A fast and scalable method for inferring phylogenetic networks from trees by aligning lineage taxon strings

Louxin Zhang,[1] Niloufar Abhari,[2] Caroline Colijn,[2] and Yufeng Wu[3]

[1]*Department of Mathematics and Centre for Data Science and Machine Learning, National University of Singapore, Singapore 119076, Singapore;* [2]*Department of Mathematics, Simon Fraser University, Burnaby, British Columbia V5A 1S6, Canada;* [3]*Department of Computer Science and Engineering and Institute for Systems Genomics, University of Connecticut, Storrs, Connecticut 06269, USA*

The reconstruction of phylogenetic networks is an important but challenging problem in phylogenetics and genome evolution, as the space of phylogenetic networks is vast and cannot be sampled well. One approach to the problem is to solve the minimum phylogenetic network problem, in which phylogenetic trees are first inferred, and then the smallest phylogenetic network that displays all the trees is computed. The approach takes advantage of the fact that the theory of phylogenetic trees is mature, and there are excellent tools available for inferring phylogenetic trees from a large number of biomolecular sequences. A tree–child network is a phylogenetic network satisfying the condition that every nonleaf node has at least one child that is of indegree one. Here, we develop a new method that infers the minimum tree–child network by aligning lineage taxon strings in the phylogenetic trees. This algorithmic innovation enables us to get around the limitations of the existing programs for phylogenetic network inference. Our new program, named ALTS, is fast enough to infer a tree–child network with a large number of reticulations for a set of up to 50 phylogenetic trees with 50 taxa that have only trivial common clusters in about a quarter of an hour on average.

[Supplemental material is available for this article.]

In this study, phylogenetic networks over a set of taxa are rooted, directed acyclic graphs in which leaves represent the taxa, the nonleaf indegree-1 nodes represent speciation events and the nodes with multiple incoming edges represent reticulation events. The nonleaf indegree-1 nodes are called tree nodes; the other nonleaf nodes are called reticulate nodes. We assume that each tree node is of outdegree 2; each reticulate node and the network root is of outdegree 1 in a phylogenetic network (Fig. 1). Phylogenetic trees are just phylogenetic networks with no reticulate nodes and thus are binary. (Basic concepts and notation can be found in the Supplemental Methods.)

Now that a variety of genomic projects have been completed, reticulate evolutionary events (e.g., horizontal gene transfer, introgression, and hybridization) have been shown to play important roles in genome evolution (Koonin et al. 2001; Gogarten and Townsend 2005; Marcussen et al. 2014; Fontaine et al. 2015). Although phylogenetic networks are appealing for modeling reticulate events (Koblmüller et al. 2007), it is extremely challenging to apply phylogenetic networks in the study of genome evolution. One reason for this is that a computer program has yet to be made available for analyzing data as large as what current research is interested in (Wu 2020; Molloy et al. 2021), although recently, Bayesian methods have been used to reconstruct reassortment networks, which describe patterns of ancestry in which lineages may have different parts of their genomes inherited from distinct parents (Müller et al. 2020, 2022).

Here, we focus on computing phylogenetic networks that display a given set of gene trees (Wu 2010; Albrecht et al. 2012; Whidden et al. 2013; Elworth et al. 2019; van Iersel et al. 2022). In this approach, trees are first inferred from biomolecular sequences and then used to reconstruct a phylogenetic network with the smallest hybridization number (HN) that displays all the trees (see Elworth et al. 2019), where the HN is defined as the sum over all the reticulate nodes of the indegree of each reticulate node minus 1. This approach takes advantage of the fact that the theory of phylogenetic trees is mature, and there are excellent tools available for inferring trees from a large number of sequences. It has been used in evolutionary studies (Koblmüller et al. 2007; Marcussen et al. 2014).

Although this parsimonious approach is faster than the maximum likelihood approach (Lutteropp et al. 2022), the parsimonious network inference problem is still NP-hard even for the special case when there are only two input trees (Bordewich and Semple 2007). For the two-tree case, the fastest programs include MCTS-CHN (Yamada et al. 2020) and HYBRIDIZATION NUMBER (Whidden et al. 2013). For the general case in which there are multiple input trees, HYBROSCALE (Albrecht 2015) and its predecessor (Albrecht et al. 2012), PRIN (Wu 2010), and PRINs (Mirzaei and Wu 2015) have been developed. All of these methods reconstruct a tree–child network with the smallest HN. Some of the methods insert reticulate edges or use other editing operations to search a network in the network space. Others reduce the tree–child network reconstruction problem to finding maximum acyclic agreement forests for the set input trees. Finally, some methods combine both of these techniques. Unfortunately, none of them will work for inferring a network from more than 30 trees if the trees have 30 or more taxa and do not have any nontrivial taxon clusters in common, where a nontrivial taxon cluster of a tree consists of all taxa below a tree node that is neither a leaf nor the root.

Corresponding author: matzlx@nus.edu.sg
Article published online before print. Article, supplemental material, and publication date are at https://www.genome.org/cgi/doi/10.1101/gr.277669.123. Freely available online through the *Genome Research* Open Access option.

