Lpnet: Reconstructing Phylogenetic Networks from Distances Using Integer Linear Programming

Mengzhen Guo,[1] Stefan Grünewald,[*,1]
[1]Shanghai Institute of Nutrition and Health, University of Chinese Academy of Sciences, Chinese Academy of Sciences, Shanghai 200031, PR China
***Corresponding author:** E-mail: stefan@picb.ac.cn.
Associate Editor:

Abstract

We present Lpnet, a variant of the widely used Neighbor-net method that approximates pairwise distances between taxa by a circular phylogenetic network. We use integer linear programming to replace a heurisristic part of the agglomeration procedure based on local information by an exact global solution. This approach achieves an improved approximation of the input distance for the clear majority of experiments that we have run for simulated and real data. We release an implementation in R that can handle up to 94 taxa and usually needs about one minute on a standard computer for 80 taxa.

Key words: Phylogenetic networks, Neighbor-net, Integer linear programming, Circular split systems, Distance-based phylogenetics

Introduction

Neighbor-net is by far the most widely applied method to construct phylogenetic networks. The article presenting the method (Bryant and Moulton, 2004) and the paper that is associated with the software package Splitstree4 (Huson and Bryant, 2006) which implements the algorithm, have received 1896 and 7307 citations, respectively, according to Google Scholar in October 2021.

Neighbor-net is a distance-based method that approximates the input distance by (the distance induced by) a weighted circular split system on the same taxa set, which is then visualized as a splits graph. It is the network analogue of the neighbor-joining (NJ) method (Saitou and Nei, 1987). First, a circular ordering of the taxa is computed heuristically, and then a non-negative least squares (NNLS) procedure is applied to find optimal weights for all splits that agree with that circular ordering. The heuristic consists of joining two clusters, using the same criterion as neighbor-joining, and then choosing one taxon from each cluster such that those two taxa will be adjacent in the final circular ordering. The first part of this agglomeration defines a phylogenetic tree on all taxa (Levy and Pachter, 2011) and the second part chooses a circular ordering that agrees with all splits of that tree. It follows from (Semple and Steel, 2004) that there are 2^{n-3} circular orderings with that property for n taxa.

Article

Here we present Lpnet, a variant of Neighbor-net that does not apply the second heuristic step of the agglomeration. Instead, we only construct a phylogenetic tree heuristically. Then we use integer linear programming to find an optimal circular ordering, and finally we use the same NNLS procedure as Neighbor-net to assign split weights. We have run experiments to compare Lpnet and Neighbor-net, using random distances, simulated data sets, and a real data set, and we observe that Lpnet tends to approximate the input distance clearly better than Neighbor-net. Since the integer linear programming problem in Lpnet uses a quadratic number of variables and a cubic number of constraints, Lpnet is computationally more demanding than Neighbor-net. Our implementation in R can handle up to 94 taxa and usually needs about one minute on a standard computer for 80 taxa. It is available at https://github.com/yukimayuli-gmz/lpnet.

New Approach

As NJ and Neighbor-net, Lpnet uses a matrix with pairwise distances between the taxa as its input. However, all of these methods can be interpreted as quartet-based in the following sense: A quartet $ab|cd$ consists of two disjoint pairs of taxa $\{a,b\}$ and $\{c,d\}$. A weight of $ab|cd$ can be defined for every quartet, based on the pairwise distances between taxa in $\{a,b,c,d\}$ and quantifies the support for separating a and b from c and d. Then all objective functions and selection criteria of the agglomerative part of the considered distance-based methods can be written as functions of the weights of some quartets. This has been noted in (Mihaescu *et al.*, 2009) and (Levy and Pachter, 2011), and we will give details in the Materials and Methods section, but now we will only use the quartet interpretation to describe the differences between Lpnet and Neighbor-net. The Lpnet algorithm can be divided into three steps:

1) Construct a phylogenetic tree from the input distance matrix. We use several methods to do this construction, including NJ.
2) Use Integer Linear Programming to find a circular ordering which is consistent with the tree to maximize the sum of all quartet weights contained in the circular ordering.
3) Estimate split weights from the circular ordering by using non-negative least squares.

The tree construction and the split weights estimation are similar to what NJ, Neighbor-net and other tree or network algorithms do. We only sketch the part using Linear Programming in this section, because this step is the main distinction between Lpnet and Neighbor-net.

Neighbor-net combines the agglomeration of bottom-up clustering with a second selection step that essentially defines an ordering of the taxa in the newly added cluster. Every cluster can be considered a path with its taxa as vertices, and the path will be an interval in the final circular ordering. When two clusters are merged,

one end vertex from each path is selected, and those vertices are connected to a new path for the union of the clusters. The selection maximizes a weighted sum of all quartets that are supported by every circular ordering containing the merged path but not by every ordering containing the two short paths. Crucially, the decision about the ordering of the new cluster is made based on the information available at the time of merging.

For Lpnet, we choose to delay the ordering of the clusters until the whole tree is known. We do so by solving an integer linear programming problem that finds a circular ordering of all taxa that maximizes the sum of the weights of its supported quartets among all orderings that agree with all splits of the tree.

Use Linear Programming to maximize all quartet weights

When a phylogenetic tree is drawn in the plane, this embedding defines a circular ordering of the taxa which can be observed by traversing the tree such that every edge is visited once in each direction. The circular orderings associated with a given tree have been studied by Semple and Steel (Semple and Steel, 2004) who showed that, for a binary tree with n taxa, there are exactly 2^{n-3} orderings. Starting with an embedding, every possible ordering can be obtained by flipping an arbitrary subset of the interior edges.

