Tanglegrams for rooted phylogenetic trees and networks

Celine Scornavacca*, Franziska Zickmann and Daniel H. Huson*
Center for Bioinformatics (ZBIT), Tübingen University, Sand 14, 72076 Tübingen, Germany

ABSTRACT
Motivation: In systematic biology, one is often faced with the task of comparing different phylogenetic trees, in particular in multi-gene analysis or co-speciation studies. One approach is to use a tanglegram in which two rooted phylogenetic trees are drawn opposite each other, using auxiliary lines to connect matching taxa. There is an increasing interest in using rooted phylogenetic networks to represent evolutionary history, so as to explicitly represent reticulate events, such as horizontal gene transfer, hybridization or reassortment. Thus, the question arises how to define and compute a tanglegram for such networks.
Results: In this article, we present the first formal definition of a tanglegram for rooted phylogenetic networks and present a heuristic approach for computing one, called the AN-tanglegram method. We compare the performance of our method with existing tree tanglegram algorithms and also show a typical application to real biological datasets. For maximum usability, the algorithm does not require that the trees or networks are bifurcating or bicombining, or that they are on identical taxon sets.
Availability: The algorithm is implemented in our program Dendroscope 3, which is freely available from www.dendroscope.org.

1 INTRODUCTION
In systematic biology, one is often faced with the task of comparing different phylogenetic trees, in particular in multi-gene analysis or co-speciation studies (Burt and Trivers, 2008; Charleston, 1998; Charleston and Perkins, 2003; Lee and Stock, 2010; Machado et al., 2005; Merkel et al., 2010). One way to visualize similarities and differences is to draw two phylogenetic trees as rooted trees side by side and to draw lines (which we will call connectors) between taxa that correspond to each other in the two trees, see Figure 1. Such a depiction is called a tanglegram and different variations of the problem of computing an optimal tanglegram have been studied in the literature.

For example, a number of articles (Bansal et al., 2009; Böcker et al., 2009; Buchin et al., 2009; Fernau et al., 2005; Nöllenburg et al., 2009; Venkatachalam et al., 2010) consider the One-Tree Crossing Minimization (OTCM) and the Two-Tree Crossing Minimization (TTCM) problems that both aim at minimizing the number of crossings between connectors. In the former problem, the layout of one of the trees is fixed and that of the other is mutable whereas in the latter formulation the layout of both trees are allowed to be changed. For binary trees, OTCM is solvable in $O(n \log n)$ time (Venkatachalam et al., 2010), while TTCM is NP-complete (Fernau et al., 2010). In Dwyer and Schreiber (2004), the authors describe a ‘seesaw’ heuristic for the TTCM problem for binary (or bifurcating) trees, which operates by repeatedly solving the OTCM problem, each time switching the roles of the two trees. A branch-and-bound approach for binary trees that works in $O(n^3)$ time and gives a 2-approximation for complete binary trees is presented in Buchin et al. (2009). A generalization of the algorithm to unbalanced binary trees is described in Nöllenburg et al. (2009), though in this case the approximation factor does not hold. In addition, this article gives an ILP formulation and an exact branch-and-bound algorithm for binary trees, where the latter has a worst-case running time of $O(n^3 + n^{2.5})$. Other approaches use fixed-parameter tractability (FPT) parameterized by the number $k$ of connector crossings (Fernau et al., 2005). A generalization to non-binary trees is discussed in Venkatachalam et al. (2010). The only algorithm that is able to compute tanglegrams for binary trees with many-to-many connections is described in Bansal et al. (2009). Their algorithm requires $O(k \log k / \log \log k)$ time (where $k$ is the number of connectors) in the case that one tree is fixed. Additionally, they present some alternating and local search strategies for the TTCM problem.

While evolutionary histories are usually described by rooted phylogenetic trees, in some cases rooted phylogenetic networks may provide a more accurate evolutionary scenario, especially when mechanisms such as horizontal gene transfer, hybridization, recombination, reassortment or incomplete lineage sorting have played a role in shaping the history. There is currently much research on the development of computational methods for computing rooted phylogenetic networks, for an overview see Huson and Scornavacca (2011a; Huson et al. (2011).

The goal of this article is to introduce the concept of a tanglegram for rooted phylogenetic networks and to provide a useful heuristic for computing such tanglegrams. Unlike trees, rooted phylogenetic networks are not necessarily planar and so the definition of an optimal tanglegram is not immediately obvious for them. Our heuristic does not require that the networks are bifurcating or
2 BASICS

Throughout this article, we follow the terminology and notation defined in Huson et al. (2011) and assume that the reader is familiar with graphs and related terminology. In this section we introduce some basic concepts and results.