© 2023 Zhang et al. This article, published in *Genome Research*, is available under a Creative Commons License (Attribution-NonCommercial 4.0 International), as described at http://creativecommons.org/licenses/by-nc/4.0/.

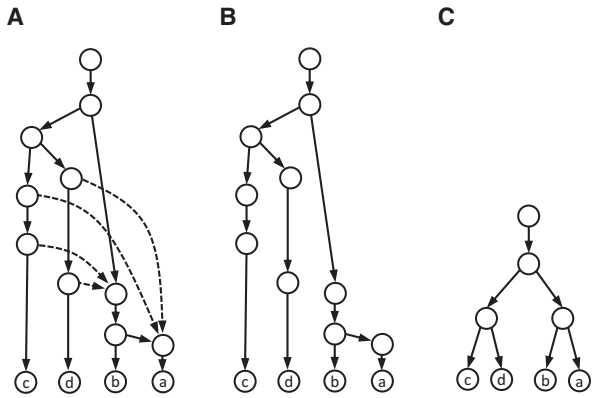


Figure 1. The display of a phylogenetic tree in a tree–child network. (*A*) A tree–child network with two reticulate nodes on the taxa (*a* to *d*). (*B*) A subtree that was obtained by the removal of the dashed incoming edges of the reticulate nodes in the network. (*C*) A tree displayed in the network, which was obtained from the subtree in *B* by removing all degree-2 nodes through combining their unique incoming and outgoing edges into an edge.

Because the network space is vast and cannot be fully sampled, attention has been switched to the inference of tree–child networks (Cardona et al. 2009), in which every nonleaf node has at least one child that is not reticulate, or, recently, a tree-based network (Pickrell and Pritchard 2012). Tree–child network is a superclass of phylogenetic trees with a completeness property that for any set of phylogenetic trees, there exists always a tree–child network that displays all the trees (Linz and Semple 2019). Other desired properties of tree–child networks include the fact that all the tree–child networks are efficiently enumerated (Zhang 2019a,b; Cardona and Zhang 2020).

Results

We mainly report a scalable computer program for inferring tree–child networks from multiple gene trees. The program ALTS takes a different approach that reduces the network inference problem to aligning the lineage taxon strings (LTSs) computed from the input trees with respect to (w.r.t.) an ordering on the taxa.

The inference algorithm

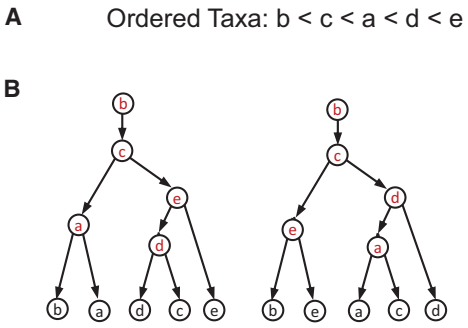
Consider a set X of taxa. Let T be a binary phylogenetic tree on X, and let N be a tree–child network on X. N displays T if T can be obtained from N by (1) removing all but one incoming edge for each reticulation node (Fig. 1A,B) and then (2) deleting all degree-2 nodes (which were reticulation nodes in N) (Fig. 1C).

The inference algorithm we introduce here will check all possible orderings on the taxon set to obtain the tree–child networks with the smallest HN (and equivalently the smallest number of nodes). Let X be a taxon set such that $|X|=n$, and let $\pi=\pi_1\pi_2\cdots\pi_n$, representing a (total) ordering of X, by which π_i is "less than" π_{i+1} for each $i<n$. For any nonempty subset X' of X, we use $\min_\pi(X')$ and $\max_\pi(X')$ to denote the minimum and maximum taxon of X' w.r.t. π, respectively.

Because the root of T is of outdegree 1, T has n nonleaf nodes, called internal nodes. We label the n internal nodes of T one-to-one with the taxa w.r.t. π by assigning the smallest taxon to the degree-1 root and assigning $\max_\pi\{t_v, t_w\}$ to an internal node with children v and w, where t_v is the smallest taxon below v (Labeling) (Supplemental Methods). For instance, let $X=\{a, b, c, d, e\}$ and $\pi=bcade$ (Fig. 2A). The two trees on X in Figure 2B have their internal nodes labeled w.r.t. πusing Labeling.

Let τ be a specific taxon of X such that $\tau\neq\pi_1$. We consider the unique path from the root ρ to the leaf ℓ that represents τ in T: $u_0=\rho, u_1, u_2, \cdots, u_k=\ell$. Then, $\min_\pi(C(u_k))=\tau$, whereas $\min_\pi(C(u_0))=\min_\pi(C(u_1))=\pi_1<_\pi\tau$. Because $\min_\pi(C(u_k))\leq_\pi \min_\pi(C(u_{k+1}))$, there is a unique index j such that $1\leq j<k$ and $\min_\pi(C(u_j))<\tau=\min_\pi(C(u_{j+1}))$. This implies that u_j was labeled with τ by applying Labeling and that no other internal node got the same label. The sequence consisting of the labels of $u_{j+1}, u_{j+2}, \cdots, u_{k-1}$ is called *the lineage taxon string* (LTS) of τ. The LTSs computed in the trees given in Figure 2B are listed in Figure 2C.

