TriLoNet: Piecing together small networks to reconstruct reticulate evolutionary histories

James [Oldman]$^{\sharp,1}$, Taoyang Wu$^{\sharp,1}$, Leo van Iersel^{2}, Vincent Moulton^{*,1}
[1] School of Computing Sciences, University of East Anglia, Norwich, United Kingdom
[2] Delft Institute of Applied Mathematics, Delft University of Technology, The Netherlands.
♯These authors equally contributed to this work.
***Corresponding author:** E-mail: vincent.moulton@cmp.uea.ac.uk.
Associate Editor: ?
?

Abstract

Phylogenetic networks are a generalisation of evolutionary trees that can be used to represent reticulate processes such as hybridisation and recombination. Here we introduce a new approach called *TriLoNet* to construct such networks directly from sequence alignments which works by piecing together smaller phylogenetic networks. More specifically, using a bottom up approach similar to Neighbor-Joining, TriLoNet constructs level-1 networks (networks that are somewhat more general than trees) from smaller level-1 networks on three taxa. In simulations we show that TriLoNet compares well with Lev1athan, a method for reconstructing level-1 networks from three-leaved trees. In particular, in simulations we find that Lev1athan tends to generate networks that overestimate the number of reticulate events as compared with those generated by TriLoNet. We also illustrate TriLoNet's applicability using simulated and real sequence data involving recombination, demonstrating that it has the potential to reconstruct informative reticulate evolutionary histories. TriLoNet has been implemented in JAVA and is freely available at https://www.uea.ac.uk/computing/TriLoNet.

Key words: Phylogenetic network, reticulate evolution, networks reconstruction, supernetwork

Article

Introduction

Phylogenetic networks are a generalisation of evolutionary trees that can be used to represent reticulate evolutionary processes such as horizontal gene transfer, hybridisation and recombination (Bapteste *et al.*, 2013). The importance of such processes in genome evolution is becoming increasingly appreciated, and several approaches have been introduced to compute phylogenetic networks in recent years (see Gusfield, 2014; Huson *et al.*, 2010; Nakhleh, 2011; Woolley *et al.*, 2008, and the references therein). There are various types of phylogenetic networks (cf. Huson *et al.*, 2010); in this article we are interested in *rooted networks* which aim to explicitly represent reticulate events. As with rooted evolutionary trees, these networks have vertices and branches or arcs, a single root vertex,

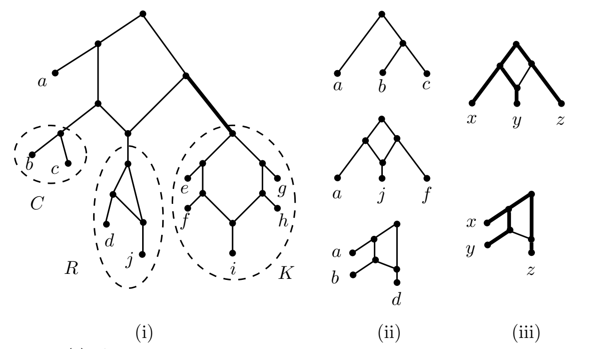

FIG. 1. (i) A level-1 phylogenetic network N, with a cherry C, a reticulated cherry R and a cactus K indicated by the dotted ellipses. The bold arc is a cut arc since its removal disconnects the network. (ii) Three of the trinets displayed by N. (iii) Two level-1 networks that display different trinets but exhibit the same set of triplets; the bold arcs indicate how the triplet $xy|z$ is exhibited.

and their leaves are labelled by the taxa, also known as OTU's. However, unlike trees, they can contain vertices with more than one parent, giving rise to *cycles*. For example, in Fig. 1(i) we present a rooted network which contains three cycles, and which represents the evolutionary history of the taxa $a,b,\ldots,j$. The vertices with two parents in the cycles, or *reticulate vertices*, each represent a reticulate evolutionary event.

Several methods have been developed for constructing rooted networks, with some implemented in software packages such as PhyloNet (Than *et al.*, 2008), PADRE (Lott *et al.*, 2009), TripNet (Poormohammadi *et al.*, 2014), and Dendroscope 3 (Huson and Scornavacca, 2012). In this paper we focus on constructing *level-1 networks* (also known as *galled trees* in Wang *et al.*, 2001), an important family of rooted networks in which no two distinct cycles share a common vertex. These networks are appropriate for situations where modest amounts of reticulation is believed to have occurred (Gusfield, 2014) and they have been used to, for example, represent the evolution of the fungus *F. graminearum* (Huson *et al.*, 2010), and that of HIV and yeast (Huber *et al.*, 2011). Current methods for computing level-1 networks (Huber *et al.*, 2011; Jansson and Sung, 2006; Jansson *et al.*, 2006) aim to exhibit a set of triplets (rooted trees with three leaves), and one is implemented in the Lev1athan software (Huber *et al.*, 2011). All of these triplet-based methods can be regarded as extensions of the well-known Aho algorithm (Aho *et al.*, 1981) and its derivatives (Semple *et al.*, 2004) for constructing a tree from a collection of triplets.

A general issue with the current triplet-based approaches for computing level-1 networks is that they are not *consistent*. In other words, even if their input consists of all of the triplets exhibited by a level-1 network, they do not necessarily output that network (cf. Gambette and Huber, 2012). To understand why this is the case, consider the two simple level-1 networks on the three leaves x,y,z in Fig. 1(iii). These two networks both exhibit the triplets $xy|z$ and $xz|y$, and so any triplet based method will be confounded by the problem of not being able to distinguish between these networks for every subset of three taxa. A similar problem also arises for larger networks containing cycles with four nodes. In addition, when applying triplet-based approaches to sequence alignments, it is first necessary to compute triplets. This is typically done by computing phylogenetic trees on separate regions of a sequence alignment, breaking these trees up into triplets and then combining them to make a collection of triplets (Huber *et al.*, 2011). Although not necessary, in practice this can mean that breakpoints also need to be computed, which can be challenging (Lemey *et al.*, 2009).

Here we introduce a new algorithm called *TriLoNet (***Tri***net* **L***evel-***o***ne* **Net***work algorithm)* to build level-1 networks. The method works by piecing together three-leaved, level-1 networks or *trinets* (see e.g. Fig. 1(ii)). In particular, TriLoNet can be thought of as a supernetwork method for constructing rooted networks from smaller rooted networks (cf. e.g. Grunewald *et al.* (2013); Huson *et al.* (2004) for examples of supernetwork approaches for unrooted networks). In contrast to triplets, the trinets displayed by a level-1 network do determine the network (Huber and Moulton, 2012). Essentially, the problem illustrated by the two networks in Fig. 1(iii) does not arise as there is only one possible trinet on each subset of three taxa displayed by a network, a fact that we exploit to show that TriLoNet is consistent. In addition, we develop a method to compute trinets from a sequence alignment without the need to compute breakpoints, thus eliminating the need to preprocess alignments. This provides a way to infer networks directly from sequences, which is an important goal in the theory of phylogenetic networks (Yu *et al.*, 2014, p.16453).