2.1 Phylogenetic networks and networks

Let \( X \) be a set of taxa. An unrooted phylogenetic network on \( X \) is an unrooted, connected graph whose leaves are bijectively labeled by the taxa in \( X \). A rooted network is defined as a directed graph that is free of directed cycles and contains precisely one node with indegree zero, called the root. A rooted phylogenetic network \( N \) on \( X \) is a rooted network whose set of leaves is bijectively labeled by the taxa in \( X \). A node \( v \) in \( N \) is said to be a tree node if its indegree is less than two and a reticulate node otherwise. An edge of \( N \) is called a tree edge, unless it leads to a reticulate node, in which case it is called a reticulate edge. A (rooted) phylogenetic network is a (rooted) phylogenetic network for which it is not possible to delete any edge without producing a graph that is not connected.

Let \( N \) be a rooted phylogenetic network on \( X \). Any tree edge \( e \in N \) defines a cluster \( C(e) \) which is given by the set of all taxa that label leaves that lie below \( e \) in \( N \). We use \( \mathcal{C}(N) \) to denote the set of all clusters obtainable from \( N \) in this way.

Let \( X \) be a set of taxa. A split \( A \mid B \) on \( X \) is a partition of \( X \) into two non-empty sets. Any edge \( e \) in an unrooted phylogenetic tree \( T \) on \( X \) defines a split \( S(e) = A \mid B \), where \( A \) and \( B \) are all taxa that label nodes that lie on one side or the other side of the edge, respectively. A split \( A \mid B \) is said to separate two taxa \( x \) and \( y \) if they are contained in different parts of the split, that is, if \(|\{x, y\} \cap A| = |\{x, y\} \cap B| = 1\) holds. The restriction of a split \( S \) to a set of taxa \( X' \), denoted by \( S|_{X'} \), is defined as the split \( S' = A' \mid B' \), where \( A' = A \cap X' \) and \( B' = B \cap X' \).

Let \( S \) be a set of splits on \( X \). A weighting of \( S \) is given by a map \( \omega \) that assigns a non-negative weight \( \omega(S) \) to each split \( S \in \Sigma \). Let \( S \) be a set of weighted splits on \( X \). We define a distance matrix of split distances \( D(S) = (d_{S_{ij}}) \) on \( X \) by setting \( d_{S_{ij}} = \sum_{e \in S_{ij}} \omega(e) \), where \( X(e) \) denotes the set of all splits in \( S \) that separate the taxa \( x \) and \( y \). If \( S \) is unweighted, then we use this definition with \( \omega(S) = 1 \) for all \( S \).

Definition 2.1 (Compatibility). Two splits \( A_1 \mid B_1 \) and \( A_2 \mid B_2 \) are compatible if at least one of the sets \( A_1 \cap A_2 \), \( A_1 \cap B_2 \), \( B_1 \cap A_2 \) or \( B_1 \cap B_2 \) is empty. Otherwise they are called incompatible. A collection of splits \( S \) is compatible if and only if all splits are pairwise compatible.

A set of compatible splits can always be represented by a tree (Buneman, 1971). More generally, any set of splits \( S \) can always be represented by a split network \( N \) (Dress and Huson, 2004). This is an unrooted phylogenetic network with the property that every split \( S = A \mid B \) in \( S \) is represented in \( N \) by a set of parallel edges, such that deleting all edges in the set will result in exactly two connected components, one labeled by the taxa in \( A \) and the other labeled by \( B \).

A class of splits of particular interest are the circular splits.

Definition 2.2 (Circular splits). A set of splits \( S \) on \( X \) is called circular if there exists a linear ordering \( \pi = (x_1, x_2, \ldots, x_n) \) of the elements of \( X \) for \( S \) such that each split \( S \in \Sigma \) is interval-realizable, that is, has the form

\[
S = \{(y_p, y_{p+1}, \ldots, y_q) \mid (x_1, x_2, \ldots, x_n)\},
\]

for appropriately chosen \( 1 \leq p \leq q \leq n \).

Such an ordering \( \pi = (x_1, \ldots, x_n) \) is called a circular ordering for \( S \), as it holds that \((x_1, \ldots, x_n)\) is a circular ordering for \( S \), if and only if the inverse ordering \((x_n, x_{n-1}, \ldots, x_1)\) and \((x_2, x_3, \ldots, x_{n-1}, x_1)\) both are.

Circular splits are of particular interest because any set of circular splits can always be represented by a split network that can be drawn in the plane such that no two edges intersect and all labeled nodes lie on the outside of the network, see Figure 2.