Conversely, for the LTS of each taxon τ, we construct a directed path whose nodes are labeled one-to-one with the taxa of the LTS, and add a leaf labeled with τ below the path. After we connect



C The lineage taxon strings

Taxon	Left tree	Right tree
b	c, a	c, e
c	e, d	d, a
a	empty	empty
d	empty	empty
e	empty	empty

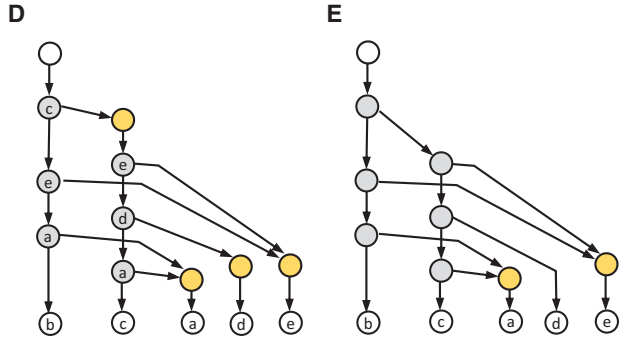


Figure 2. The construction of a tree–child network that displays two phylogenetic trees. (*A*) An ordering on {*a, b, c, d, e*}. (*B*) Two trees, where the internal nodes are labeled w.r.t. the ordering using the Labeling algorithm. (*C*) The lineage taxon strings (LTSs) of the taxa obtained from the labeling in *B*. (*D*) The rooted directed graph constructed from the shortest common supersequences (SCS) of the LTSs of the taxa (in C) using Tree–child Network Reconstruction. The SCS is [*c, e, a*] for [*c, a*] and [*c, e*] and is [*e, d, a*] for [*e, d*] and [*d, a*]. (*E*) The tree–child network obtained after the removal of the degree-2 nodes.

the first node of the resulting path ending with each taxon other than π_1 to all the nodes labeled with the taxon in other paths, we obtain T. Thus, the LTSs obtained from T under any ordering π on X can be used to recover uniquely T.

A string s is said to be a common supersequence of multiple strings if all the strings can be obtained from s by erasing zero or more symbols. Let $\{T_1, T_2, \cdots, T_k\}$ be a set of k trees on X. Let α_{ji} be the LTS of π_i in T_j for each i from 1 to n. (Note that α_{jn} is the empty string for each j.) Assume that, for each i, β_i is a common supersequence of all $\alpha_{1i}, \alpha_{2i}, \cdots, \alpha_{ki}$ on X. We can construct a tree–child network $N_\pi(\beta_1, \beta_2, \cdots, \beta_{n-1})$ on X using the TREE–CHILD NETWORK CONSTRUCTION algorithm given below.

Tree–Child Network Construction

1. (**Vertical edges**) For each β_i, define a path P_i with $|\beta_i|+2$ nodes:
   $h_i, v_{i1}, v_{i2}, \cdots, v_{i|\beta_i|}, \ell_{\pi_i}$,
   where β_n is the empty sequence.
2. (**Left–right edges**) Arrange the n paths from left to right as $P_1, P_2, \cdots, P_n$.
   If the mth symbol of β_i is π_j, we add an edge (v_{im}, h_j) for each i and each m.
3. For each $i>1$, if h_i is of indegree 1, eliminate h_i by removing h_i, together with its incoming and outgoing edge, and adding a new edge from its parent to its child.

The algorithm is illustrated in Figure 2, D and E, where the SCSs are $[c, e, a]$ and $[e, d, a]$ for $\pi_1=b$ and $\pi_2=c$, and the empty sequence for $\pi_3=a$ and $\pi_4=d$.

The network output from TREE–CHILD NETWORK CONSTRUCTION is always a tree–child network (Proposition 2) (Supplemental Methods). Combining LABELING and TREE–CHILD NETWORK CONSTRUCTION, we obtain the following exact algorithm for the network inference problem, for which the correctness is proved in Section A of the Supplemental Methods.

Algorithm A

Input: K trees $T_1, T_2, \cdots, T_k$ on X, $|X|=n$.
0. Set $M=\infty$ and define $n-1$ string variables $S_1, S_2, \cdots, S_{n-1}$;
1. For each ordering $\pi=\pi_1\pi_2\cdots\pi_n$ on X:
   1.1. Call LABELING to label the internal nodes in each T_i;
   1.2. For each taxon π_j, compute its LTS s_{ij} in each T_i;
   1.3. Compute the SCS s_j of $s_{1j}, s_{2j}, \cdots, s_{kj}$ for each $j<n$;
   1.4. If $M > \sum_{j=1}^{n-1}|s_j|$, update M to the length sum; update S_j to s_j for each j;
2. Call TREE–CHILD NETWORK CONSTRUCTION to compute a tree–child network from the strings $S_1, S_2, \cdots, S_{n-1}$.

A scalable version

Because there are $n!$ possible orderings on n taxa and 15! is already too large, ALGORITHM A is not fast enough for a set of multiple trees on 15 or more taxa. Another obstacle to scalability is computing the SCS for the LTS of each taxon. We achieved high scalability by using an ordering sampling method and a progressive approach for the SCS problem.