TriLoNet uses a bottom up approach that is similar in style to the Neighbor-Joining algorithm (Saitou and Nei, 1987). Essentially, as with Neighbor-Joining which selects a cherry at each stage, TriLoNet identifies either a (possibly reticulated) cherry or a cactus that hangs off the bottom of a level-1 network as illustrated by the dotted ellipses C, R, and K in Fig. 1(i). It then replaces the selected cherry or cactus with a single leaf, recomputes the trinet set, and continues to iteratively look for cherries and cactuses until a level-1 network is constructed. This yields a polynomial time algorithm whose full description is presented in the Materials and Methods section, and whose consistency is proven in the Supplementary Material. Note that alternative algorithms have been presented for deciding whether or not a collection of trinets fits perfectly on a level-1 network (e.g. Huber and Moulton, 2012; Huber *et al.*, 2015) but, unlike TriLoNet, they are unable to construct a network for more general collections of trinets that do not fit perfectly on any level-1 network.

Results and Discussion

We refer to the Materials and Methods section for the terminology used in this section.

Comparison study

We begin by analysing the effect of introducing noise into a set of trinets that is consistent with a level-1 network for both TriLoNet and Lev1athan. The idea of this approach is to see how the two methods perform as the data becomes less and less like that for a given level-1 network. To do this, we used an experimental scheme adapted from Huber *et al.* (2011). Central to this scheme is a parameter ϵ which gives the percentage of noise to be introduced. In particular, for each ϵ equal to $0,1,2,5,10,15,20,25,30$, using the network generator in Huber *et al.* (2011) we obtain a collection $\mathcal{M}$ of random level-1 networks that contains 100 networks with leaf sizes in the range $1+(10\times j)$ to $10\times(j+1)$ for each $2\leq j\leq 9$. Then, for every network M in $\mathcal{M}$, we constructed the trinet collection $\mathcal{T}_\epsilon(M)$ by randomly replacing ϵ% trinets in $\mathcal{T}(M)$, the collection of all trinets displayed by M, with ones of different type chosen uniformly at random. To construct a triplet collection with a noise level comparable to that of the trinet collection, we also consider the collection $\mathrm{Tr}_\epsilon(M)$ of all triplets that are exhibited by some trinet in $\mathcal{T}_\epsilon(M)$. The collections $\mathcal{T}_\epsilon(M)$ and $\mathrm{Tr}_\epsilon(M)$ are then used as inputs for TriLoNet and Lev1athan, respectively.

To measure the similarity between the input network M and output network N, we compute the triplet consistency score

$$C'(N,M)=\frac{|\mathrm{Tr}(N)\cap\mathrm{Tr}(M)|}{|\mathrm{Tr}(M)|}$$

as defined in Huber *et al.* (2011) and a trinet consistency score $C(N,M)$ using the same formula with $\mathrm{Tr}(N)$ and $\mathrm{Tr}(M)$ replaced by $\mathcal{T}(N)$ and $\mathcal{T}(M)$, respectively. Here $\mathrm{Tr}(N)$ and $\mathrm{Tr}(M)$ denote the collection of all triplets exhibited by N and M, respectively. Note that both scores take on values in $[0,1]$. Moreover, $C(N,M)=1$ implies that N is equal to M (Huber and Moulton, 2012), although this does not necessarily hold for the C'-score (cf. Huber *et al.*, 2011, p.643).

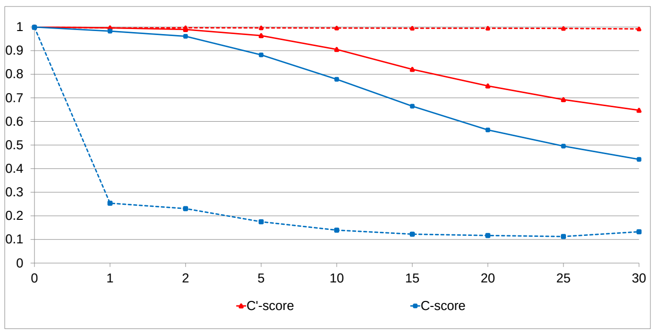

FIG. 2. The plot of ϵ (x axis) against the average triplet (lines with triangles) and trinet (lines with squares) consistency score (y axis). TriLoNet scores are indicated by solid lines, whilst those for Lev1athan by dashed lines.

The average C- and C'-scores that we obtained over all inputs are summarised in Fig. 2. Note that when $\epsilon=0$ (i.e. there is no noise), the average C-score for TriLoNet is 1, as expected, and 0.999 for Lev1athan. So, for a very small portion of networks in $\mathcal{M}$ Lev1athan outputs a slightly different network, possibly due to the small cycle problem mentioned above. For the C'-score, Lev1athan performs very well and has an average score close to one, although this does not imply that it produces networks identical to the input ones. As probably to be expected, when ϵ increases, the average C-score decreases for both TriLoNet and Lev1athan, but TriLoNet has much higher C-score, which indicates a higher topological similarity to the input network in terms of trinets. For instance, for $\epsilon=5$, the average C-score for the networks constructed by TriLoNet is 0.88 whilst 0.17 for Lev1athan. In addition, we also computed the difference between the number of reticulation vertices in input and output networks for TriLoNet and Lev1athan. The results are summarised in Fig. 3. They indicate that compared to TriLoNet, Lev1athan tends to generate networks with more reticulations than necessary to represent the input data.

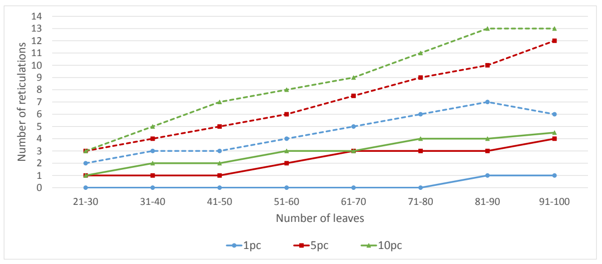

FIG. 3. The median of the difference between the number of reticulations in the networks constructed by TriLoNet (solid lines) and Lev1athan (dashed lines) compared with the input for the networks generated in the noise study. The x-axis is labelled by the number of leaves and the y-axis by the median differences. For $\epsilon = 1,5,10$, ϵ-percent means that the input data sets for the algorithms are respectively from the collection $\mathcal{T}_\epsilon(M)$ and $\mathrm{Tr}_\epsilon(M)$ as detailed in the text.

Simulated Data

We also studied the behaviour of TriLoNet on simulated sequence data. Following the scheme detailed in Holland *et al.* (2002, p.2054) for identifying recombinants, we generated artificial multiple sequence alignments for the six level-1 networks $N_1,\ldots N_6$ given in Fig. 4. Briefly, each network N_i contains precisely one reticulate vertex, the parent of taxon R_i, and two trees: the left (resp. right) tree consists of all the arcs of N_i except the arc directed towards the reticulate vertex from the right-hand (resp. left-hand). In particular, the taxon R_i represents a single recombinant sequence. For each network N_i, we then generated 100 DNA alignments of length 50,000bp on nine sequences $a,b,\cdots,h,R_i$ by concatenating two subalignments of length 25,000bp that were simulated respectively along these two trees using Seq-Gen (Rambaut and Grass, 1997) with the K2P model and transition-transversion bias 4.