In terms of supported quartets, for every four taxa a,b,c,d, there is one quartet, say $ab|cd$, that is displayed by the tree in the sense that there is at least one split separating a,b from c,d. Every circular ordering that agrees with the tree will support $ab|cd$ and one of the other two quartets $ac|bd$, $ad|bc$. If the ordering of an initial embedding of the tree supports $ad|bc$, then another ordering will support $ad|bc$, if and only if the number of flipped edges separating a,b from c,d is even. We search all orderings that agree with a tree by starting with an initial embedding and introducing a binary variable $X_{i,j}$ for every pair of interior vertices which will be set to 1, if an odd number of edges on the path between i and j is flipped, and to 0 else. We introduce linear constraints that make sure that the allowed assignments to the variables correspond precisely to the circular orderings that agree with the tree. Then the sum of the weights of all supported quartets that are not displayed by the tree is a linear function of the variables, and we can globally maximize this objective function using binary linear programming.

Results

In this section, we mainly compare the performance of Neighbor-net and Lpnet for different input distances. In order to evaluate different networks from the same input, we use the LSFit, which is the normalised sum of the squares of the difference between input and output distances. Before the split weights are computed, Neighbor-net and Lpnet both try to maximize the sum of the weights of the quartets that agree with the chosen circular ordering. Therefore, we use

this sum to evaluate the algorithms at that state and refer to it as *sum of quartets*. Within Lpnet, the result may depend on the choice of the tree reconstruction method. We use Neighbor-joining and its common variants BioNJ and UNJ, as well as two methods that try to mimic the internal tree building of Neighbor-net that we call *NNet tree* and *symmetric NNet tree* (see the Materials and Methods section for details).

We run examples on four different kinds of input distances: First we present a simple artificial example with only seven taxa and two reticulations that shows how the early ordering of clusters by Neighbor-net can cause problems. The second example uses random distances between taxa and represents the general approximation problem of an input metric by a circular split system, without any phylogenetic signal. The third example uses simulated sequences from random trees. The last example is a published morphological data set of *Aphelandra* plants which contains artificial hybrids and has been suggested as a benchmark for phylogenetic networks methods.

An artificial example

We start with an artificial example that demonstrates the disadvantage of ordering clusters locally. As visualised in Figure 1, we assume that seven taxa mainly evolved under a clocklike tree, with the exception of two gene transfer events I_1 and I_2 which independently involve 10% and 20% of the genome, respectively. This means that the genome consists of four parts representing sequences affected by one or both or none of the insertions. Each part follows its own tree, and the observed distance is a convex combination of those four tree distances where the coefficients are the fractions of the genome that follow the trees. This distance corresponds to ten non-trivial splits which do not fit on any circular ordering, and seven trivial splits. The non-trivial splits with the weights and the contributing trees are listed in Table 1.

When Neighbor-net is applied to this distance, it correctly identifies the clusters {1,2} and {3,4} first and then decides to join those clusters. The second selection criterion of Neighbor-net chooses to make 2 and 3 neighbours in the circular ordering, because it relies on quartets that have at most one taxon from {5,6,7}. This decision based on local information makes it impossible to later include the cluster {3,5}, which corresponds to a much stronger split than {2,3}. Lpnet first correctly finds the whole main tree and then chooses a circular ordering that allows all true splits with a weight higher than 0.1.

As can be seen from Table 1, the weights of the correct splits in Lpnet tend to be clearly closer to the true weight than Neighbor-net, and even 23|14567, the only true split that is allowed by the Neighbor-net ordering and not by the Lpnet one, finally gets a negligible weight in the Neighbor-net. Both methods achieve a very high LSFit (Neighbor-net 99.99445 and Lpnet 99.99965), but

Table 1. Nontrival splits and positive weights for the original phylogenetic tree and the three trees representing transfer events, the mixture of the input trees and for Neighbor-net and Lpnet.

original tree (72%)		TI_1 (8%)		TI_2 (18%)		TI_{12} (2%)			Tree mixture	NNet	Lpnet
split	weight	split	weight	split	weight	split	weight	split	weight	weight	weight
12	4			12	4			12|34567	3.6	3.6	3.6
34	4							12567|34	2.88	2.65	2.8375
567	2	567	2					1234|567	1.6	1.6	1.6
		23	1					14567|23	0.08	6.215 e-16	
		234	4					1567|234	0.32	0.35	0.35
				35	2	35	1	12467|35	0.38		0.375
				345	4			1267|345	0.72	0.9	0.7125
				67	2	67	2	12345|67	0.4	0.4	0.4
						167	4	167|2345	0.08	0.1	0.1
						235	1	1467|235	0.02		
								12347|56	0		3.73 e-16

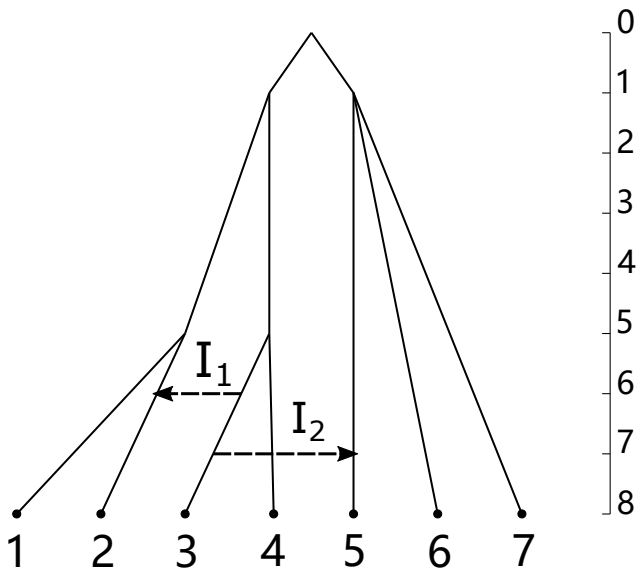

FIG. 1. A tree-like phylogentic network with two reticulations.

the gap to 100 is still more than fifteen times greater for Neighbor-net than for Lpnet.