First, the ordering sampling starts with an arbitrary ordering on the taxa and finishes in $\lfloor n/2 \rfloor$ iterative steps. Assume that Π_m is the set of orderings obtained in the mth step ($m \geq 1$) such that $|\Pi_m| \leq H$ for a parameter H predefined to bound the running time. In the $(m+1)$ step, for each ordering $\pi=\pi_1\pi_2\cdots\pi_n \in \Pi_m$, we generate $(n-2m+1)(n-2m)$ new orderings by interchanging π_{2m-1} with π_i and interchanging π_{2m} with π_j for every possible i and j such that $i \neq j$, $i>2m$ and $j>2m$. For each new ordering $\pi' = \pi'_1\pi'_2\cdots\pi'_n$, we compute a SCS s_i of the LTSs of taxon π'_i in the input trees for each $i \leq 2m$. We compute Π_{m+1} by sampling, at most, H new orderings that have the smallest length sum $\sum_{1\leq i\leq 2m}|s_i|$.

Second, different progressive approaches can be used to compute a short common supersequence for LTSs in each sampling step (Fraser 1995). We use the following approach:

A common supersequence of n strings is computed in $n-1$ iterative steps. In each step, a pair of strings s_i and s_j such that the SCS of s_i and s_j, SCS(s_i, s_j), has the minimum length, over all possible string pairs, is selected and replaced with SCS(s_i, s_j).

Although the above algorithm had good performance for our purpose according to our test, it cannot always output the shortest solution for all possible instances. The reason is that finding the SCS for arbitrary strings is NP-hard in general (Garey and Johnson 1979), and our algorithm is as a linear-time algorithm unlikely to be the exact algorithm.

After the sampling process finishes, we obtain a set $\Pi_{\lfloor n/2 \rfloor}$ of good ordering; for each ordering, we obtain a short common supersequence of the LTSs of a taxon obtained from the input trees. To further improve the tree–child network solution, we also use the dynamic programming algorithm to recalculate a short common supersequence for the LTSs of each taxon, subject to the 1 GB memory usage limit. We then use whichever is shorter to compute a tree–child network.

Implementation of the algorithm

Another technique for improving the scalability is to decompose the input tree set into irreducible sets of trees if the input trees are reducible (Wu 2010; Albrecht et al. 2012) (Sec. B, Supplemental Methods). Here, a set of trees are reducible if there is at least one common cluster except the singletons and the whole taxon set.

Our program is named ALTS, an acronym for "Aligning Lineage Taxon Strings." It can be downloaded from the GitHub site (see Software availability). We also developed a program that assigns a weight to each edge of the obtained tree–child network if the input trees are weighted (Sec. C, Supplemental Methods).

In summary, the process of reconstructing a parsimonious tree–child network involves the following steps. First, decompose the input tree set S into irreducible tree sets, say $S_1, S_2, \cdots, S_t$. Second, infer a set N_i of tree–child networks for each S_i. Third, assemble the tree–child networks in $N_1, N_2, \cdots, N_t$ to obtain the networks that display all the trees in S. Fourth, if the input trees are weighted, the branch weights are estimated for the output tree–child networks.

Validation experiments

We assessed the accuracy and scalability of ALTS on a collection of simulated data sets that were generated using an approach reported in Wu (2010) (see Methods section).

The optimality evaluation

We compared ALTS with two heuristic network inference programs: PRINs (Mirzaei and Wu 2015), which infers an arbitrary phylogenetic network, and van Iersel et al.'s method (van Iersel

et al. 2022), which infers a tree–child network. We first ran the three methods on 50 sets of trees on 20 and 30 taxa, each containing 10 trees. Van Iersel et al.'s program is a parallel program. It could run successfully only on 44 (out of 50) tree sets in the 20-taxon case and 27 (out of 50) tree sets in the 30-taxon case. It was aborted for the remaining data sets after 24 h of clock time (or ~1000 CPU hours) had elapsed.

ALTS outputs tree–child networks with the same HN as van Iersel et al.'s method on all but three data sets, where the latter ran successfully. The HN of the tree–child networks inferred with ALTS was one more than that inferred with the latter on two 20-taxon 10-tree data sets and three more than the latter on one 30-taxon 10-tree data set. Moreover, Van Iersel et al.'s method only outputted a tree–child network, whereas ALTS computed multiple tree–child networks with the same HN.

PRINs ran successfully on 49 out of 50 data sets in the 20-taxon case. In theory, the HN is inherently equal to or less than the HN of the optimal tree–child networks for every tree set. In the 20-taxon 10-tree case, the tree–child HN inferred with ALTS was equal to that inferred with PRINs on 20 data sets. The 29 discrepancy cases are summarized in the first row of Table 1. In the 30-taxon case, the HN difference of the two programs was also at most four (Table 1, row 2). The tree–child HN inferred by ALTS was even one less than the HN inferred by PRINs on one data set.

In summary, ALTS is almost as accurate as van Iersel et al.'s method in terms of minimizing network HN. The comparison between ALTS and PRINs indicated that the tree–child HN is rather close to the HN for multiple trees when the number of taxa is not too big.