We ran TriLoNet on the resulting alignments. We found that for networks N_4,N_5 and N_6 (for which the left and the right trees are more symmetrical), the TriLoNet networks were the same as the generating network in all 100 runs and for networks N_1 and N_2, that they were the same for 94 and 96 out of 100 runs, respectively. For network N_3, the output tended not to be identical to N_3, but it still shared 83% of the trinets with N_3 on average. Note that network N_3 also caused difficulties for recombinant detection in Holland *et al.* (2002). A closer inspection of the output networks indicated that they differed from the input mainly because they contained a cherry with taxa a and b (data not shown), whilst a and b do not form a cherry in N_3. We also repeated the simulations with sequences length 100,000bp, and obtained similar results (data not shown).

Biological data

To illustrate the applicability of our method, we present its application to three data sets for which some reticulate events have been documented in the literature.

HIV: We first consider an HIV data set consisting of eight HIV sequences with length 9953bp, representing subtypes A,B,D,F,G,J,H, as well

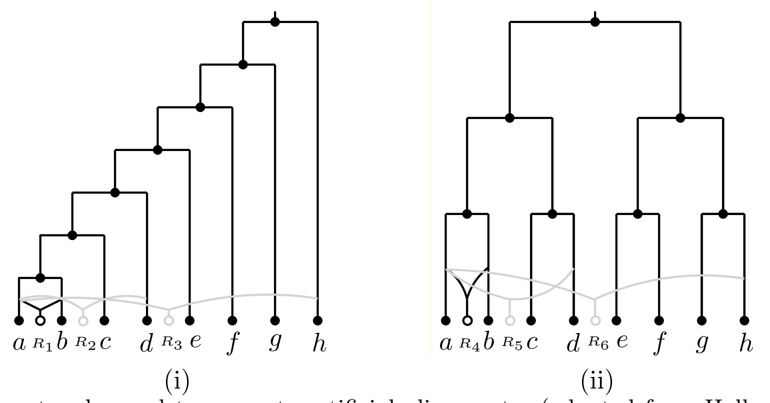

FIG. 4. Six level-1 networks used to generate artificial alignments (adapted from Holland *et al.*, 2002, Fig. 6). Here N_1 (left) and N_4 (right) are drawn in black, while N_i (i=2,3) (resp. i=5,6) is obtained from N_1 (resp. N_4) by replacing the parent of R_1 and the three arcs incident with it by the parent of R_i and the three arcs incident with it (in light grey). Branch lengths are drawn to scale; the expected number of substitutions from the root to each leaf is 0.3.

as KAL-153, a recombinant sequence between subtypes A and B (cf. Lemey *et al.*, 2009, Chapter 16).

This data set was also used to illustrate Lev1athan in Huber *et al.* (2011). Since Lev1athan is not designed to construct level-1 networks directly from sequence alignments, Huber *et al.* (2011) first constructed three gene trees using Neighbor Joining and the two breakpoints inferred in Lemey *et al.* (2009). Then, to obtain a network with KAL-153 being the only recombinant sequence, Huber *et al.* (2011) had to use a variant of Lev1athan that explicitly assumed that only one reticulate event had occurred and also restricted their analysis to the triplets derived from two of three gene trees. Note also that Huber *et al.* (2011) reported that two other network reconstruction methods, the cluster network and the galled network as implemented in Huson *et al.* (2007), had problems too, postulating between 2 and 4 reticulation vertices (Huber *et al.*, 2011, p.646).

In Fig. 5(i) we present the TriLoNet network based on the whole sequence alignment which was computed without any additional assumptions. This indicates that one reticulate event took place. However instead of KAL-153, it identifies H as being the recombinant sequence. To explore this further, we reran our analysis with sequence H removed. In Fig. 5(ii) we display the resulting network, in which KAL-153 now appears as a recombinant of A and B subtypes, which agrees with the analysis in Lemey *et al.* (2009). In particular, this indicates that TriLoNet has the potential to identify recombinant sequences, although some care needs to be taken when interpreting results.

It is interesting to also compare the TriLoNet for this data set with the split network in Fig. 6 generated by the NeighborNet

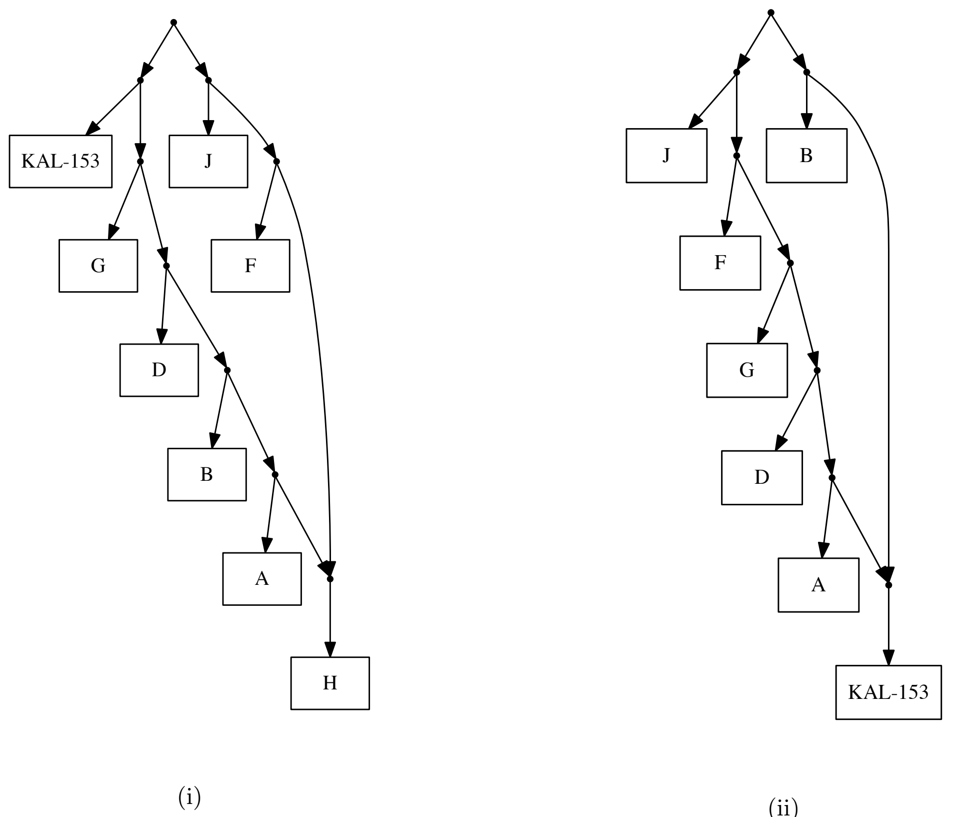

FIG. 5. Phylogenetic networks inferred by TriLoNet for the HIV data set in the text.

algorithm (Bryant and Moulton, 2004) implemented in SplitsTree (Huson and Bryant, 2006) using the default settings. This network is not rooted and so, in contrast to the TriLoNet, the edges do not have directions. The network does however display bipartitions or splits of the data that are supported by the Hamming distance matrix calculated directly from the sequence alignment. The splits are represented by sets of parallel edges in the network all having the same length. In the split network we see a split represented which separates KAL-153 and subtype A from the rest of the subtypes, and another split which separates KAL-153 and subtype B from the rest of the subtypes. This is consistent with KAL-153 being a recombinant of subtypes A and B. The main difference is that, as the split network is not rooted, it does not represent an explicit evolutionary history for the data set, whereas the TriLoNet does (although it should be emphasised that NeighborNet was not designed to do this).