2.3 The Neighbor-Net algorithm

Given a distance matrix \( D \) on \( X \), the Neighbor-Net algorithm (Bryant and Moulton, 2004) computes a circular ordering \( \pi \) of \( X \) from \( D \) and then a set of weighted splits \( S \) that are interval-realizable with respect to \( \pi \). A distance matrix \( D \) on \( X \) is said to be circular if and only if there exists a set of circular weighted splits \( S \) on \( X \) such that \( D(S) = D \).

The result of the Neighbor-Net algorithm is given by the consistency of the Neighbor-Net method:

Theorem 2.3 (Neighbor-Net consistency (Bryant et al., 2007)).

Given a circular distance matrix \( D \) on \( X \), the Neighbor-Net algorithm produces a circular ordering \( \pi \) and a set of weighted splits \( S \) that are interval-realizable with respect to \( \pi \) such that \( D = D(S) \).
3.1 Definition of a tanglegram for networks

Let $N_1$ and $N_2$ be two rooted phylogenetic networks on taxa sets $X_1$ and $X_2$, respectively. We will use $M \subseteq X_1 \times X_2$ to denote a set of connectors between $X_1$ and $X_2$. If the two networks are on the same taxon set, then $M$ is the set of identity connectors that connects each taxon to itself, which we will denote by $M_T$ below for emphasis.

In the case of a host-parasite comparison, $M$ will pair hosts and parasites.

Unlike trees, rooted phylogenetic networks are not necessarily planar and so the definition of a tanglegram is not immediately obvious for them:

**Definition 3.2 (Tanglegram for networks).** Let $N_1$ and $N_2$ be two rooted phylogenetic networks on $X_1$ and $X_2$, respectively, and let $M$ be a set of connectors between $X_1$ and $X_2$. A tanglegram $Z$ for $N_1$, $N_2$, and $M$ is specified by a system $(N_1, X_1, \tau_1, \pi_1, N_2, X_2, \tau_2, \pi_2, M)$ where $\tau_i$ is a topological embedding of $F(N_i)$ and $\pi_i$ is a non-interleaving total order of $X_i$ with respect to $\tau_i$, for $i = 1, 2$.

Let $N$ be a rooted phylogenetic network on $X$. Consider a concrete drawing $\theta$ of $N$ in the plane. We call $\theta$ a rooted outer-labeled (tree-planar embedding) if all taxa labels are placed on a line, the root node occurs on the outside of the embedding, and no two (tree) edges cross. If $N$ possesses a non-interleaving order $\pi$ of $X$ with respect to a topological embedding $\tau$ for $F(N)$, then there exists a rooted outer-labeled tree-planar embedding for $N$ in which the taxa appear along a line in the order specified by $\pi$. As mentioned above, the non-interleaving property of $\pi$ ensures that we can lineup appropriate embeddings of all the trees in $F(N)$ in the order induced by $\pi$.

A drawing of a tanglegram $Z := (N_1, X_1, \tau_1, \pi_1, N_2, X_2, \tau_2, \pi_2, M)$ consists of a rooted outer-labeled tree-planar embedding of both $N_1$ and $N_2$, together with a set of lines representing the connectors between $X_1$ and $X_2$. Such a drawing of $Z$ can be obtained in the following steps. First, draw all trees in $F(N_1)$ and $F(N_2)$ in such a way that the two orderings $\tau_1$ and $\tau_2$ are respected and all the leaves of $N_1$ and $N_2$ are lined up in the order specified by $\pi_1$ and $\pi_2$. Second, add all reticulate edges to the diagram. These two steps can always be done in such a way that no two tree edges cross and that the roots of $N_1$ and $N_2$ occur on the outside of the drawing, due to the fact that both $\tau_1$ and $\tau_2$ are non-interleaving. Finally, draw lines between the leaves of the two networks so as to connect taxa as specified in $M$.

Note that, if $N_1$ and $N_2$ are two trees, $F(N_1)$ and $F(N_2)$ coincide with $N_1$ and $N_2$, respectively. This means that, in this case, a tanglegram between $N_1$ and $N_2$ defines univocally the rooted outer-labeled tree-planar embeddings of $N_1$ and $N_2$. Given an embedding, drawing a tree is straightforward (see Huson et al., 2011). Therefore, our definition of a tanglegram for two rooted phylogenetic networks generalizes the definition of a tanglegram for two rooted phylogenetic trees, and so, in particular, the problem of computing an optimal tanglegram for networks is NP-complete (Fernau et al., 2010).

