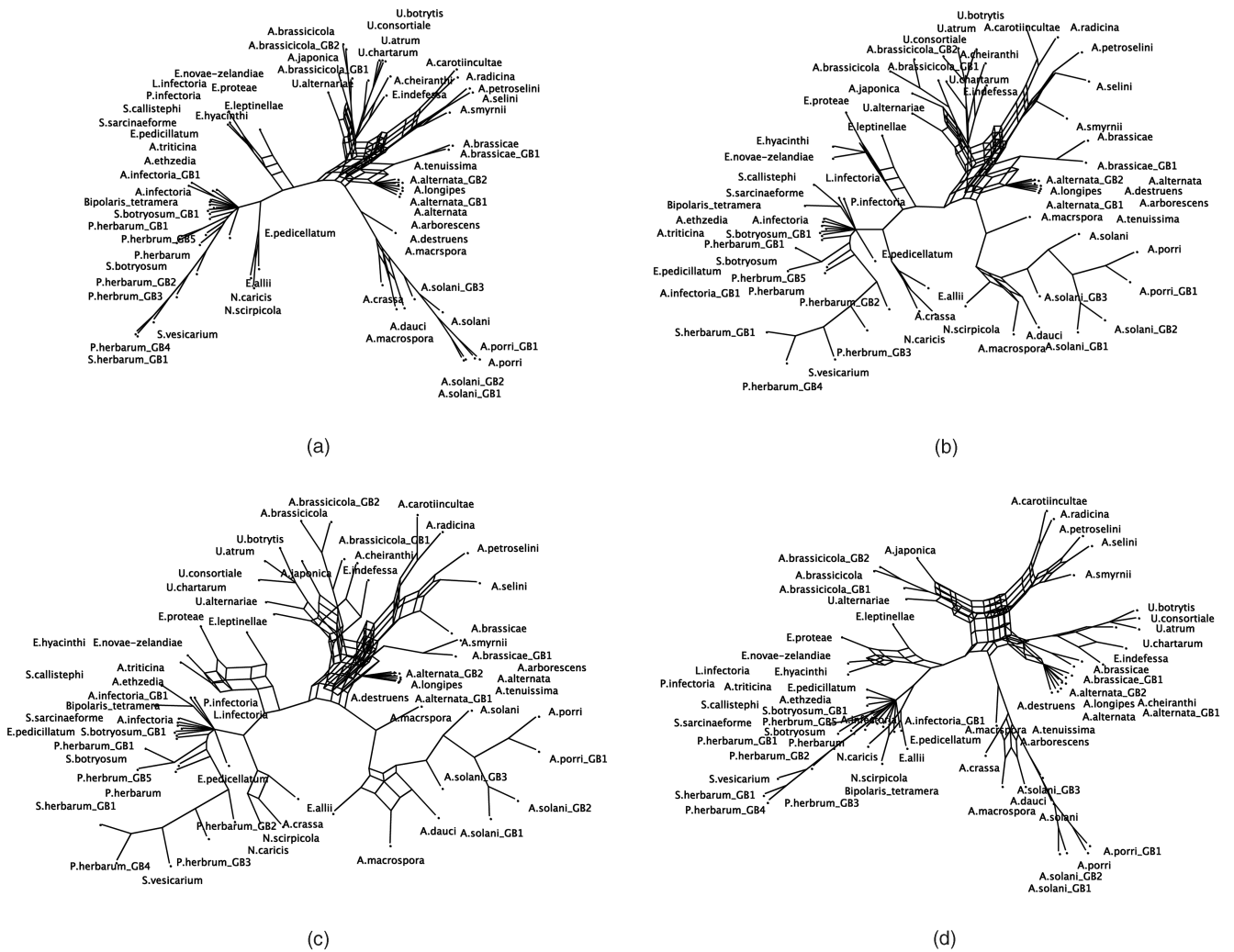
(a)

(b)

(c)

(d)

Fig. 8. Different layouts of the same split network representing five gene trees for fungal species [15], [14], [11] produced using (a) the unmodified equal angle algorithm as described in [5], (b) together with 10 iterations of the equal-daylight optimization, (c) and, additionally, 10 iterations of the box-opening heuristic (d) or with 700 iterations of the spring embedder algorithm. The four drawings took 3, $3+20$, $3+20+12$, and $3+210$ seconds to compute on a laptop, respectively.