Giardia: We now consider a giardia data set, which consists of seven sequences with lengths approximately 17,000bp concatenated from three partial chromosomes, 3, 4, 5 with lengths roughly 6,000, 1,500 and 9,500 respectively (Cooper *et al.*, 2007). Isolate WB represents genotype A1 and all other isolates genotype A2. In addition, sequences 303 and 305 are identical, and isolate 335 is believed to be a recombinant (Cooper *et al.*, 2007).

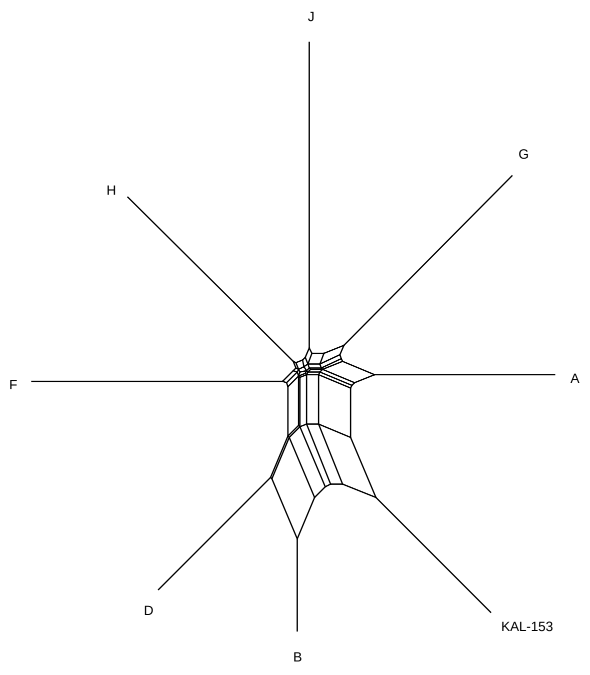

FIG. 6. The split network constructed by the NeighborNet method for the HIV data set.

As the segments from the three chromosomes are known *a priori*, and have quite different lengths, we experimented with introducing some scaling into trinet calculation (see Step A1 below) to take these lengths into account. In particular, for $i=3,4,5$, denoting by n_i the length of chromosome segment i, and by $w_i(xy|z)$ the number of the sites on chromosome segment i for which sequences x and y have the same character while x and z have different characters, we replaced the quantity $w(xy|z)$ in the computation of trinets by

$$\frac{n_3w_3(xy|z)+n_4w_4(xy|z)+n_5w_5(xy|z)}{n_3+n_4+n_5}$$

and do similar replacement for $w(yz|x)$ and $w(xz|y)$.

The networks inferred by TriLoNet with and without rescaling incorporated are given in Fig. 7(i) and Fig. 7(ii), respectively. For reference, we also picture the split network generated by the NeighborNet approach in Fig. 2 of the Supplementary Material. In both of the networks generated by TriLoNet, WB appears as an outgroup, as should be the case. Moreover, the two TriLoNets are quite similar although the second one postulates that 335 is a recombinant and is more representative of the three tree topologies given for each of the three chromosome segments presented in Cooper *et al.* (2007, Figure 3). More specifically, sequences 55, JH, 335 cluster together in the second network, which is in general agreement with the analysis presented in Cooper *et al.* (2007, p.1984). This analysis suggests that TriLoNet is again able to produce some informative histories, and also that it could be useful to rescale when prior breakpoint information is known concerning the input alignment.

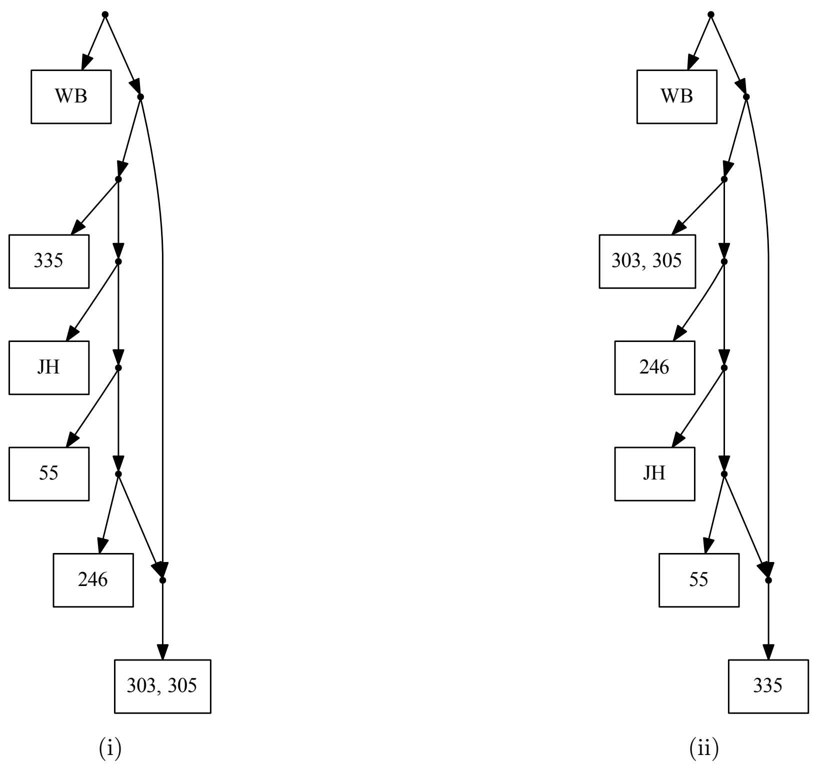

FIG. 7. Phylogenetic networks inferred by TriLoNet for the giardia isolates.

HBV: To investigate the applicability of TriLoNet to larger data sets, we considered 25 HBV sequences of length 3229bp sampled from five genotypes (denoted A-D and F) that were presented in Bollyky *et al.* (1996). In Fig. 8 we present the TriLoNet network. As can be seen, genotypes F, C and A all appear within different clusters within the network, and genotype D is almost clustered together, with the exception for the recombinant sequence HPBADW1. The split network generated by the NeighborNet approach (see Fig. 3 in the Supplementary Material) also clusters together the five genotypes.

For the two recombinant sequences between different genotypes identified in Bollyky *et al.* (1996) the network identifies one, HBVDNA, with parent sequences from genotype A and D, as reported in Bollyky *et al.* (1996). This is also reflected in the NeighborNet, where there are splits which group HBVDNA with both A and D genotypes, in a similar fashion to the KAL-153 recombinant in the HIV data set above. The other, HPBADW1, does not appear as a recombinant, but instead as a leaf in the cactus that also contains HBVDNA as a leaf. When we removed HBVDNA from the analysis, HPBADW1 was subsequently identified as a recombinant sequence of genotypes A and B by TriLoNet (see Fig. 4 in the Supplementary Material), which is in line with the analysis in Bollyky *et al.* (1996). Also, in this network genotype D is disentangled from A and B groupings, which the HBVDNA sequence probably brought together being a recombinant of A and D genotypes. This suggests that although TriLoNet is able to identity potential recombinants, a higher level network might be necessary to provide a better representation of this particular data set.