Let $N$ be a rooted phylogenetic network on $X$, let $\tau$ be a topological embedding of $F(N)$ and let $\pi$ be a non-interleaving total order on $X$. We define the reticulation crossing number as the minimum number of crossings involving reticulation edges in any drawing of $N$ respecting $\tau$ and $\pi$. An optimal tanglegram can now be defined as follows:

**Definition 3.3 (Optimal tanglegram).** Let $N_1$ and $N_2$ be two rooted phylogenetic networks on $X_1$ and $X_2$, respectively, and let $M$ be a set of connectors between $X_1$ and $X_2$. A tanglegram $Z = (N_1, X_1, \tau_1, \pi_1, N_2, X_2, \tau_2, \pi_2, M)$ is called optimal if the crossings between connectors in $M$ is minimized by $\tau_1$ and $\tau_2$ and, among the
Tanglegrams for rooted phylogenetic networks

tanglegrams minimizing this value, it can be drawn so as to minimize the sum of reticulation crossing numbers for \(N_1\) and \(N_2\).

Let \(N_1\) and \(N_2\) be two rooted phylogenetic networks on \(X_1\) and \(X_2\), respectively, and let \(M\) be a set of connectors between \(X_1\) and \(X_2\). For two linear orderings \(\pi_1\) and \(\pi_2\) of \(X_1\) and \(X_2\), the number of crossings \(C(T(\pi_1, \pi_2, M))\) among connectors in \(M\) can be calculated as \(|\{(a, b) \in M \times M | a = (p, q), b = (x, y) \text{ with } p <_M x <_M q <_M y \} \cap \{(p, q) \times (x, y) \text{ with } p <_M x \land \neg (q <_M y)\}|\). A heuristic that computes the reticulate crossing number for a drawing of a rooted phylogenetic network is described in (Huson, 2009).

3.2 Neighbor-net heuristic for tanglegrams

Let \(N_1\) and \(N_2\) be two rooted phylogenetic networks on \(X_1\) and \(X_2\), respectively, and let \(M\) be a set of connectors between \(X_1\) and \(X_2\). For the purposes of this paper, we will assume that \(M\) pairs the leaves of the two trees in a one-to-one manner. (Note, however, that the example shown in Figure 1 does not have this property.) If \(M\) pairs two distinct taxa \(a\) and \(b\), as for example in a host/parasite study, then we identify the labels \(a\) and \(b\) with each other while computing the tanglegram, but then distinguish between the two labels when drawing the tanglegram.

In the following, we present a heuristic for obtaining an optimal tanglegram for \(N_1\) and \(N_2\). We call this the NN-tanglegram approach.

In our approach, we first compute a distance matrix \(H\) on the total set of taxa \(X = X_1 \cup X_2\) that reflects the topology of the two networks \(N_1\) and \(N_2\), then apply the Neighbor-Net algorithm to \(H\) to obtain an ordering \(\pi\) of \(X\), and finally construct a tanglegram for \(N_1\) and \(N_2\) based on \(\pi\). In the following, we will assume that both networks contain a single leaf connected to the root of the network that is labeled by a special taxon \(a\), which we will refer to as a formal outgroup.

To compute a distance matrix on \(X\), we need to construct a set of splits for each of the two networks. We describe the process for the network \(N_1\). For each tree edge \(e\) in \(N_1\) let \(C_1(e)\) be the set of all taxa that label a leaf that lies below \(e\) in \(N_1\). We define the split associated with \(e = (C_1(e), X_1 \setminus C_1(e))\). Let \(\Sigma(N_1)\) denote the set of all splits obtained in this way. We compute \(\Sigma(N_2)\) similarly and we then define \(\Sigma(N_1) \cup \Sigma(N_2) = \Sigma(N_1) \cup \Sigma(N_2)\). (The restriction to \(X_1 \cup X_2\) ensures the applicability of Theorem 3.5 if \(X_1 \neq X_2\).)

We obtain the distance matrix \(H\) on \(X\) by setting the distance between two taxa \(x\) and \(y\) equal to the number of splits that separate the two taxa, where any split that occurs both for \(N_1\) and \(N_2\) is counted twice. In other words, we set \(H = D(S(N_1|X_1 \cup X_2)) + D(S(N_2|X_1 \cup X_2))\).

In the simulation study in Section 5.1, we will also present the results when using \(H = D_{sab}(N_1) + D_{sab}(N_2)\) as distance matrix, where \(D_{sab}(N) = (d_{ij})\) such that \(d_{ij}\) is the length of the shortest path between \(x_i\) and \(y_j\) in \(N_2\). As we will see, this variant actually performs better on networks than using splits-based distances.