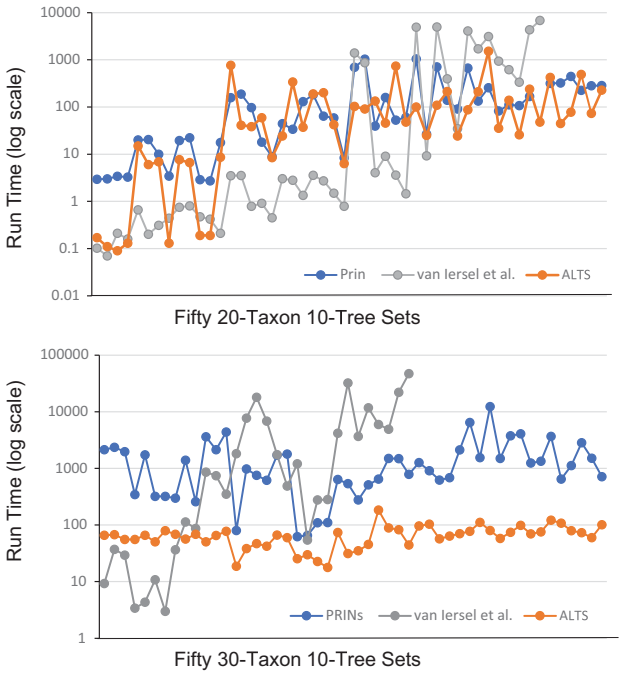


Figure 3. Run time (in seconds) of the three methods on 100 data sets. Each data set contains 10 trees on 20 or 30 taxa. The data sets are sorted in the increasing order according to the HN output from PRINs. Van Iersel et al.'s method had some missing data points because of an abort after 24 h of clock time.

The scalability evaluation

The wall-clock times of the three methods on 100 data sets, each having 10 trees on 20 or 30 taxa, are summarized in Figure 3. In the 20-taxa 10-tree case, the HN inferred by PRINs ranged from five to 17. ALTS finished in 0.09 sec to 25 min 14 sec (with the mean being 2 min 21 sec). On the 49 (out of 50) 20-taxa 10-tree data sets on which PRINs finished, it took 2.94 sec to 17 min 19 sec (with the mean being 2 min 58 sec). ALTS was faster than PRINs on 35 tree sets. On average, PRINs and ALTS were comparable in time for this case.

On the 44 20-taxa 10-tree data sets on which van Iersel et al.'s method finished, its run time ranged from 0.07 sec to 82 min 22 sec (with the mean being 13 min 3 sec). Van Iersel et al.'s method ran faster than ALTS on 26 data sets in which the HN inferred by PRINs was less than 11. One reason for this is probably that the former is a parallel program. However, ALTS was faster than van Iersel et al.'s method on the remaining 18 tree sets in which the HN inferred by PRINs was 12 or more.

In the 30-taxon case, the HN of the solution from PRINs ranged from eight to 21. As shown in Figure 3, ALTS was faster than PRINs on every data set. Van Iersel et al.'s method finished on 31 (out of 50) data sets, for which the HN of the solution obtained with PRINs was 15 or more. ALTS was faster than Van Iersel et al.'s method on 23 data sets, whereas Van Iersel et al.'s method was faster than ALTS on the remaining eight data sets. On average, in the 30-taxon case, ALTS was 24 and 53 times faster than PRINs and the van Iersel et al.'s method, respectively.

Lastly, we further ran ALTS on 100 data sets, each containing 50 trees on 40 or 50 taxa. PRINs finished on twenty-eight 40-taxon 50-tree data sets and five 50-taxon 50-tree data sets. In the 40-taxon 50-tree case, ALTS finished in 3 sec to 31 min 52 sec (with the mean being 7 min 14 sec). In contrast, PRINs finished on 28 tree sets, taking 3 min 19 sec to 15 h 34 min 52 sec (with the mean being 3 h 49 min 46 sec) (Fig. 4).

In the 50-taxon 50-tree case, ALTS finished in 2 sec to 45 min 12 sec (with the mean being 9 min 24 sec) (Fig. 4). In contrast, van Iersel et al's method could not finish on any irreducible set of 50 trees on 50 taxa. PRINs finished on five tree sets in 2 h 25 min on average (Fig. 4).

Taken together, these results suggest that ALTS has high scalability and is fast enough to infer tree–child networks for large tree sets.