Conclusion

We have introduced and implemented a novel method called TriLoNet to infer level-1 networks directly from sequence alignments without having to, for example, compute breakpoints. It is the first supernetwork approach to construct rooted networks from real data by putting together smaller networks into a larger one, much like triplet-based supertree methods.

Our simulations indicate that the new approach compares favourably with the Lev1athan method for inferring level-1 networks from triplets. First, in simulations TriLoNet produces networks that are topologically more similar to the input networks, based on trinet-comparison. Moreover, TriLoNet does not require additional assumptions, and does not appear to add in as many additional reticulation vertices to represent the data. This may be related to the fact that trinets determine level-1 networks whereas triplets do not in general. In addition, our artificially generated alignments indicate that TriLoNet is quite good at reconstructing level-1 network topologies for some fairly simple scenarios without requiring breakpoints, and the real data sets illustrate that our method is able to build networks that can be helpful for understanding reticulate histories. In particular, TriLoNet could be useful for combining small networks computed using model based approaches. Note that this approach has proven useful for constructing phylogenetic trees (see, e.g. Schmidt *et al.*, 2002).

There are various directions in which our method might be extended. For example, Lev1athan can work with partial triplet data, and so it would be interesting to develop a method that can cope with missing trinets. However, this could be challenging in view of the recent hardness result by Huber *et al.* (2015). In addition, instead of subnetworks with three leaves, one could consider subnetworks with larger number of leaves, say the so-called quarnets (subnetworks with four-leaves). However, the number of quarnet topologies is much larger than that of trinets, and inferring quarnets from sequence alignments could be much more complicated. Even so, it could be of interest to construct quarnets that always include an out group as a first step.

Another interesting direction could be to develop better ways of computing trinets from sequence data. One possibility could be a likelihood approach which would require the development of appropriate models. However, this could be challenging since even though there are methods for computing likelihoods for networks (e.g. Yu *et al.*, 2014), these are not immediately applicable as they work by computing likelihood on trees in the network which do not necessarily determine the network even when branch lengths are known (Pardi and Scornavacca, 2015). In this regard the recent work of Nguyen

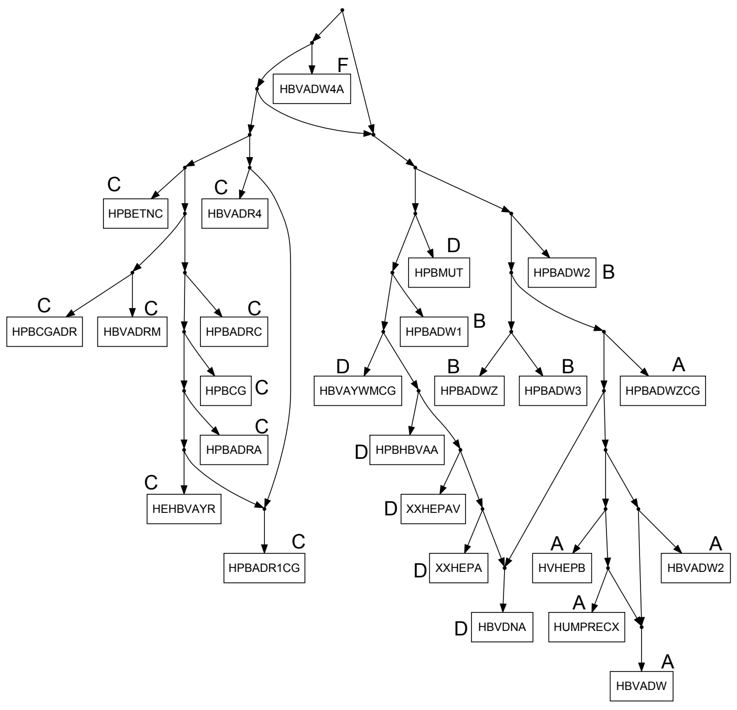

FIG. 8. The level-1 network constructed by TriLoNet for the HBV data set.

and Roos (2015) might hold some promise, as it does not require trees to compute networks, although it would have to be adapted to ensure that it always generated level-1 trinets.

It would also be of interest to consider higher level networks. Although a level-1 network is useful to model and represent the reticulate processes in some data sets, we have seen in our HBV example that higher level networks could be more appropriate for more complex data sets. In this direction, it is known that the trinets in a so-called level-2 network (i.e., a binary network in which each of the components obtained from removing all cut arcs contains at most two reticulation vertices) determine the network (van Iersel and Moulton, 2014), and so a method to construct level-2 networks should be feasible. However, some careful thought will be necessary as to how to compute level-2 trinets, as these are more complex and numerous than level-1 trinets, and it will probably also be much more intricate to put level-2 trinets together. In this regard, it might make more sense to restrict to a simpler subset of level-2 trinets.

In conclusion, we believe that our supernetwork based reconstruction method is a useful alternative for inferring informative networks, especially for data sets with a small number of reticulate events. We hope that this approach will serve to inspire new methods for constructing rooted networks by puzzling together small networks, a strategy that has already proven its worth for phylogenetic trees.

Materials and Methods

Phylogenetic networks

We begin by presenting some preliminaries concerning networks. A *rooted phylogenetic* *network* N, or a network for short, is a directed graph containing a unique root with neither directed cycles nor vertices that have one parent and one child, and in which each leaf is uniquely labelled by a taxon from a given set of taxa. A network is *binary* if each vertex has at most two children, and at most two parents, and those vertices with two parents (the *reticulations*) have one child. In addition, a network is *level-1* if each reticulation is contained in precisely one (undirected) cycle (Huson *et al.*, 2010). Such networks are also known as galled trees (Gusfield, 2014). Unless stated otherwise, all our networks are level-1. An arc in a network is called a *cut arc* if it is not incident with a leaf and its removal disconnects the network. Note that given a cut arc $\{u,v\}$ in a level-1 network such that there is no cut arc below v, the network consisting of v and all vertices and arcs below v is either a cherry (i.e., the two vertices below v are both leaves), a reticulated cherry (i.e., v and its two children form a cycle and the two leaves below v are incident with this cycle) or a cactus (i.e., v is in a cycle such that all of the vertices below v are either in the cycle or incident with a vertex in the cycle) with three or more leaves; see Fig. 1 (i).

The building blocks used in our algorithm are networks with two and three leaves, known respectively as *binets* and *trinets*. As depicted in Fig. 9. there are precisely two types of binets and eight types of trinets (up to relabelling the leaves) (Huber and Moulton, 2012). Note that all the trinets have a cut arc except for those of type S_1 or S_2.

A binet or trinet T is *displayed* by a network N if there exists a vertex u in N such that T can be obtained from N by deleting all vertices and arcs that are not on a directed path from u to a taxon contained in T and then repeatedly suppressing vertices with one parent and one child and replacing parallel arcs by single arcs until neither operation is applicable. The set consisting of all trinets displayed by N is denoted by $\mathcal{T}(N)$. Note that it is necessarily *dense*, that is, it contains precisely one trinet for each combination of three taxa. It is known that a binary level-1 network is encoded by the collection of trinets that it displays (Huber and Moulton, 2012).