We apply the Neighbor-Net algorithm to the distance matrix \(H\) (or \(H'\) in the case that the shortest path distance matrix is used) so as to obtain a circular ordering \(\xi = (x_1, \ldots, x_n)\) of \(X\). The ordering \(\xi\) is computed in this way because this ensures that the NN-tanglegram method returns the optimal solution under optimal conditions (see Theorem 3.5). Let \(i\) denote the position of the formal outgroup taxon \(a\) in \(\xi\). We obtain a linear ordering \(\pi\) of \(X\) by breaking the ordering \(\xi\) at position \(i\), that is, by setting \(\pi = (x_1, x_2, \ldots, x_i, x_{i+1}, \ldots, x_n)\). In the following, we will assume that \(\pi\) is the ordering of \(X\), we now have to compute two embeddings \(\tau_1\) and \(\tau_2\) for the forests \(F(N_1)\) and \(F(N_2)\) such that \(\pi\) is non-interleaving with respect to \(\tau_1\) and \(\tau_2\). Given a rooted phylogenetic network \(N\) and a node \(u\) of \(N\), we use \(X_u(a)\) to denote the set of taxa that label the leaves below \(u\).

To compute \(\tau_1\), we first delete all reticulate edges in \(N_1\) to produce the forest \(F(N_1)\). Then, for each \(\tau^* \in F(N_1)\), we determine a topological embedding \(\tau^*\) that minimizes the number of crossings among connectors \(C(r, \pi, M)\), where \(r\) is the ordering of \(X^\ast\) induced by the embedding \(\tau^*\). This optimization is easily solved in a bottom-up traversal of each tree in \(F(N_1)\).

Note that the place that is assigned to a node \(v\) such that \(X_u(v) = \emptyset\) or \(X_u(v) \cap X_1 = \emptyset\) in the topological embedding of its parent is not relevant for the computation of the number of crossings among connectors \(C(r, \pi, M)\) and can be chosen arbitrarily. The set of topological embeddings for all trees \(\tau^* \in F(N_1)\) constitutes \(\tau_1\). To obtain \(\tau_2\), we add the taxa of \(X_1\) to \(\tau_1\) one by one, in such a way that \(\pi_1\) remains non-interleaving w.r.t. \(\tau_1\) and the value of \(C(\tau_1, \pi, M)\) is minimized. The ordering \(\tau_2\) is computed in exactly the same way but using the network \(N_2\) instead of \(N_1\).

Let \(C(N)\) be the set of clusters associated with \(N\). We say that \(C(N)\) is interval-realizable with respect to \(\pi = (x_1, x_2, \ldots, x_n)\) if each cluster \(C\) in \(C(N)\) has the form \([x_p, x_{p+1}, \ldots, x_q]\), for appropriately chosen \(1 \leq p < q \leq n\). We have the following result:

**Lemma 3.4** (Interval realizability). Let \(N\) be a phylogenetic network on \(X\). If \(N\) has a rooted outer-labeled planar embedding \(\theta\) and \(\tau_0\) is the linear order on \(X\) that is defined by \(\theta\), then \(C(N)\) is interval-realizable with respect to \(\tau_0\).

This lemma is used to prove the following theorem:

**Theorem 3.5** (Zero crossings solution). Let \(N_1\) and \(N_2\) be two rooted phylogenetic networks on \(X_1\) and \(X_2\), respectively, and let \(M\) be a set of connectors between \(X_1\) and \(X_2\). If a planar drawing exists for the tanglegram of \(N_1, N_2\) and \(M\), then the NN-tanglegram heuristic will find a solution with zero crossings among connectors.

The proof of both results can be found in Appendix A. Note that Theorem 3.5 ensures that, if an optimal tanglegram with cost zero exists for two trees, then our algorithm will find it, because a tree does not contain any reticulate edges.

This is not true for networks. Indeed, in this case the Neighbor-Net algorithm may have more than one optimal solution. Theorem 3.5 ensures that any linear ordering \(\pi\) computed as described in Section 3.2 can be realized with zero crossings among connectors, but it does not guarantee that the resulting drawing will have zero crossings involving reticulate edges. For example, for the two networks in Figure 4, both orders \((a, b, c, d)\) and \((b, c, d, a)\) are circular with respect to \(H = D(S(N_1)) + D(S(N_2))\) and can be obtained from the distance matrix \(H\) using Neighbor-Net. Both orderings give a solution with zero crossings among connectors; yet, while a planar drawing for \((a, b, c, d)\) exists [see Fig. 4(a)], a drawing respecting the ordering \((b, c, d, a)\) will contain some crossings involving reticulate edges and thus fail to be optimal [see Fig. 4(b)]. However, if all optimal solutions of Neighbor-Net given \(H\) can be considered, then the NN-tanglegram approach will find the solution with cost zero. In such a case, our algorithm can be used to solve the planar layout (Lozano et al., 2008) or drawability problem (Fernau et al., 2010) for two networks [solved in linear time for two binary trees in Fernau et al. (2010)].
The heuristic for calculating an optimal tanglegram for rooted viewing a whole collection of networks or trees simultaneously. Our program Dendroscope 3 and can be used to ‘align’ the taxa when the networks or trees under consideration. This is implemented in (as implemented in Dendroscope 3).