Table 1. Summary of the HN discrepancy between ALTS and PRINs in 20-taxon and 30-taxon data sets each containing 10 trees

	HN_{ALTS} minus HN_{PRINs}					
Data type	**−1**	**0**	**1**	**2**	**3**	**4**
20-taxon trees		20	11	9	6	3
30-taxon trees	1	5	13	14	16	1

The accuracy evaluation

Evaluating the accuracy of ALTS (and the other two methods) is not straightforward. The random networks that were used to generate the tree sets used in the last two subsections are not tree–child networks and contain frequently a large number of deep reticulation events. On the other hand, by the principle of parsimony, the networks inferred by the three programs contain far fewer numbers of reticulation events. As such, we assessed the accuracy of ALTS by using a Jaccard score that measures the symmetric

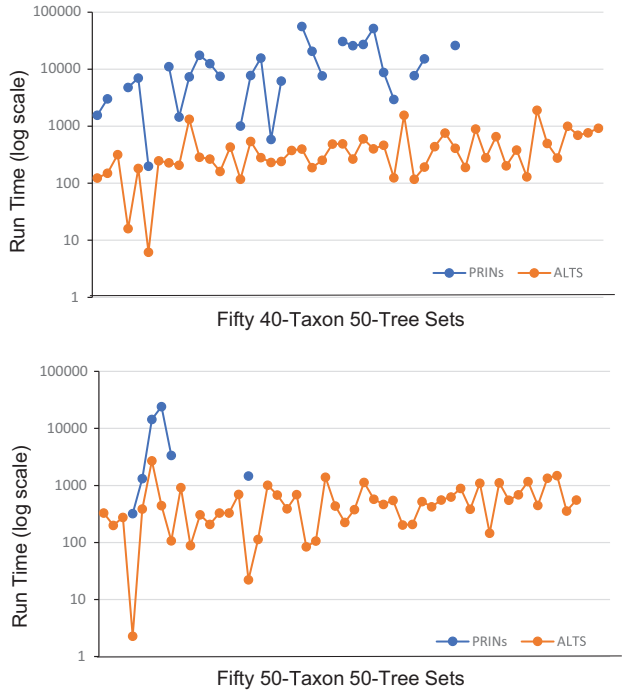


Figure 4. The run time (in seconds) of ALTS and PRINs on 100 data sets. Each data set contains 50 trees on 40 or 50 taxa. The data sets are sorted in the increasing order according to the HN of the tree–child networks inferred by ALTS. PRINs had some missing data points because of an abort after 24 h of clock time.

difference between the set of clusters in the original networks and in the network inferred by ALTS (see Methods) (Huson et al. 2010).

We considered two simulated networks containing 16 binary reticulations (Network 1) (Supplemental Fig. S1) and 19 binary reticulations (Network 2) (Supplemental Fig. S2). The two networks were produced using the same simulation program as used for the optimality evaluation but with a lower ratio of reticulation events. We also examined simplified versions of the two networks that were obtained by merging a reticulate node and its child if the reticulate node has a unique child and the child is also a reticulation node. The two simplified networks have nine and 10 reticulation events, respectively (Supplemental Figs. S1, S2, bottom). For each network and each $k = 20$, 30, 40, 50, we generated 10 k-tree sets. For each tree set, we inferred a network using ALTS and computed the Jaccard score for it and the original network. The dissimilarity analyses are summarized in Figure 5.

Network 1 (and its simplified version) contains fewer reticulation events than Network 2. We had slightly better reconstruction accuracy for Network 1 than Network 2 (mean Jaccard score range [0.3, 0.45] vs. [0.55, 0.65]) (Fig. 5). Also, the reconstruction from the trees sampled from each network was not significantly better than that from its simplified version. Given that all four networks can contain as many as 2^{17} trees, the results suggest that 50 trees are far fewer than enough for accurate reconstruction of both non-binary networks.

On the other hand, ALTS performed well for inferring a binary tree–child network with 13 binary reticulation nodes on 22 taxa. We sampled trees from the binary tree–child network given in Supplemental Figure S3. We could reconstruct the network on one out of 10 random five-tree sets, six out of 10 random 10-tree sets, and all 10 random 20-tree sets.

Last, we also examined the accuracy of reconstructing a network from the trees inferred from DNA sequence data using the following setting (for details, see Methods):

Generate randomly a network.
Sample a gene tree with branch lengths in the network.
Simulate DNA evolution to obtain a sequence of 1000 bp on the gene tree.
Infer a maximal likelihood tree from the simulated sequence.

On each random network, we sampled 2000 "true" gene trees and inferred 2000 trees accordingly.

We examined two networks (Network 3 and Network 4 hereafter) (Supplemental Fig. S4) on 30 taxa that contain five and six binary reticulation events, respectively. Because the inferred trees were noise, we used five-tree data sets for testing. Inference with more than five inferred trees had low accuracy, whereas inference with more than five true gene trees had high accuracy. We ran ALTS on 50 random tree sets for each of the three cases. In the first case, a data set consists of five inferred trees. In the second case, a data set consists of five "consensus" inferred trees that appeared six or more times in the list of inferred trees. Note that a consensus inferred tree is much more likely a true gene tree than a tree that was only inferred once in our experiment. In the third case, a data set consists of five "true" gene trees.

The results are summarized in Figure 6. For Network 3, the average Jaccard score for each inference test was 0.161, 0.079, and 0.043 in Case 1, 2, and 3, respectively. In addition, ALTS reconstructed Network 3 correctly on 27 out of 50 tree sets in Case 3. The performance of ALTS is similar for the testing on Network 4. These results suggest that accurate inference of gene trees from sequence data is vital for network inference with ALTS.