Random distances

We use random numbers between 0 and 1 from the uniform distribution as pairwise distances, and then add a large enough constant to all distances to guarantee the triangle inequality.

We generate 10000 random distance matrices with 30 taxa. Then we compare Lpnet using different methods to construct phylogenetic trees, with Neighbor-net. We use SplitsTree4 (Huson and Bryant, 2006) with default setting to get the Neighbor-net result. Then we compare the sum of quartets and the LSFit value for Lpnet and Neighbor-net (Table 2). We observe that for all five tree building methods, for the sum of quartets and LSFit, the Lpnet algorithm clearly tends to get better scores than Neighbor-net. While Lpnet achieves a higher LSFit for roughly 80% of the input metrics, this fraction is more than 98% for the sum of quartets. Comparing the Lpnets using different tree building methods, we find that we often get the same circular ordering. Nevertheless, Table 3 shows that UNJ performs significantly better than the other methods.

Simulated sequences

We randomly generate a tree for 30 taxa by using the function 'sim.taxa' from the R package TreeSimGM (Hagen and Stadler, 2018). We let the parameter 'waiting time until speciation' for 'sim.taxa' be exponentially distributed with the rate parameter $\lambda = 1.2$. Then we use the software Dawg (Cartwright, 2005) to simulate

Table 2. Comparison of Lpnets with Neighbor-net in terms of the LSFit value and the sum of quartets, for 10000 random distances and five different methods of tree construction.

	Same circular ordering	LSFit		sum of quartets	
	as Neighbor-net	Lpnet > NNet	Lpnet < NNet	Lpnet > NNet	Lpnet < NNet
Neighbor-joining	0	7795	2205	9838	162
Symmetric NNet tree	0	8323	1677	9974	26
NNet tree	0	8320	1680	9977	23
UNJ	0	8362	1638	9975	25
BioNJ	0	8056	1944	9895	105

Table 3. Comparison of Lpnets using UNJ with Lpnets using four other tree reconstruction methods for 10000 random distances.

	Same circular ordering	LSFit		sum of quartets	
	as UNJ	> UNJ	< UNJ	> UNJ	< UNJ
Neighbor-joining	1097	3977	4926	3383	5520
Symmetric NNet tree	5489	2158	2353	1990	2521
NNet tree	3965	2870	3165	2509	3526
BioNJ	2097	3716	4187	3537	4366

DNA sequences of length 10000bp from the random tree under the Jukes-Cantor model. Finally, we use SplitsTree4 to compute Jukes-Cantor distances with default settings. We repeat this process 10000 times and compare Lpnet and Neighbor-net in the same way as for the random metrics. Table 4 shows the result of comparing the LSFit value and the sum of all quartets between Lpnet and Neighbor-net. We see that for all five tree construction methods, the advantage of Lpnet compared to NeighborNet increases. The sum of quartets is now always higher and the LSFit better for almost 95% of the data sets when we use Lpnet. The input data sets for this experiment can be interpreted as a tree metric plus some random noise, and the results show that in this situation the strategy of Lpnet to complete the tree reconstruction before embedding the tree pays off. The various tree building methods yield the same circular ordering more often than for random metrics, but again UNJ achieves significantly better scores than the other variants of NJ (see Table 5).

Analysis of a published data set

For real data, we choose a study of the *Aphelandra pulcherrima* complex by Lucinda McDade (McDade, 1990, 1984, 1992, 1997). The raw data is a phenotype matrix for 12 species of *Aphelandra*, a genus of flowering plants, and 17 artificial hybrids which each have two of the 12 species as parents (McDade, 1990). We use the Hamming difference from the binary phenotype data to compute the Neighbor-net (Fig 2) and the Lpnet (Fig 3). Following the previous results, we chose UNJ as the tree reconstruction method for Lpnet.

This data set has been proposed as an example for testing phylogenetic networks method by an influential but now inactive blog (http://phylonetworks.blogspot.com/p/datasets.html). The advantage is that we know the hybrids and their parents, so ideally a phylogenetic network

Table 4. Comparison of Lpnets with Neighbor-net in terms of the LSFit value and the sum of quartets, for 10000 simulated sequence alignments.

	Same circular ordering	LSFit		sum of quartets	
	as Neighbor-net	Lpnet > NNet	Lpnet < NNet	Lpnet > NNet	Lpnet < NNet
Neighbor-joining	0	9473	527	10000	0
Symmetric NNet tree	0	9473	527	10000	0
NNet tree	0	9473	527	10000	0
UNJ	0	9474	526	10000	0
BioNJ	0	9472	528	10000	0

Table 5. Comparison of Lpnets using UNJ with Lpnets using four other tree reconstruction methods for 10000 simulated sequence alignments.

	Same circular ordering	LSFit		sum of quartets	
	as UNJ	> UNJ	< UNJ	> UNJ	< UNJ
Neighbor-joining	9436	243	321	229	335
Symmetric NNet tree	9500	211	289	209	291
NNet tree	9511	206	283	202	287
BioNJ	9439	240	321	225	336