For computing a concrete drawing of the optimized tanglegram describe a heuristic for minimizing this value and a method in a forthcoming paper (Scornavacca and Huson, 2011), we will reticulation crossing numbers for at random, whether or not it happens to minimize the sum of crossing among connectors. In such a case, one of them is chosen at random, whether or not it happens to minimize the sum of reticulation crossing numbers for $N_1$ and $N_2$ (see Definition 3.3). In a forthcoming paper (Scornavacca and Huson, 2011), we will describe a heuristic for minimizing this value and a method for computing a concrete drawing of the optimized tanglegram that tries to minimize the crossing involving reticulation edges (as implemented in Dendroscope 3).

3.3 Minimizing the reticulation crossing number and drawing the tanglegram

The previous section describes a heuristic for computing an optimal tanglegram. This heuristic aims at minimizing the crossings among connectors but it does not try to minimize the reticulation crossing numbers for $N_1$ and $N_2$. For example, if $X_1 \neq X_2$ then multiple choices of $\pi_1$ and $\pi_2$ can exist that minimize the number of crossing among connectors. In such a case, one of them is chosen at random, whether or not it happens to minimize the sum of reticulation crossing numbers for $N_1$ and $N_2$ (see Definition 3.3).

In a forthcoming paper (Scornavacca and Huson, 2011), we will describe a heuristic for minimizing this value and a method for computing a concrete drawing of the optimized tanglegram that tries to minimize the crossing involving reticulation edges (as implemented in Dendroscope 3).

4 ALIGNMENT OF PHYLOGENETIC NETWORKS

The heuristic for calculating an optimal tanglegram for rooted phylogenetic networks that we describe in Section 3.2 is easily extended to a set of more than two rooted phylogenetic networks or trees, simply by computing the matrix $H$ based on splits from all the networks or trees under consideration. This is implemented in our program Dendroscope 3 and can be used to ‘align’ the taxa when viewing a whole collection of networks or trees simultaneously.

5 VALIDATION

To validate the approach, we first report on a simulation study that we have undertaken and then apply the algorithm to a published dataset to illustrate how the algorithm may be used in practice.

5.1 Simulation study

In the first part of the simulation study, we compared our implementation with the best available software for computing tanglegrams of trees. In more detail, we compared against the bb-1st-sol algorithm, as it is the best performing algorithm of the five presented in Nöllenburg et al. (2009), and against the $b$ algorithm, as it has similar performance to the other algorithms described in Bansal et al. (2009), while being faster. Since these algorithms only accept binary trees, we first restricted our attention to binary trees. We compared these two published methods against our NN-tanglegram heuristic and two variants of it, which we will refer to as NN-tanglegram+1S and NN-tanglegram+5S. These two variants first compute $\pi_1$ and $\pi_2$ as described in Section 3.2, then define $\pi$ as the order $\pi_1$ restricted to the common taxa and finally use $\pi$ to compute the new orders $\pi_1$ and $\pi_2$ as described in Section 3.2. NN-tanglegram+5S executes this step 5 times, alternating $\pi_1$ and $\pi_2$ to compute the new $\pi$. Since the NN-tanglegram heuristic performs similarly on trees when using the splits-based distance $H$ or the shortest path distance matrix $H'$, we will present the results only for the former variant. All runs were executed on a 2.53 GHz processor with 4 GB of RAM.

For the first dataset, which we will refer to as $D_1$, we created 6 random binary trees on the same taxon set for five different sizes, namely on 20, 60, 100, 140 and 180 taxa. This dataset contains 15 instances to solve for each taxon set. Each instance was formulated as an ILP (integer linear program) as described in Nöllenburg et al. (2009) and then solved using lpSolve (freely available from lpsolve.sourceforge.net/5.5/). The number of replicates considered for each parameter setting was limited by the long running time of the ILP solver. In the second dataset, $D_2$, we created 10 random binary trees for each of the 5 sizes listed above, ensuring in each case that a tanglegram with zero crossings among connectors exists. This dataset contains 45 instances to solve for each taxon set.

For each tanglegram and each method, we computed the performance ratio ($PR$), that is, the ratio $(cn+1)/(cn_{opt}+1)$, where $cn$ and $cn_{opt}$ are the computed and the optimum number of crossings among connectors, respectively. The performance ratio values and the average running time for each method are shown in Figure 5.