6   THE EQUAL-DAYLIGHT ALGORITHM

Given a set of X-splits Σ and a circular ordering $(x_1, \ldots, x_n)$, the equal-angle algorithm is guaranteed to produce an outer-planar embedding of a corresponding split network. If the number of taxa is large, then, as with the equal-angle algorithm for trees (see [6, p. 578] for details), the angle at which two edges meet may be very small, and this can make the graph difficult to read. In the case of trees, Felsenstein [6, p. 582] describes an *equal-daylight algorithm* that modifies the angles of edges around a node v so that the amount of "daylight" between any two neighboring edges that reaches v is equal. In Figs. 8b, 8c, and 8d, we demonstrate the effect of using the equal-daylight algorithm.

Let $G = (V, E)$ be an embedded split network with node set V and edge set E. The equal-daylight algorithm is applied to each node of G in turn as follows: Let v be a node in G. Let $E_v = \{e_1, \ldots, e_s\}$ denote the set of edges incident to v, listed in the order that they are encountered when circling once around v in a mathematically positive orientation.

First, we determine the set $\mathcal{C} = \{C_1, C_2, \ldots, C_t\}$ of all connected components of the graph G^{-v} obtained by

removing v. If $t = 1$, then the equal-daylight algorithm cannot be applied to v. Otherwise, consider any component
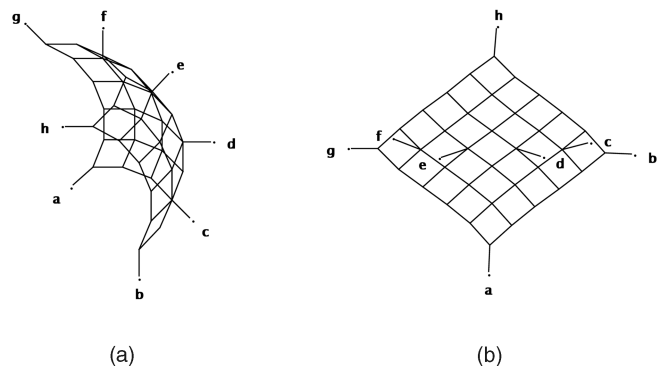

(a)

(b)

Fig. 9. Both networks display all splits contained in the two trees $((((a, b), c), d), (e, (f, (g, h))))$ and $((((h, b), c), d), (e, (f, (g, a))))$. (a) This network was constructed using the equal-angle algorithm. As it is not outer-planar, the layout produced by the algorithm is very poor. (b) This network was constructed by additionally running the described spring embedder technique.
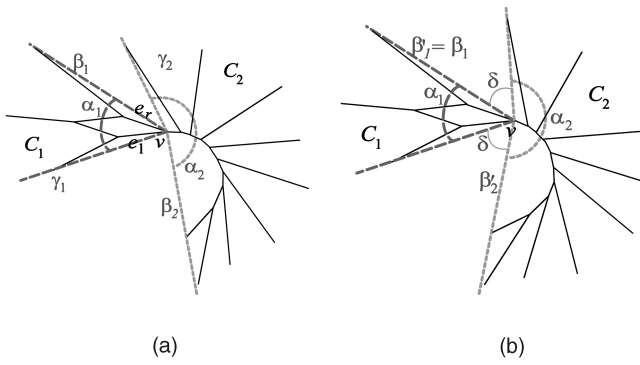
(a)

(b)

Fig. 10. (a) For this node v, the graph G^{-v} has precisely two components C_1 and C_2, and we indicate the two covered angles α_1 and α_2. (b) The equal-daylight algorithm will modify the layout of the two components so that the two angles between the two components both become equal to δ.
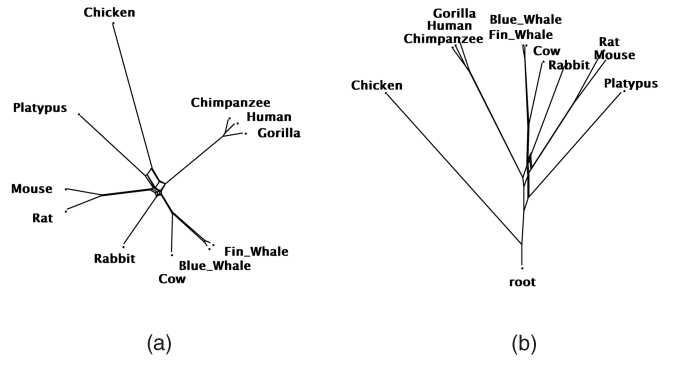

(a)

(b)

Fig. 11. (a) An unrooted split network computed for a set mammals and a chicken. (b) The corresponding rooted split network, rooted using the chicken as the outgroup.