A phylogenetic network for 13 wheat-related grass species

We also validated our tool by inferring phylogenetic relationships for a set of wheat-related grass species. Bread wheat (*Triticum aestivum*) is a hexaploid species (genome AABBDD) formed through two rounds of hybridization between three diploid progenitors (i.e., *Triticum urartu* of the A subgenome, an unknown species of the B subgenome,

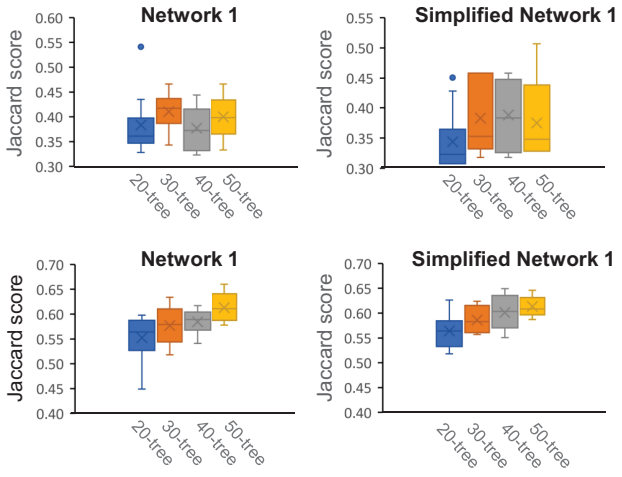


Figure 5. The box and whisker plots for the Jaccard scores for the original networks (Supplemental Figs. S1, S2) and ones inferred by ALTS in four cases. In each plot, the four bars from *left* to *right* summarize the Jaccard scores for the original network and 10 networks inferred from 20-, 30-, 40-, and 50-tree sets, respectively.

and *Aegilops tauschii* of the D subgenome) (Marcussen et al. 2014; Levy and Feldman 2022). A recent comprehensive genomic study of Glémin et al. (2019) suggests that hybridizations were pervasive in the evolution of *T. urartu*, *Se. tauschii*, and 11 other grass species. Using a hypothesis testing approach, they detected six reliable and two possible reticulated events. Here, to eliminate the effect of incomplete lineage sorting (ILS) and tree inference errors, we simplified the 247 gene trees reported in their paper and selected 33 of them for network inference (see Methods section). Using ALTS, we obtained the phylogenetic network depicted in Figure 7. The network displays 73 out of 247 simplified gene trees and contains on average 79% nontrivial node clusters of the remaining trees.

The network model contains two binary reticulate events and four clusters of reticulate events. Events 2 and 4 and event cluster 6 are consistent with the findings reported by Glémin et al. (2019). In particular, Event 2 is the hybridization between A and B lineages that formed the D-subgenome clade (Fig. 7, middle; Marcussen et al. 2014). The gene flows from an ancestor of *Aegilops speltoides* in the cluster 6 reveals that the Sitopsis species are closer to *A. speltoides* than to *Ae. tauschii*, consistent to the cytogenetic analyses reported by Kihara (1954). Our model also suggests that complex reticulate events (Cluster 5) occurred between *Ae. tauschii* and the ancestors of *Aegilops caudata* and *Aegilops umbellulata*. The complex gene flows in the event cluster 5 has not been reported in literature, but it is compatible with a chloroplast capture model (Fig. 2B; Li et al. 2015). Conversely, the two possible reticulate events reported by Glémin et al. (2019) are not supported by our model. Further verification of these inconsistent interspecific reticulate events may need gene order information on the related genomes and additional genomes of D-genome clade.

Discussion

We have presented ALTS. It is based on an algorithmic innovation that reduces the minimum tree–child network problem to computing the SCS of the LTSs of the taxa, obtained from the input trees w.r.t. a predefined ordering on the taxa. ALTS is fast enough to infer a parsimonious tree–child network for a set of 50 trees on 50 taxa in a quarter of an hour on average even if the input trees do not have any nontrivial taxon clusters in common. Another contribution is an algorithm for assigning weights to the edges of the reconstructed tree–child network if the input trees are weighted. Our work makes network reconstruction more feasible in the study of evolution and phylogenomics.

The accuracy analyses suggest that 50 trees are likely not enough for accurately inferring a phylogenetic network model that has 10 or more reticulation events. Therefore, a program that can process more than 100 trees is definitely wanted. We remark that ALTS can be made even more scalable by distributing the computing tasks for taxon orderings into a number of processors using distributed computing programming. This is because the computing tasks for different orderings are independent from each other.

Phylogenetic relationships obtained for the 13 wheat-related grass species and for *Hominin* (Sec. D, Supplemental Methods) provide an illustration of good performance of ALTS on empirical data. The analyses show that how to eliminate the effects of ILS is important for inference of phylogenetic networks. We will further investigate how to improve the accuracy of ALTS by incorporating the genomic sequences of the taxa and a process of removing ILS events into network inference.

Methods

Method for generating random tree data sets

The simulated tree data sets were generated using an approach appearing in work by Wu (2010). For each $k \in \{20, 30, 40, 50\}$, a phylogenetic network on k taxa was first generated by simulating speciation and reticulation events backward in time with the weight ratio of reticulation to speciation being set to 3:1. Fifty trees displayed in the networks were then randomly sampled. This process was repeated to generate 2500 trees for each k.