For both datasets, the best-performing method is $bb$-1st-sol, having the lowest $PR$ values and the lowest average running time. Note that this method is guaranteed to find a solution with zero crossings, if one exists (Nöllenburg et al., 2009), just like our NN-tanglegram heuristic [see Fig. 5(d)]. However, the method is restricted to binary trees, unlike our method, which also applies to multifurcating trees. On the first dataset it appears to perform well and its $PR$ values are comparable with those of $bb$-1st-sol. However, when the number of instances per taxon set is increased (as in dataset $D_2$), this method can perform very badly for some instances [see Fig. 5(d)], although the average $PR$ values remain low. Moreover, the average running time of $b$ is unacceptably high for use in an interactive visualization tool [on average >100 s when the cardinality of the taxon set is 180, see Fig. 5(a,c)].

Our new method, although designed for the general case of networks, also performs well for binary trees, while the average running time is low. Comparing the performance of NN-tanglegram with NN-tanglegram+1S and NN-tanglegram+5S, we can see that the swapping step, as expected, improves the $PR$ values but increases the average running time. However, swapping one time (as done in NN-tanglegram+1S) is highly recommended because the achieved improvement of the $PR$ values is worth the small increase in running time. In our implementation, the user can choose how many times to swap or can abort the swapping procedure after a given amount of time.

In the second part of the simulation, we studied the performance of our methods on two different network datasets. For both...
Tanglegrams for rooted phylogenetic networks

datasets ($D_1$ and $D_4$), we created 15 random binary networks (not necessarily bicombining) on the same taxon set for 5 different sizes, namely on 20, 60, 100, 140 and 180 taxa, ensuring in each case that a tanglegram with zero crossings among connectors exists. This leads to 105 instances to solve for each taxon set. The two datasets differ by the probability to add a reticulate edge between two nodes, which is higher for $D_4$. (This implies that $D_4$ on average contains more reticulations than $D_3$ and thus is a more complicated dataset than the latter.)

For both datasets, we compared the performance of NN-tanglegram with NN-tanglegram+1S and NN-tanglegram+5S. Since the NN-tanglegram heuristic performs a lot better on networks when using the shortest path distance matrix $H'$ rather than the splits-based distance $H$ [see Figure 6(b), NN-tanglegram+1S vs NN-tanglegram+1S], we will discuss the results only for the former variant.

As expected, the PR values and average running times are higher than for the binary tree datasets but still acceptably low for use in an interactive visualization tool [see Fig. 6]. The PR values increase both with the number of leaves and the number of reticulations in the networks [see Fig. 6(b,d)]. The pattern of relations among NN-tanglegram, NN-tanglegram+1S and NN-tanglegram+5S is the same than for the tree datasets (i.e. the swapping step improves the PR values but increases the average running time).

In general, although the average PR values remain low, the methods can produce tanglegrams with high numbers of crossings among connectors for some instances. Note that, when the number of leaves in the networks is large, the crossing number can easily be very high. For example, if the two networks under consideration have 180 leaves each and if only one taxon is incorrectly placed at the two different ends of the networks, then the crossing number will be at least 179. Nevertheless, in this case the tanglegram may still be useful for visualizing similarities and differences among the two networks.

5.2 Application to published data

Persicaria is a genus of plants in the family Polygonaceae. In (Kim and Donoghue, 2008), the authors present evidence of hybrid speciations within this genus using cpDNA regions and nuclear ITS sequences. The strict consensus tree from the most parsimonious (MP) trees and the maximum likelihood (ML) tree were computed and drawn superposed on each other for both the cpDNA and nuclear ITS datasets. A tanglegram between the two superposed drawings...
C.Scornavacca et al.

Fig. 6. (a) Average running time (RT) and (b) performance ratio values (PR) for datasets $D_3$. (c) Average running time (RT) and (d) performance ratio values (PR) for datasets $D_4$.

(continued)

(continued)

6 CONCLUSIONS

Tanglegrams are a useful tool for comparing rooted phylogenetic trees. In this article, we have extended them to rooted phylogenetic networks and have described a practical approach to their computation. The simulation study proves that our new method, although designed for the general case of networks, also performs well for binary trees, while the average running time stays low. Moreover, the performance of our method on networks is good enough for use in an interactive visualization tool. Our implementation in the popular tree-drawing program Dendroscope will make tanglegrams for trees and networks easily accessible to biologists and other users.

ACKNOWLEDGEMENTS

The authors would like to thank Mukul S. Bansal and Martin Nöllenburg for providing them with an implementation of their methods.