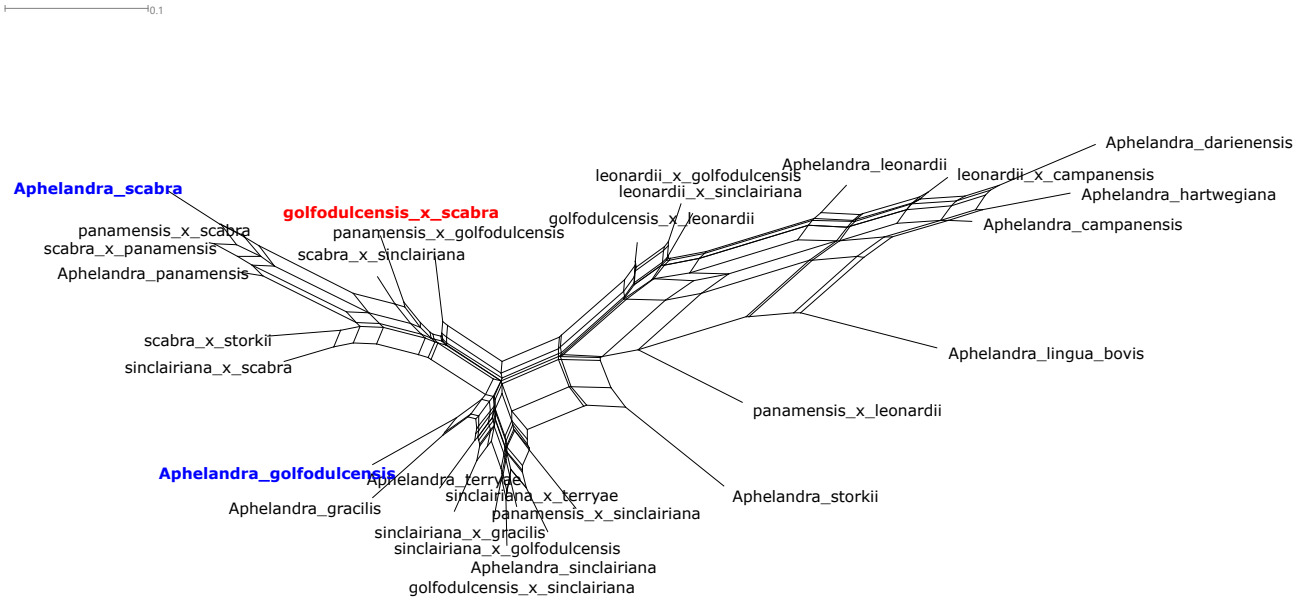

FIG. 2. Neighbor-net. Aphelandra_golfdulcensis and Aphelandra_scabra (blue) are the parents of golfdulcensis_x_scabra (red).

would show the correct tree of the non-hybrids and then group the hybrids together with both parents. However, the data for the non-hybrids is already not very tree-like, and several species are parents of many hybrids. Since Neighbor-net and Lpnet are both limited to circular split systems, they can not display all true phylogenetic relationships. We can still compare the two methods by counting the number of hybrids that are placed appropriately in the networks. We define a hybrid to be placed correctly, if 1) the hybrid is on the shorter of the two paths between the parents in the circular ordering, and 2) there is no split that separates both parents on one side

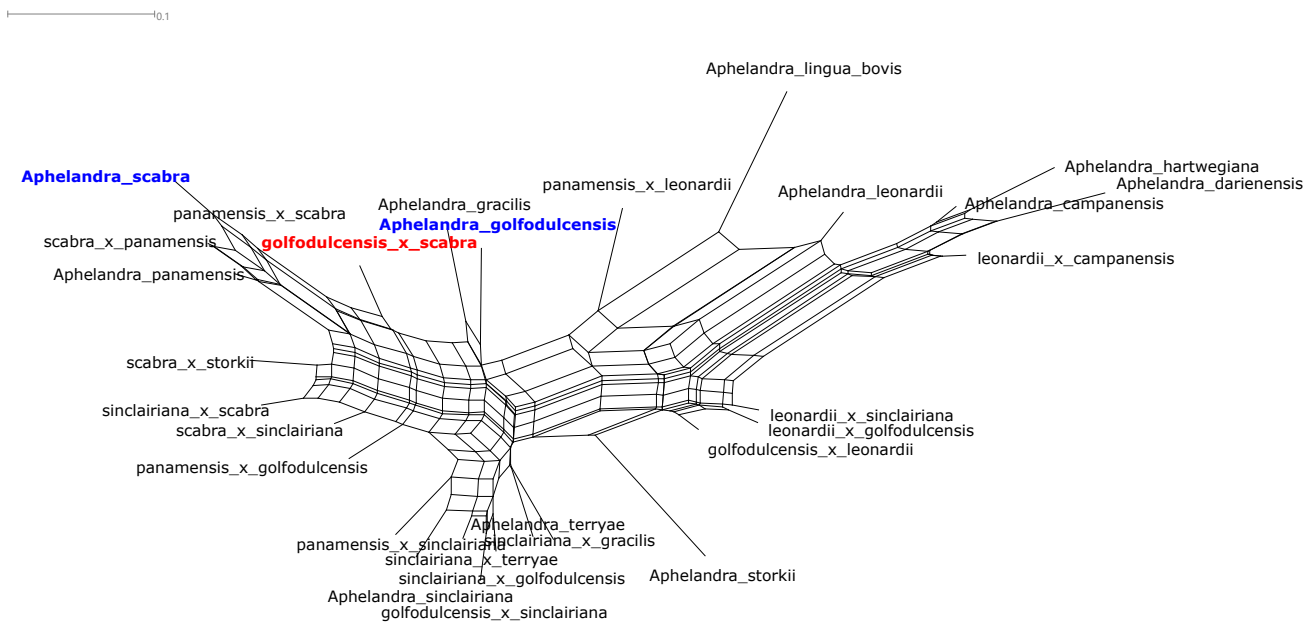

FIG. 3. Lpnet. Aphelandra_golfdulcensis and Aphelandra_scabra (blue) are the parents of golfdulcensis_x_scabra (red).

from the hybrid and a non-hybrid species on the other side.

In Figs 2 and 3, we highlight the hybrid golfdulcensis_x_scabra (GS) in red, and its parents *Aphelandra_golfdulcensis* (GO) and *Aphelandra_scabra* (SC) in blue. We observe that in the Neighbor-net, GS is not placed between GO and SC. In the Lpnet, GS is between GO and SC, and *Aphelandra_gracilis* (GR) is the only non-hybrid species between GO and SC. Since there is no split separating GS and GR from GO and SC, the hybrid GS is placed correctly. Doing this analysis for all hybrids yield that Lpnet places 6 out of 17 hybrids correctly, and Neighbour-net only 4.