Note that trinet $T_1(x,y;z)$ in Fig. 9 is just a tree or *triplet* and is also denoted by $xy|z$; note that the other two triplets on these three taxa are $xz|y$ and $yz|x$. The triplet $xy|z$ is *exhibited* by a network N if $T_1(x,y;z)$ can be obtained from the trinet T in $\mathcal{T}(N)$ with leaf set $\{x,y,z\}$ by deleting some (or none) arcs and suppressing the resulting vertices that have one parent and one child. For instance, $xy|z$ is exhibited by $S_2(x;y;z)$ but not by $S_1(x,y;z)$. Note that if the trinet $T_1(x,y;z)$ is displayed by N, then the triplet $xy|z$ is exhibited by N, but the converse does not necessarily hold. For example, triplet $xy|z$ is exhibited by $S_2(x;y;z)$, but the trinet $T_1(x,y;z)$ is not displayed by $S_2(x;y;z)$. The set of triplets exhibited by N is denoted by $\mathrm{Tr}(N)$.

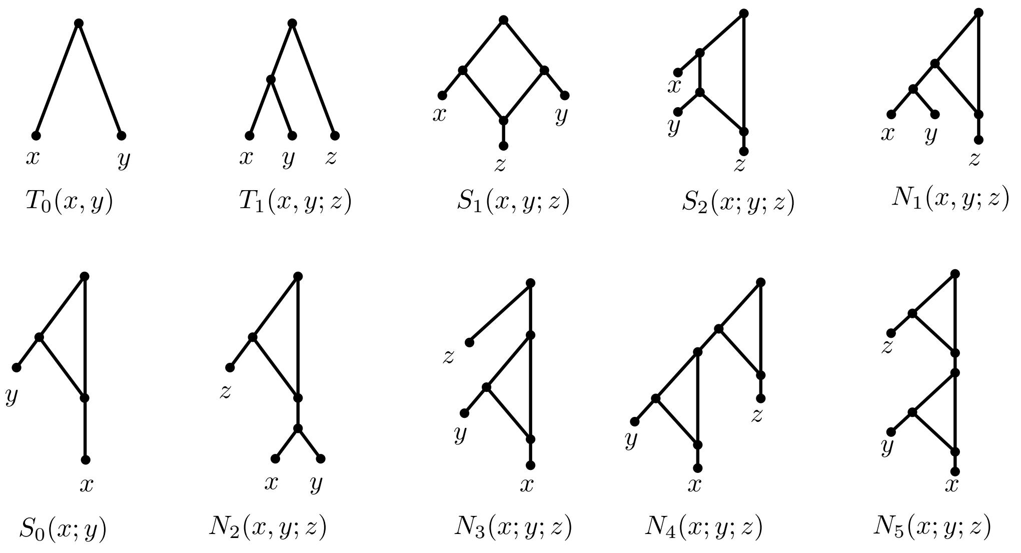

FIG. 9. The two types of binets and the eight types of trinets.

Trinets from sequences

The first stage in our approach is to compute a dense set of trinets from a multiple sequence alignment (MSA) on a given set of taxa X. More precisely, for each triple of taxa in X, we assign a trinet to the triple using the following three steps.

Step A1: For each triple $t=\{x,y,z\}$ of taxa x,y,z from X, we consider the subalignment of the MSA on x,y and z. For each of the three possible triplets on t, say $xy|z$, we compute a weight $w(xy|z)$ defined as the number of sites in the subalignment such that the character states (e.g., nucleotides for DNA) are the same for x and y and different to the one for z. In addition, assuming $w(xy|z)\geq w(xz|y)\geq w(yz|x)$, we introduce the following score

$$\delta_t=\frac{w(xy|z)-w(xz|y)}{w(xz|y)-w(yz|x)},$$

with the convention $\delta_t=w(xy|z)-w(xz|y)$ if the denominator in the definition equals zero. Intuitively, this score indicates whether the trinet associated to t should contain a cut arc or not. In other words, a higher δ_t score gives greater support for assigning t a trinet that contains a cut arc separating x and y from z. Note that this δ_t score is closely related to the δ-score used to measure 'tree-likeness' in statistical geometry (see, e.g. Holland *et al.*, 2002, and the references therein).

Step A2: Using the score δ_t computed in the first step and a threshold κ, we partition the set of triples of taxa from X into two subsets. The first is Σ_κ that contains all triples of taxa whose δ-score is greater than or equal to κ. All other triples form the second subset, denoted by Σ^c_κ. The basic idea is that a triple in Σ^c_κ is less likely to contain a cut arc and hence will be assigned a trinet of type S_1 or S_2, while a triple in Σ_κ will be assigned to a trinet of other types.

To obtain a κ value for applications, we simulated sequences along a representative collection of weighted trinets for all eight types of trinets in Fig. 9 (see the Supplementary Material for more details). For each of the weighted trinets, we generated and concatenated sequences along all trees with three leaves embedded in the trinet using the K2P model with transition-transversion bias 4 and computed the δ-scores. In most cases a κ value of 6 or 7 could correctly distinguish trinets with types S_1 and S_2 from the other types of trinets. We therefore took a default value of $\kappa = 6.5$.

Step A3: In this step we assign a trinet of type S_1 or S_2 to each triple $t = \{x_1, x_2, x_3\}$ in Σ_κ^c. Without loss of generality, we assume that the number of triples in Σ_κ containing x_i is greater than or equal to that containing x_j for $1 \leq i < j \leq 3$. Let w_t be the minimum weight among the three triplets on t. Then we assign the unique trinet of type S_1 or S_2 such that this trinet contains x_1 below its reticulate vertex and does not exhibit the triplet with the minimum weight. More precisely, we assign the trinet $S_1(x_2, x_3; x_1)$ to t if $w_t = w(x_2 x_3 | x_1)$, and $S_2(x_2; x_3; x_1)$ if $w_t = w(x_1 x_2 | x_3)$, and $S_2(x_3; x_2; x_1)$ otherwise. We denote the set of trinets obtained in this step by $\mathcal{T}_S$.

Step A4: The last step is to assign a trinet to each triple $t = \{x, y, z\}$ in Σ_κ. For simplicity, assume as before that the triplet $xy|z$ has the maximum weight amongst the three possible triplets on t. We then assign a trinet T to t in which there exists a cut arc separating x and y from z (i.e. a trinet T of the form $T_1(x, y; z)$, $N_1(x, y; z)$, $N_2(x; y; z)$, $N_3(x; y; z)$, $N_4(x; y; z)$ or $N_5(x; y; z)$) so that the number of trinets that have already been assigned to some triple and share a binet with T is maximised.

Cut arc sets

As mentioned in the Introduction, a fundamental step in our algorithm is the selection of a (possibly recticulated) cherry or a cactus. These lie below cut arcs in the network, and so we shall now explain how subsets of leaves that lie below a cut arc can be related to certain subsets of the taxa that can be derived by just considering the trinets displayed by the network.