Conflict of Interest: none declared.

REFERENCES


Fig. 7. A tanglegram between the networks obtained by analyzing the nrITS dataset (left) and the cpDNA dataset (right). For each dataset, the edges present only in the ML tree or in the strict consensus MP tree, are shown as bold black lines, or bold dotted gray lines, respectively. Data from (Kim and Donoghue, 2008).


A.1 Proof of Lemma 3.4

To obtain a contradiction, assume that \( \pi_0 = \pi \). Let \( C \) be a cluster in \( \mathcal{C}(N) \) that is not interval-realizable with respect to \( \pi \). Then there exist three taxa \( a, b, c \in X \) such that \( a, b \in C \), \( c \notin C \) and \( a <_\pi c <_\pi b \). Let \( v \) be the target node of a tree edge in \( N \) that represents \( C \) and let \( p_1 \) and \( p_2 \) be two paths connecting \( v \) to the leaves labeled by \( a \) and \( b \), respectively. By definition of a rooted phylogenetic network, there exists a direct path \( p_3 \) connecting \( c \) to the root node \( \rho \). Since any \( p_3 \) cannot include \( v \), the Jordan curve theorem implies that \( p_3 \) has to cross \( p_1 \) or \( p_2 \), a contradiction.

A.2 Proof of Theorem 3.5

For ease of exposition, assume that \( X_1 = X_2 \) holds. Since both \( N_1 \) and \( N_2 \) can be represented by rooted outer-labeled planar graphs, it follows from Lemma 3.4 that \( \mathcal{C}(N_1) \) and \( \mathcal{C}(N_2) \) are interval-realizable for some orders \( \pi_1 \) and \( \pi_2 \) of \( X \), respectively. From the definition of \( \Sigma(\cdot) \) we have that both \( \Sigma(N_1) \) and \( \Sigma(N_2) \) fulfill Definition 2.2 with respect to \( \pi_1 \) and \( \pi_2 \) and thus are circular split systems. Thus, by definition, \( D(\Sigma(N_1)), D(\Sigma(N_2)) \) and \( D(\Sigma(N_1)) + D(\Sigma(N_2)) \) are circular. It follows, from the consistency of Neighbor-Net (Theorem 2.3), that the split set \( \Sigma(N_1) \cup \Sigma(N_2) \) is circular with respect to the circular ordering \( \pi \) computed by the Neighbor-Net algorithm. Note that also \( \Sigma(N_1) \) and \( \Sigma(N_2) \) are circular with respect to \( \pi \) and thus the linear ordering \( \pi \) is by definition a circular ordering for \( \Sigma(N_1) \) and \( \Sigma(N_2) \) too. The definitions of \( \Sigma(\cdot) \) and of \( \pi \) imply that the sets \( \Sigma(N_1) \) and \( \Sigma(N_2) \) are interval-realizable with respect to \( \pi \).

It remains to be established that the ordering \( \pi_1 \) (similar for \( \pi_2 \)) returned by the Neighbor-Net tanglegram heuristic is such that \( \pi_1 = \pi \). To establish this, we have to show that, if \( \mathcal{C}(N_1) \) is interval-realizable with respect to \( \pi \), then there exists an embedding \( \pi_1 \) of \( \mathcal{F}(N_1) \) such that \( \pi_1 \) is a non-interleaving order w.r.t. \( \tau_1 \). Note that, for each \( T^* \in \mathcal{F}(N_1) \) on a taxon set \( X^* \), the cluster set \( C(T^*) \) is a subset of \( \mathcal{C}(N_1) \) and thus is interval-realizable with respect to \( \pi_1 \). Second, \( C(T^*) \) is compatible. This implies that we can construct an embedding \( \tau^* \) of \( T^* \) with \( \pi_1(\tau^*) = \pi_1(T^*) \) that can be drawn in such a way that no two tree edges cross. Thus, the set of topological embeddings for all trees \( T^* \) in \( \mathcal{F}(N_1) \) constitutes a topological embedding \( \tau_1 \) for \( \mathcal{F}(N_1) \) such that \( \pi_1 \) satisfies condition (1) of Definition 3.1 w.r.t. \( \tau_1 \). But \( \pi_1 \) also satisfies condition (2) of Definition 3.1 w.r.t. \( \tau_1 \), otherwise \( \mathcal{C}(N_1) \) would not be interval-realizable with respect to \( \pi \). From these observations, it follows that we can construct an embedding \( \tau_1 \) for the forest \( \mathcal{F}(N_1) \) such that \( \pi_1 \) is a non-interleaving order w.r.t. \( \tau_1 \) as described in Section 3.2. This concludes the proof.