Comparing the approximation error, the LSFit value of Neighbor-net is 98.802, and for Lpnet it is better (99.135). We conclude that for the somewhat subjective number of correctly placed hybrids, as well as for the LSFit value, Lpnet is an improvement.

Discussion

Seventeen years after the release of Neighbor-net, our new variant Lpnet provides an alternative that approximates the input distance better for the clear majority of the data sets we have tried. The main disadvantage of Lpnet is that it is slower and needs more memory, but most data sets that have been analyzed by Neighbor-net have less than 80 taxa and can therefore be handled by Lpnet as well.

The main application of split graphs and in particular Neighbor-nets is to find the main signals in an early stage of a data analysis (Huson and Bryant, 2006). In practice, a Neighbor-net often contains a few strong splits and many tiny

ones which are usually interpreted as irrelevant noise. We expect that the clear signals will often be detected by both methods, while differences between the minor splits will cause a slightly higher LSFit value for Lpnet. In such cases it does not matter much which method is used.

However, our realistic artificial example demonstrates that there can be significant differences. If reticulations are anticipated in a data set, then the goal has to be to reconstruct an explicit phylogenetic network as shown in Figure 1. While it is generally hard to guess that network from a splits graph, this task would be easier for the Lpnet than for the Neighbor-net. We therefore anticipate that Lpnet will be useful for interpreting real data sets in the future.

We provide five different algorithms to construct phylogenetic trees. All of them turn out to yield the best LSFit occasionally, and we have no strong preference. In the average, UNJ performed best for our data sets, and it seems most consistent with the general approach taken by Lpnet that treats all pairs of taxa and all quartets equal. Therefore, we select UNJ as the default method, but we recommend to try other methods as well. Other tree building methods like minimum evolution or even not distance-based methods like maximum likelihood might be worth trying, and any binary tree can be input to Lpnet. It is generally interesting to see whether all splits of a reasonable tree will have positive weights in the Lpnet, and to compare the weights of the strongest other splits with the weights of the conflicting tree splits. However, users should be aware that we only have a consistency proof for the neighbor-joining variants that we provide.

Even though we replaced a heuristic part of Neighbor-net by an exact algorithm, Lpnet is still a heuristic method. It relies on a heuristic tree construction, and it optimizes the sum of quartet weights, while the final score function for a weighted split system is the LSfit value. The weights of the supported quartets indicate but do not guarantee that the distance can be approximated well by the allowed splits, and the discrepancy causes almost all cases where the LSFit of Lpnet is worse than Neighbor-net. It would be desirable to have a method that directly optimizes the least squares fit, but this would not allow any agglomerative construction and we are not aware of any such algorithm other than trying every possible ordering.

QNet (Grünewald *et al.*, 2007) is the quartet analogue of Neighbor-net. It uses quartet weights directly obtained from the raw data instead of distances to reconstruct a weighted circular split system. The strategy of Lpnet to first construct a tree and then use linear programming to get a circular ordering, can also be applied to modify QNet, and the approximation is expected to improve.

Materials and Methods

Compatible splits and circular split systems

A split S divides a set X of taxa to two nonempty parts A and B and is denoted $S=A|B$. Two splits $A_1|B_1$ and $A_2|B_2$ are *compatible*, if at least one of the intersections $A_1 \cap A_2$, $A_1 \cap B_2$, $B_1 \cap A_2$, $B_1 \cap B_2$, is empty. For a phylogenetic tree where the leaf nodes are labeled by the set X of taxa, every branch of the tree represents a split of X and all splits are compatible with each other (Semple and Steel, 2003).

Three splits $S_1=A_1|B_1$, $S_2=A_2|B_2$ and $S_3=A_3|B_3$ are *weakly compatible*, if there are no four taxa $\{t,u,v,w\}$ with $\{t,u\} \in A_1$, $\{v,w\} \in B_1$, $\{t,v\} \in A_2$, $\{u,w\} \in B_2$, $\{t,w\} \in A_3$, $\{u,v\} \in B_3$ (Bandelt and Dress, 1992). A *circular ordering* of a set $X=\{x_1,...,x_n\}$ of $n \geq 3$ taxa can be obtained by labelling all vertices of an n-gon by the taxa (see Fig 4). A split $A|B$ *agrees* with a circular ordering, if both A and B label consecutive paths on the circle. A circular ordering is defined by the permutation of X that we get by starting at an arbitrary taxon and then following all vertices of the cycle in a clocklike or anti-clocklike fashion. Therefore, there are $2n$ different permutations associated with the same circular ordering.

A *circular split system* of X is a set of splits of X such that all splits agree with a single circular ordering. It has long been known that circular split systems are weakly compatible and have up to $\binom{n}{2}$ splits (Bandelt and Dress, 1992). Further, compatible split systems are circular, thus an unrooted phylogenetic tree can be considered a circular split system. When non-negative weights are assigned to the splits, this *weighted circular split system* can be visualised by a planar *splits graph*, a network where the taxa are embedded as vertices, and every split $A|B$ of weight w corresponds to a set of parallel edges which all have length w. Further, removing all those edges decomposes the network into two connected components containing A and B, respectively (Dress and Huson, 2004). Splits graphs are commonly referred to as implicit phylogenetic networks, and Splitstree4 can be used to draw such a network from an input weighted circular split system (Huson and Bryant, 2006).

Every weighted split system induces a pairwise distance on the taxa where the distance between two taxa x and y is the sum of the weights of all splits that separate x from y.