To this end, we call a subset A of the leaf set of a network a *cut arc (CA-) set* if there exists a cut arc (u, v) in the network such that A contains precisely the taxa below v. Since a cut arc is not incident with a leaf, a CA-set contains at least two taxa. For example, the CA-sets of network N in Fig. 1 are $\{b, c\}$, $\{d, j\}$ and $\{e, f, g, h, i\}$. We call a CA-set A *minimal* if no proper subset B of A is a CA-set. Note that a minimal CA-set in a level-1 network is necessarily the leaf set of a cherry, a reticulated cherry or a cactus.

We now explain how the problem of finding minimal CA-sets in a network N can be translated into a graph theoretical problem given in terms of $\mathcal{T}(N)$. This has the advantage of allowing us to formulate an algorithm to deal with arbitrary dense trinet sets which uses standard graph theory algorithms.

To this end, given a dense collection $\mathcal{T}$ of trinets with leaves labelled by elements in a set X we associate the digraph $\Omega(\mathcal{T})$ which has vertex set

X and arc set consisting of those (x,y) such that there exists no taxon $z \in X - \{x,y\}$ for which $\{x,z\}$ is a CA-set for the trinet in $\mathcal{T}$ with leaf set $\{x,y,z\}$. For example, Fig. 10 depicts the digraph $\Omega(\mathcal{T}(N))$ for the trinet collection $\mathcal{T}(N)$ induced by the network N in Fig. 1(i).

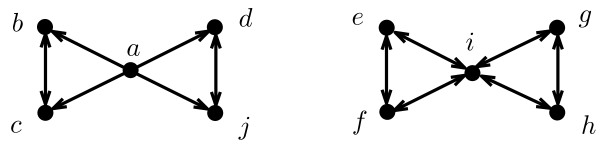

FIG. 10. An example of digraph Ω.

Now, recall that a subset A of the vertex set of a digraph is called a *sink set* if there exists no arc (u,v) in the digraph with u in A and v not in A. In addition, we call a sink set A in a digraph *small* if A is non-singleton and none of its proper non-singleton subsets is a sink set. For the network N depicted in Fig. 1(i), the minimal CA-sets are $\{b,c\}$, $\{d,j\}$ and $\{e,f,g,h,i\}$, which are exactly the same as the small sink sets in the digraph $\Omega(\mathcal{T}(N))$ in Fig. 10. This an illustration of the following result whose proof is given in the Supplementary Material.

Theorem A. *Suppose that N is a binary level-1 phylogenetic network on X with at least three leaves. If A is a proper subset of X, then the following assertions are equivalent:*

(i) *A is a minimal CA-set in N.*

(ii) *A is a small sink set in $\Omega(\mathcal{T}(N))$*

The TriLoNet algorithm

As with the Neighbor-Joining algorithm (Saitou and Nei, 1987) for inferring phylogenetic trees, our TriLoNet algorithm is based on a bottom up approach. Using Steps A1-A4 above if necessary, we shall assume that the input is a dense collection of trinets $\mathcal{T}$ on X. As outlined in the Introduction, our algorithm works by iteratively identifying cherries, reticulated cherries or cactuses. We now briefly present the algorithm in three steps, with a full description and complexity analysis included in the Supplementary Material.

Step B1: We begin by identifying a non-singleton subset Y of X that corresponds to a (possibly reticulated) cherry or cactus. To do this, for i ranging between 1 and $|X|-1$, we compute the smallest i for which the graph $\Omega_i(\mathcal{T})$ contains at least one arc, where $\Omega_i(\mathcal{T})$ has vertex set X and arc set consisting of those (x,y) such that there are less than i taxa $z \in X - \{x,y\}$ for which $\{x,z\}$ is a CA-set for the trinet in $\mathcal{T}$ with leaf set $\{x,y,z\}$. Note that $\Omega_1(\mathcal{T}) = \Omega(\mathcal{T})$, and so $\Omega_i(\mathcal{T})$ can be thought of an augmentation of $\Omega(\mathcal{T})$ which allows us to compute small sink sets even in case there are none to be found in $\Omega(\mathcal{T})$. The existence of a smallest index i follows since each arc in $\Omega_i(\mathcal{T})$ is also contained in $\Omega_{i+1}(\mathcal{T})$, and there exists an arc between each pair of vertices in $\Omega_{|X|-1}(\mathcal{T})$.

Now, to identify the subset Y of X, we simply compute a small sink set in $\Omega_i(\mathcal{T})$ for the smallest index i. This can be done in polynomial time by using Tarjan's algorithm (Tarjan, 1972) for computing the strongly connected components of a digraph. We give the full details in the Supplementary Material.

Step B2: We now associate a network N_Y that is either a (possibly reticulated) cherry or a cactus to the small sink set Y computed in Step 1. If Y contains two taxa, then N_Y is just the cherry or reticulated cherry that is displayed by the majority of trinets in $\mathcal{T}$. Otherwise, Y contains at least three taxa. Let $\mathcal{T}_Y$ be the subset of trinets whose leaf set is a subset of Y. Then we construct the cactus N_Y with leaf set Y as follows. First, the child of the reticulate vertex in N_Y is the taxon z in Y that maximises the number of type S_1 and S_2 trinets in $\mathcal{T}_Y$ which have z as the child of their reticulate vertices. Second, the split and relative ordering of taxa on the sides of the cactus is determined by considering the relative ordering of taxa determined by S_2 trinets in $\mathcal{T}_Y$. The details are given in the Supplementary Material.

Step B3: If the subset Y obtained in Step 1 is X itself, the algorithm stops and outputs the network N_Y. Otherwise (i) we compute the trinet set $\mathcal{T}^*$ induced by $\mathcal{T}$ on the set X^* formed by replacing every element in the subset Y with a new taxon y^*, (ii) obtain a level-1 network N^* for $\mathcal{T}^*$ recursively, and (iii) combine the two networks N_Y and N^* to form a level-1 network on X by replacing the taxon y^* in N^* with N_Y.

Consistency and Implementation

In the Supplementary Material we prove that the TriLoNet algorithm is consistent. More specifically, we prove:

Theorem B. *If the TriLoNet algorithm is applied to $\mathcal{T}(N)$ for a level-1 network N, then it will output N.*

This property was key in developing the TriLoNet algorithm as it guided the way in which we chose the selections given in Steps B1 and B2 of the algorithm. We have implemented the TriLoNet algorithm in JAVA and it is available for download at https://www.uea.ac.uk/computing/TriLoNet. It accepts three kinds of inputs: a NEXUS or FASTA file containing a sequence alignment, or a file specifying a dense set of trinets. The network constructed by the algorithm is outputted in the eNewick format (see, e.g. Cardona *et al.*, 2008) and/or the DOT format (Gansner *et al.*, 2006), which can be visualised by using Dendroscope (Huson and Scornavacca, 2012) and GraphViz (Ellson *et al.*, 2002), respectively. Although the complexity of TriLoNet is $O(|X|^4)$, it runs in reasonable time on fairly large data sets. For example, in Fig. 5 of the Supplementary Material we include the network inferred by TriLoNet for a data set consisting of 200 HIV sequences downloaded from http://www.hiv.lanl.gov/, which was computed in 7 hours 34 minutes on a MacBook Pro computer with an i7 processor and 16 GB RAM.