C_i. We need to determine the minimum angle and maximum angle of a line segment from the node v to any node w in C_i. To do so, note that there exists a leftmost edge $e_l \in E_v$ and a rightmost edge $e_r \in E_v$ that connect v to some node in C_i. To determine the minimum angle β_i, initially set β_i equal to the angle of e_l. Then, leave v via the edge e_l and in a depth-first search and visit the whole component C_i, modifying the angle β_i whenever the angle of the current node, observed from v, is smaller than the current value β_i. To determine the maximum angle γ_i, proceed from e_r in a similar fashion. If $e_l = e_r$, then both angles can be computed in a single pass. We define the angle *covered by* C_i to be $\alpha_i = \gamma_i - \beta_i$ (see Fig. 10a).

We call $\Delta = 2\pi - \sum_{i=1}^{t} \alpha_i$ the *total* daylight angle for v and $\delta = \frac{\Delta}{k}$ the *equal* daylight angle. We now aim to rotate the components C_i around v so that the angle between any two adjacent components is δ.

Define $\beta_j^* = \sum_{i=1}^{j} \alpha_i + j\delta$. The goal is to rotate all components around v so that for each component C_i, the "leftmost" angle changes from β_i to β_i^*. To achieve this, for each edge e that connect any two nodes in $C_i \cup \{v\}$, add $\beta_i^* - \beta_i$ to the angle associated with e.

The equal-daylight algorithm modifies the angles associated with different edges in the graph. Again, a simple traversal of the network can then be used to assign coordinates to all nodes in the graph based on the angles (see Fig. 10b).

As described above, each node is treated separately, and a rearrangement around some node v might change the daylight angles around some other node w. Hence, in practice, the algorithm will be iterated a number of times.

7 CONSTRUCTING A ROOTED SPLIT NETWORK

In the visualization of an unrooted split network, edges point in many different directions, and taxa occur all around the boundary of the graph. In a rooted network, all edges point away from the root node (see Fig. 11). More precisely, if the root node is to be drawn at the bottom of the graph, then all edges will have angles that lie in a fixed range $[90 \text{ degrees} - \phi, 90 \text{ degrees} + \phi]$, with ϕ being a fixed parameter between 0 and 90 degrees.

Let $X = \{x_1, \ldots, x_n\}$ be a set of taxa and $\Sigma = \{S_1, \ldots, S_k\}$ a set of splits that are assumed to be circular with respect to the ordering $(x_1, \ldots, x_n)$. Assume that X contains precisely one outgroup taxon, x_1, for example, and that Σ contains the trivial split that separates x_1 from all other taxa.

Consider $X' = X \cup \{x_0\}$, where x_0 is a new taxon that will represent the root. In terms of the network, we want to attach a node labeled x_0 via a short edge to the interior of the leaf edge that leads to the node labeled by the outgroup x_1.

In terms of splits, this is done as follows: First, extend every X-split $S_i \in \Sigma$ to an X'-split by adding x_0 to the split part that contains x_1 and let Σ' be the set of all new splits obtained in this way. Additionally, add two new trivial splits: one for x_0 and the other for x_1.

To construct a rooted split network for Σ, we first use the algorithms described in [5] to compute the split network N' for Σ' topologically. Then, for each split $S_i = \frac{\{x_p, \ldots, x_q\}}{X - \{x_p, \ldots, x_q\}} \in \Sigma'$ (with $1 \leq p \leq q \leq n$), we define

$$\rho_i = 90 \text{ degrees} + \left(1 - \frac{p+q}{n}\right)\phi,$$

and thus assign an angle in the range $[90 \text{ degrees} - \phi, 90 \text{ degrees} + \phi]$ to S_i. A modification of the proofs given in [5] yields the following:

Lemma 2. *For any fixed value of ϕ in the open interval (0 degree, 90 degrees), using the angle ρ_i for each split $S_i \in \Sigma'$, we obtain a rooted plane split network representing Σ.*

REFERENCES

[1] H.-J. Bandelt and A.W.M. Dress, "A Canonical Decomposition Theory for Metrics on a Finite Set," *Advances in Mathematics,* vol. 92, pp. 47-105, 1992.