Quartets and weights

As mentioned before, Neighbor-joining and its variants can be considered a quartet method (Mihaescu *et al.*, 2009). For a distance d, and four taxa u,v,x,y, we define the weight of the quartet $uv|xy$ by $w(uv|xy)=d(u,x)+d(u,y)+d(v,x)+d(v,y)-2d(u,v)-2d(x,y)$. If d is induced by a weighted split system, then $w(uv|xy)$ equals four times the sum of the weights of all splits that separate u,v from x,y minus two times the sum of the weights of all splits that support a conflicting quartet. Therefore, the weight of a quartet quantifies the support for separating

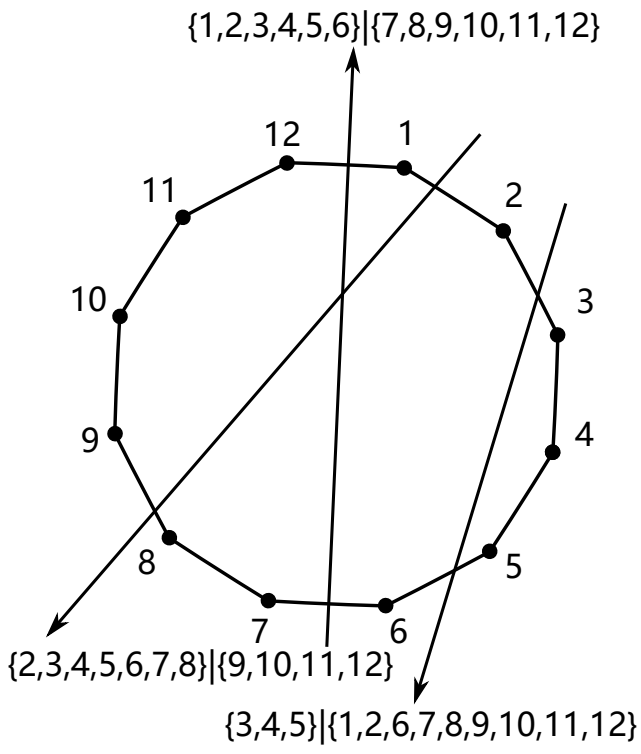

FIG. 4. A circular split system and three splits.

its pairs. The support for a tree (without edge lengths) or a circular ordering can be quantified by summing up the weights of all supported quartets, and this number is maximized by a correct tree or ordering, if the input distance is induced by a tree or a circular split system, respectively.

Distance reduction and the order dependence of Neighbor-net

The first step of neighbor-joining and all other agglomerative algorithms discussed here is to identify two taxa u,v, such that the sum of all quartet weights $w(uv|xy)$ is maximized. Then the cluster $\{u,v\}$ is considered a single taxon, where the distances between this new taxon and another taxon x is is based on the distances between u, v, and x. This process is reiterated until there are only three taxa left, and the two steps are called the *selection* and the *reduction* step. It was pointed out in (Bryant, 2005) that, in order to reconstruct trees correctly from their induced metric, the selection step is unique, while the reduction step can give different weights to the two joined clusters. Neighbor-joining always gives the same weight to both clusters, while UNJ uses weights that are proportional to the cluster size, and BioNJ tries to minimize the variance of the reduced distances.

Neighbor-net does not reduce the distance for clusters of size two, because the distinction helps to compute a circular ordering agglomeratively. Instead, it reduces three taxa to two, whenever a cluster of size two is merged with another cluster. If both clusters have size two, this reduction step has to be performed twice. This distinguishes the taxon that is not included in the first reduction from the other three. For default parameters, the taxa that are first reduced each receive 2/9 and the remaining taxon 1/3 of the total weight of the new cluster. Since the choice which three taxa are reduced first is not determined by the input distances, the output of Neighbor-net can sometimes depend on the input order of the taxa, even if no ties occur. It therefore seems reasonable to give equal weights to all four clusters. We have implemented this variant of the Neighbor-net tree construction in Lpnet and refer to it as *symmetric NNet tree*. In addition, we use *NNet tree* to mimic the original weighting by randomly assigning 1/3 to one of the two candidate taxa that might receive that weight from the Neighbor-net algorithm.

The Lpnet algorithm

As we have said in the New Approaches section, the Lpnet algorithm uses a distance matrix as its input. First it constructs a phylogenetic tree from the distances, then it uses Linear Programming to find a circular ordering which maximizes the sum of all quartet weights consistent with the circular ordering. Finally, it computes weights for all splits that agree with the circular ordering such that the LSFit is maximized.

Constructing a tree

Neighbor-net constructs a circular ordering agglomeratively. The process can be described as adding edges to a graph whose vertices are the taxa such that every component is a path (Grünewald *et al.*, 2007). It starts with the graph without edges and joins two paths in every iteration, until a single path is obtained which then defines a circular ordering. All taxa sets of the paths form a compatible cluster system and therefore define a binary tree. The difference to methods that only construct a tree is that after selecting two paths to be merged, there are up to four pairs of taxa that can be connected to get a single path. One such pair is chosen by the second selection criterion of Neighbor-net which maximizes the sum of the weights of all quartets that are supported by all circular orderings containing the new path.

As indicated in the previous subsection, the second selection step has some influence on the tree construction. It was already suggested in (Levy and Pachter, 2011) to change the distance reduction of Neighbor-net such that all splits of the neighbor-joining tree are always supported by the output ordering. The main idea of Lpnet is to skip the second selection altogether and choose a circular ordering after the tree construction. In order to observe how much of the performance difference between Lpnet and Neighbor-net is caused by this new strategy, we include the NNet tree which stays as close as possible to Neighbor-net. Noting the undesired order dependence of Neighbor-net, we also include the symmetric NNet tree.

The other tree reconstruction methods implemented by Lpnet are neighbor-joining and its variants, UNJ and BioNJ. It is also possible to input a user-defined tree, and Lpnet will compute a circular split system where all splits of the input tree agree with the circular ordering.