Supplementary Material

Supplementary text is available at Molecular Biology and Evolution online (http://www.mbe.oxfordjournals.org/).

Acknowledgments

We thank David Morrison for the website http://phylonetworks.blogspot.co.uk/p/datasets.html from which some data sets used for testing TriLoNet are downloaded. We would also like to thank one associate editor and two anonymous referees for their helpful and constructive comments on a previous version of this paper.

References

Aho, A. V., Sagiv, Y., Szymanski, T. G., and Ullman, J. D. 1981. Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions. *SIAM Journal on Computing*, 10(3): 405–421.

Bapteste, E., van Iersel, L., Janke, A., Kelchner, S., Kelk, S., McInerney, J. O., Morrison, D. A., Nakhleh, L., Steel, M., Stougie, L., *et al.* 2013. Networks: expanding evolutionary thinking. *Trends in Genetics*, 29(8): 439–441.

Bollyky, P. L., Rambaut, A., Harvey, P. H., and Holmes, E. C. 1996. Recombination between sequences of hepatitis B virus from different genotypes. *Journal of Molecular Evolution*, 42(2): 97–102.

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.

Cardona, G., Rosselló, F., and Valiente, G. 2008. Extended newick: it is time for a standard representation of phylogenetic networks. *BMC bioinformatics*, 9(1): 532.

Cooper, M. A., Adam, R. D., Worobey, M., and Sterling, C. R. 2007. Population genetics provides evidence for recombination in giardia. *Current Biology*, 17(22): 1984–1988.

Ellson, J., Gansner, E., Koutsofios, L., North, S. C., and Woodhull, G. 2002. Graphviz–open source graph drawing tools. In *Graph Drawing*, pages 483–484. Springer.

Gambette, P. and Huber, K. 2012. On encodings of phylogenetic networks of bounded level. *Journal of Mathematical Biology*, 65(1): 157–180.

Gansner, E., Koutsofios, E., and North, S. 2006. Drawing graphs with dot. Technical report, Technical report, AT&T Research. URL http://www. graphviz. org/Documentation/dotguide. pdf.

Grunewald, S., Spillner, A., Bastkowski, S., Bogershausen, A., and Moulton, V. 2013. SuperQ: computing supernetworks from quartets. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, 10(1): 151–160.

Gusfield, D. 2014. *ReCombinatorics: The Algorithmics of Ancestral Recombination Graphs and Explicit Phylogenetic Networks*. MIT Press.

Holland, B. R., Huber, K. T., Dress, A., and Moulton, V. 2002. δ plots: a tool for analyzing phylogenetic distance data. *Molecular Biology and Evolution*, 19(12): 2051–2059.

Huber, K. and Moulton, V. 2012. Encoding and constructing 1-nested phylogenetic networks with trinets. *Algorithmica*, 616: 714–738.

Huber, K., van Iersel, L., Kelk, S., and Suchecki, R. 2011. A practical algorithm for reconstructing level-1 phylogenetic networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 8(3): 635–649.

Huber, K. T., Van Iersel, L., Moulton, V., Scornavacca, C., and Wu, T. 2015. Reconstructing phylogenetic level-1 networks from nondense binet and trinet sets. *Algorithmica*, in press.

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

Huson, D. H. and Scornavacca, C. 2012. Dendroscope 3: an interactive tool for rooted phylogenetic trees and networks. *Systematic biology*, 61(6): 1061–1067.

Huson, D. H., Dezulian, T., Klopper, T., and Steel, M. A. 2004. Phylogenetic super-networks from partial trees. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 1(4): 151–158.

Huson, D. H., Richter, D. C., Rausch, C., Dezulian, T., Franz, M., and Rupp, R. 2007. Dendroscope: An interactive viewer for large phylogenetic trees. *BMC bioinformatics*, 8(1): 460.

Huson, D. H., Rupp, R., and Scornavacca, C. 2010. *Phylogenetic Networks: Concepts, Algorithms and Applications*. Cambridge University Press.

Jansson, J. and Sung, W.-K. 2006. Inferring a level-1 phylogenetic network from a dense set of rooted triplets. *Theoretical Computer Science*, 363(1): 60–68.

Jansson, J., Nguyen, N. B., and Sung, W.-K. 2006. Algorithms for combining rooted triplets into a galled phylogenetic network. *SIAM Journal on Computing*, 35(5): 1098–1121.

Lemey, P., Salemi, M., and Vandamme, A.-M. 2009. *The phylogenetic handbook: a practical approach to phylogenetic analysis and hypothesis testing*. Cambridge University Press.

Lott, M., Spillner, A., Huber, K. T., and Moulton, V. 2009. Padre: a package for analyzing and displaying reticulate evolution. *Bioinformatics*, 25(9): 1199–1200.

Nakhleh, L. 2011. Evolutionary phylogenetic networks: models and issues. In *Problem Solving Handbook in Computational Biology and Bioinformatics*, pages 125–158. Springer.

Nguyen, Q. and Roos, T. 2015. Likelihood-based inference of phylogenetic networks from sequence data by phylodag. In *Proc. 2nd International Conference on Algorithms for Computational Biology*, page in press.

Pardi, F. and Scornavacca, C. 2015. Reconstructible phylogenetic networks: Do not distinguish the indistinguishable. *PLoS Computational Biology*, 11(4): e1004135.

Poormohammadi, H., Eslahchi, C., and Tusserkani, R. 2014. Tripnet: A method for constructing rooted phylogenetic networks from rooted triplets. *PloS one*, 9(9): e106531.

Rambaut, A. and Grass, N. C. 1997. Seq-gen: an application for the monte carlo simulation of dna sequence evolution along phylogenetic trees. *Computer applications in the biosciences: CABIOS*, 13(3): 235–238.

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.

Schmidt, H. A., Strimmer, K., Vingron, M., and von Haeseler, A. 2002. Tree-puzzle: maximum likelihood phylogenetic analysis using quartets and parallel computing. *Bioinformatics*, 18(3): 502–504.

Semple, C., Daniel, P., Hordijk, W., Page, R. D., and Steel, M. 2004. Supertree algorithms for ancestral divergence dates and nested taxa. *Bioinformatics*, 20(15): 2355–2360.

Tarjan, R. 1972. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2): 146–160.

Than, C., Ruths, D., and Nakhleh, L. 2008. Phylonet: a software package for analyzing and reconstructing reticulate evolutionary relationships. *BMC bioinformatics*, 9(1): 322.

van Iersel, L. and Moulton, V. 2014. Trinets encode tree-child and level-2 phylogenetic networks. *Journal of Mathematical Biology*, 68: 1707–1729.

Wang, L., Zhang, K., and Zhang, L. 2001. Perfect phylogenetic networks with recombination. *Journal of Computational Biology*, 8(1): 69–78.

Woolley, S. M., Posada, D., and Crandall, K. A. 2008. A comparison of phylogenetic network methods using computer simulation. *PLoS One*, 3(4): e1913.

Yu, Y., Dong, J., Liu, K. J., and Nakhleh, L. 2014. Maximum likelihood inference of reticulate evolutionary histories. *Proceedings of the National Academy of Sciences*, 111(46): 16448–16453.