[2] H.-J. Bandelt, P. Forster, B.C. Sykes, and M.B. Richards, "Mitochondrial Portraits of Human Population Using Median Networks," *Genetics,* vol. 141, pp. 743-753, 1995.

[3] G. Di Battista, P. Eades, R. Tamassia, and I. Tollis, "Algorithms for Drawing Graphs: An Annotated Bibliography," *Computational Geometry: Theory and Applications,* vol. 4, no. 5, pp. 235-282, 1994.

[4] J. Diaz, J. Petit, and M. Serna, "A Survey of Graph Layout Problems," *ACM Computing Surveys,* vol. 34, no. 3, pp. 313-356, 2002.

[5] A.W.M. Dress and D.H. Huson, "Constructing Splits Graphs," *IEEE/ACM Trans. Computational Biology and Bioinformatics,* vol. 1, no. 3, pp. 109-115, July-Sept. 2004.

[6] J. Felsenstein, *Inferring Phylogenies.* Sinauer Assoc., 2004.

[7]   T. Fruchterman and E. Reingold, "Graph Drawing by Force-Directed Placement," *Software—Practice and Experience,* vol. 21, no. 11, pp. 1129-1164, 1991.

[8]   K.T. Huber, M. Langton, D. Penny, V. Moulton, and M. Hendy, "Spectronet: A Package for Computing Spectra and Median Networks," *Applied Bioinformatics,* vol. 1, pp. 159-161, 2002.

[9]   D.H. Huson, "SplitsTree: A Program for Analyzing and Visualizing Evolutionary Data," *Bioinformatics,* vol. 14, no. 10, pp. 68-73, 1998.

[10] D.H. Huson and D. Bryant, "Application of Phylogenetic Networks in Evolutionary Studies," *Molecular Biology and Evolution,* vol. 23, no. 2, pp. 254-267, www.splitstree.org, 2006.

[11] D.H. Huson, T. Dezulian, T. Kloepper, and M.A. Steel, "Phylogenetic Super-Networks from Partial Trees," *IEEE/ACM Trans. Computational Biology and Bioinformatics,* vol. 1, no. 4, pp. 151-158, Oct.-Dec. 2004.

[12] D.H. Huson, T. Kloepper, P.J. Lockhart, and M.A. Steel, "Reconstruction of Reticulate Networks from Gene Trees," *Proc. Ninth Int'l Conf. Research in Computational Molecular Biology (RECOMB '05),* 2005.

[13] D.H. Huson and T.H. Kloepper, "Computing Recombination Networks from Binary Sequences," *Proc. European Conf. Computational Biology (ECCB '05),* 2005.

[14] B.M. Pryor and D.M. Bigelow, "Molecular Characterization of *Embellisia* and *Nimbya* Species and Their Relationship to *Alternaria*, *Ulocladium* and *Stemphylium*," *Mycologia,* vol. 95, no. 6, pp. 1141-1154, 2003.

[15] B.M. Pryor and R.L. Gilbertson, "Phylogenetic Relationships among *Alternaria* and Related Fungi Based upon Analysis of Nuclear Internal Transcribed Sequences and Mitochondrial Small Subunit Ribosomal DNA Sequences," *Mycological Research,* vol. 104, no. 11, pp. 1312-1321, 2000.

[16] C. Semple and M.A. Steel, *Phylogenetics.* Oxford Univ. Press, 2003.

Philippe Gambette received the degree in computer science from the Ecole Normale Supérieure de Cachan. He is currently working toward the master's degree, doing research training in graph theory with Michel Habib, at LIAFA, Paris. He worked on phylogeny in research trainings with Olivier Gascuel at LIRMM, Montpellier, and with Daniel Huson at Tübingen University.

Daniel H. Huson received the degree in mathematics and the PhD degree in 1990 from Bielefeld University. From 1990 to 1999, he held a variety of different research positions at Bielefeld University and was supported during this time by a two-year DFG research scholarship. He then spent two years, from 1997 to 1999, as a postdoctorate working with Tandy Warnow at Princeton University. He then joined Celera Genomics as a senior research scientist working in Gene Myers' group. Since 2002, he has been a professor of algorithms in bioinformatics at Tübingen University, Germany.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.