Using Linear Programming to maximize quartet weights

Given a binary unrooted tree T with n taxa and pairwise distances, we want to find a circular ordering that agrees with all splits of T, such that the sum of the weights of all supported quartets is maximized. Given an initial ordering that agrees with T, we can obtain another such ordering by choosing a non-trivial split $A|B$ and reversing the order of A. Note that reversing the order of B yields the same circular ordering as

reversing A, and reversing both yields the initial circular ordering. This process can be interpreted as flipping the edge that separates A from B, and it follows from (Semple and Steel, 2004) that all allowed circular orderings can be obtained by a sequence of edge flips. Moreover, the final circular ordering does not depend on the order in which the edges are flipped, flipping an edge twice yields the same circular ordering as never flipping it, and flipping two different sets of edges always results in different circular orderings. Therefore, there are 2^{n-3} different circular orderings that agree with T, and they are in one-to-one correspondence with the subsets of the interior edges of T.

For four taxa i_1,i_2,j_1,j_2, assume that the quartet $i_1i_2|j_1j_2$ is displayed by T. Then all edges corresponding to splits that separate i_1 and i_2 from j_1 and j_2 form a path in T. One of the end vertices of that path, say i, is on the path from i_1 to i_2 and the other one, say j, is on the path from j_1 to j_2. If the initial circular ordering supports the quartet $i_2j_1|j_2i_1$, then another circular ordering also supports that quartet, if the number of flipped edges on the path from i to j is even, else it supports $i_1j_1|i_2j_2$. Let I_1,I_2,J_1,J_2 be the sets of taxa that are in the same component of the graph obtained from T by removing i and j as i_1,i_2,j_1,j_2, respectively. Then we define

$$c_{ij} = \sum_{a_1\in I_1, a_2\in I_2, b_1\in J_1, b_2\in J_2,} w(a_1b_1|a_2b_2) - w(a_2b_1|b_2a_1),$$

so c_{ij} quantifies how much we would like to flip an odd number of edges on the path from i to j. Introducing binary variables, X_{ij} for every pair $\{i,j\}$ of interior vertices of T, we need to find a circular ordering such that $X_{ij}=1$, if and only if the number of flipped edges on the path from i to j is odd, and $\sum_{i,j} c_{ij}X_{ij}$ is maximal. It turns out that a (0,1)-assignment to all variables corresponds to a circular ordering, if and only if the following four linear inequalities hold for every three interior vertices i,j,k of T:

$$\begin{aligned} X_{ij} + X_{ik} + X_{jk} &\leqslant 2 \\ -X_{ij} + X_{ik} + X_{jk} &\geqslant 0 \\ X_{ij} - X_{ik} + X_{jk} &\geqslant 0 \\ X_{ij} + X_{ik} - X_{jk} &\geqslant 0. \end{aligned}$$

These conditions are necessary, because every edge of the smallest subtree of T containing i,j,k is contained in exactly two of the three paths between two of those vertices. This means that the sum $X_{ij}+X_{ik}+X_{jk}$ is even, so either all three variables are zero or there are two ones and one zero. To see that the conditions are sufficient, we note that the (0,1)-assignment to all those variables X_{ij} where ij is an edge of T already defines a circular ordering. Now there is a single extension of this assignment to all variables such that all conditions hold: Let i and k be two interior vertices of T such that X_{ik} is unknown while X_{ij} and X_{jk} have already been assigned for a vertex j. As before, $X_{ij}+X_{ik}+X_{jk}$ must be even, so there is only one allowed assignment for X_{ik}.

In summary, we compute an optimal circular ordering by solving a binary linear programming problem with $\binom{n-2}{2}$ variables and $4\binom{n-2}{3}$ constraints.

Estimate split weights by Non-negative least squares

The last part of Lpnet is to compute weights for all splits that agree with the circular ordering that was constructed in the previous step. Here we follow the same approach as Neighbor-net and use the NNLS algorithm by (Lawson and Hanson, 1995) to obtain split weights such that the difference between the input distance and the distance induced by the weighted circular split system is minimized. More precisely, we minimize the Euclidian norm of $\mathbf{A}\vec{\mathbf{s}}-\vec{\mathbf{d}}$ where $\mathbf{A}$ is the (0,1)-matrix with rows indexed by the pairs of taxa and columns indexed by the allowed splits, and an entry is one, if and only the pair of taxa corresponding to its row is separated by the split corresponding to its column. Further, $\vec{\mathbf{s}}$ and $\vec{\mathbf{d}}$ are the column vectors with the non-negative split weights and the input distances, respectively. In order to measure the goodness of distance approximation, we use the least squares fit (LSFit):

$$\mathrm{LSFit} = \left(1 - \frac{(\sum_{ij}(d_{ij}-p_{ij})^2)}{(\sum_{ij} d_{ij}^2)}\right) \times 100$$

where the pairwise distances induced by the weighted split system are p_{ij} and the input distances are d_{ij}.

Consistency of Lpnet

Consistency is an important feature of phylogenetic reconstruction methods. It means that a method does not make mistakes for perfect input. Specifically, a method that reconstructs a circular split system from distances is *consistent*, if it returns the correct weighted split system whenever the input distance is induced by a weighted circular split system. Neighbor-net was shown to be consistent in (Bryant *et al.*, 2007). A more general proof was given by (Levy and Pachter, 2011), where all tree construction methods used by Lpnet are included in their definition of neighbor-joining which allows a wide class of weighting schemes for the reduction step. Their result implies that all splits of the tree constructed by Lpnet agree with some circular ordering that agrees with all splits of the underlying split system. It is easy to see that such an ordering will also maximize the sum of all supported quartets. Finally, NNLS will be able to match the input distance exactly, thus Lpnet is consistent for all used tree reconstruction methods.

Implementation

We provide an R implementation of Lpnet that allows the user to choose one of the five tree reconstruction methods listed above or to input a tree. For the linear programming problem, two solvers are supported: The R version Rglpk of the GNU Linear Programming

Kit (http://www.gnu.org/software/glpk) is free and open source, while Gurobi (http://www.gurobi.com) is one of the most powerful commercial mathematical optimization solvers. A free license of Gurobi is available for academic institutions.

In practice, the integer linear programming is the computationally most demanding part of Lpnet. The constraint matrix has $O(n^5)$ entries, and the required memory grows equally fast. There is a hard limit of 94 taxa, because the matrix is stored as a vector, and R only allows vectors of length at most $2^{31}-1$. Using Gurobi, we were able to compute solutions for all input distances with up to 94 taxa that we tried.

We list the size of the constraint matrix and the average CPU time for running our Lpnet function in R using Gurobi with binary linear programming for different numbers of taxa in Table 6. Rglpk is much slower, and the problem was often not feasible for more than 50 taxa on our machine. For 40 taxa, we observed an average CPU time of 13.76 seconds.

Integer linear programming is NP-hard (Karp, 1972), and solvers will often solve the relaxed problem where the variables are allowed to be non-integer first. For the problem solved by Lpnet, this solution happens to be an integer solution most of the time. If there are non-integer entries, the running time will increase. For example, we observed this case for less than 1% of random distances for 80 taxa and Gurobi, and then the average CPU time was 307.27 sec. In an attempt to estimate the worst case, we constructed an example where the optimal solution of the relaxed problem contains no integer at all. For 80 taxa, Gurobi needed almost 14 hours for the solution. The distance matrix is available as no_integer.nex in the examples folder of our R package at https://github.com/yukimayuli-gmz/lpnet.

Table 6. Sizes of the constraint matrix and CPU times for different number of taxa

number of taxa	CPU time	size of constraint matrix
20	0.0805 sec	3.81 MB
40	1.375 sec	180.985 MB
60	13.662 sec	1.52 GB
80	60.41 sec	6.809 GB

In summary, our Lpnet implementation can be used for up to 94 taxa, and with the Gurobi solver a solution will usually take at most a few minutes. The Rglpk solver works well for smaller instances but will struggle for more than 50 taxa. All CPU times reported here were obtained running Lpnet on a laptop with Windows 10 operating system, Intel Core i7-9750H 2.60 GHz CPU with 6 cores, and 16 GB of RAM.

References

Bandelt, H.-J. and Dress, A. W. 1992. Split decomposition: a new and useful approach to phylogenetic analysis of distance data. *Molecular phylogenetics and evolution*, 1(3): 242–252.

Bryant, D. 2005. On the uniqueness of the selection criterion in neighbor-joining. *Journal of Classification*, 22(1): 3–115.

Bryant, D. and Moulton, V. 2004. Neighbor-net: an agglomerative method for the construction of phylogenetic networks. *Molecular biology and evolution*, 21(2): 255–265.

Bryant, D., Moulton, V., and Spillner, A. 2007. Consistency of the Neighbor-net algorithm. *Algorithms for Molecular Biology*, 2(1): 1–11.

Cartwright, R. A. 2005. DNA assembly with gaps (Dawg): simulating sequence evolution. *Bioinformatics*, 21(Suppl_3): iii31–iii38.

Dress, A. and Huson, D. 2004. Constructing splits graphs. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1(3): 109–115.

Grünewald, S., Forslund, K., Dress, A., and Moulton, V. 2007. QNet: an agglomerative method for the construction of phylogenetic networks from weighted quartets. *Molecular Biology and Evolution*, 24(2): 532–538.

Hagen, O. and Stadler, T. 2018. TreeSimGM: Simulating phylogenetic trees under general Bellman–Harris models with lineage-specific shifts of speciation and extinction in R. *Methods in ecology and evolution*, 9(3): 754–760.

Huson, D. H. and Bryant, D. 2006. Application of phylogenetic networks in evolutionary studies. *Molecular biology and evolution*, 23(2): 254–267.

Karp, R. M. 1972. Reducibility among combinatorial problems. In R. E. Miller, J. W. Thatcher, and J. D. Bohlinger, editors, *Complexity of Computer Computations: Proceedings of a symposium on the Complexity of Computer Computations, held March 20–22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, and sponsored by the Office of Naval Research, Mathematics Program, IBM World Trade Corporation, and the IBM Research Mathematical Sciences Department*, pages 85–103. Springer US, Boston, MA.

Lawson, C. L. and Hanson, R. J. 1995. *Solving least squares problems*. SIAM.

Levy, D. and Pachter, L. 2011. The neighbor-net algorithm. *Advances in Applied Mathematics*, 47(2): 240–258.

McDade, L. 1990. Hybrids and phylogenetic systematics I. Patterns of character expression in hybrids and their implications for cladistic analysis. *Evolution*, 44(6): 1685–1700.

McDade, L. A. 1984. Systematics and reproductive biology of the central american species of the Aphelandra pulcherrima complex (Acanthaceae). *Annals of the Missouri Botanical Garden*, pages 104–165.

McDade, L. A. 1992. Hybrids and phylogenetic systematics II. The impact of hybrids on cladistic analysis. *Evolution*, 46(5): 1329–1346.

McDade, L. A. 1997. Hybrids and phylogenetic systematics III. Comparison with distance methods. *Systematic Botany*, pages 669–683.

Mihaescu, R., Levy, D., and Pachter, L. 2009. Why neighbor-joining works. *Algorithmica*, 54(1): 1–24.

Saitou, N. and Nei, M. 1987. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular biology and evolution*, 4(4): 406–425.

Semple, C. and Steel, M. 2003. *Phylogenetics*, volume 24. Oxford University Press.

Semple, C. and Steel, M. 2004. Cyclic permutations and evolutionary trees. *Advances in Applied Mathematics*, 32(4): 669–680.