For assessing the accuracy of ALTS for inferring phylogenetic networks from genomic sequences, we generated gene trees with branch lengths from a network by calling a coalescent simulation program named ms (Hudson 2002). We ordered the speciation and reticulation events in the input network and set the time difference between adjacent evolutionary events to 10 coalescent units. Here, a relatively long coalescent time between adjacent evolutionary events was used to reduce the effect of ILS in the simulated gene trees.

Methods for gene sequence simulation and gene tree inference

We used the Seq-Gen program (Rimbaud and Grass 1997) with the GTR substitution model to generate DNA sequences of 1000 bp on

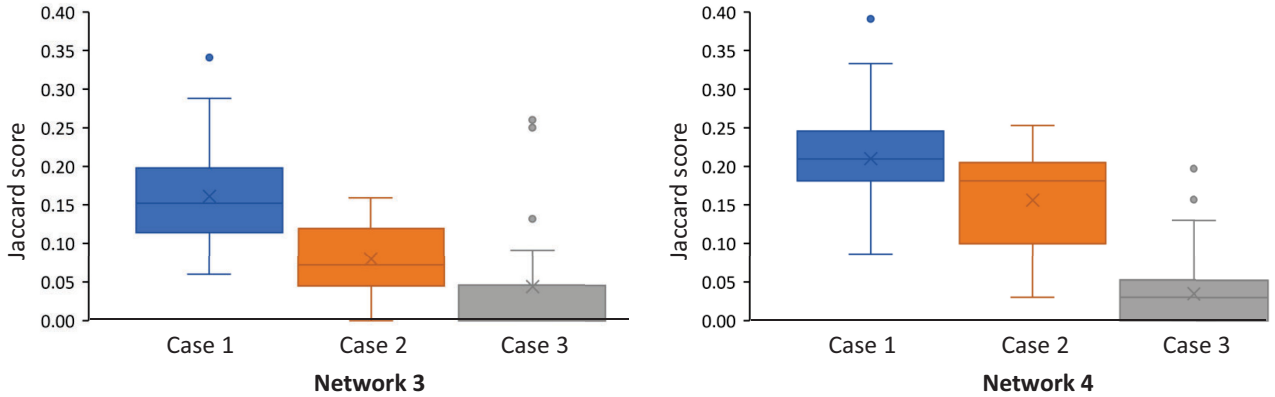


Figure 6. The box and whisker plots for the Jaccard scores for the original networks (Supplemental Fig. S4) and ones inferred using ALTS in three cases. In each plot, the three bars from *left* to *right* summarize the 50 Jaccard scores obtained using five random inferred trees (Case 1), using five random inferred trees that appeared six or more times in the list of all 2000 inferred trees (Case 2), and using the "true" gene trees (Case 3) sampled from the networks.

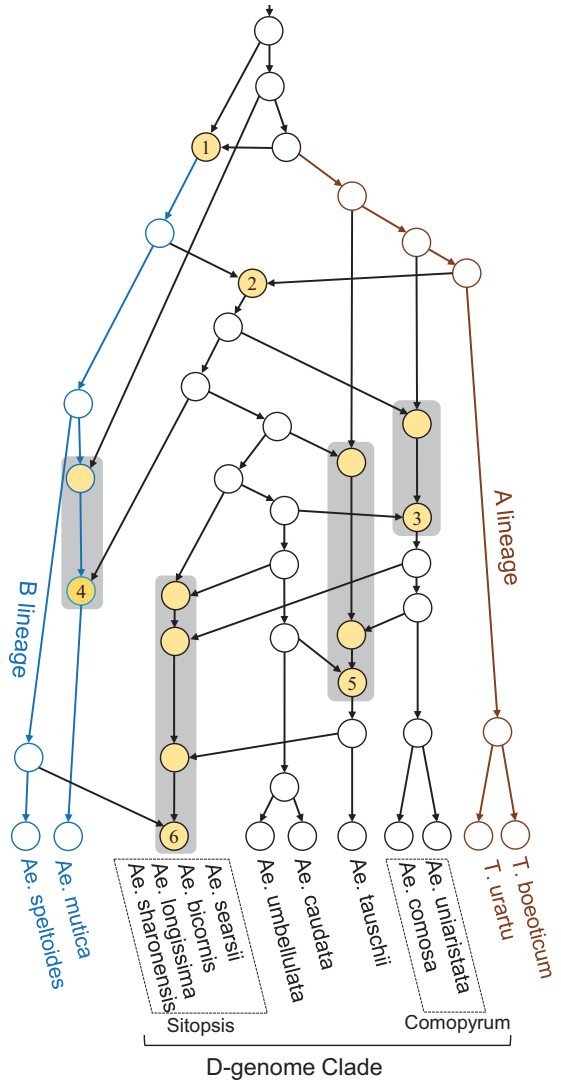


Figure 7. A phylogenetic network for 13 wheat-related grass species inferred using ALTS. The reticulation nodes are colored in yellow. The model contains two binary reticulate events (1 and 2) and four event clusters (3 to 6).

a gene tree, where the scaling factor was set to 0.001 in order to convert coalescent units to the mutational units for Seq-Gen. Conversely, we used the RAxML program (Stamatakis 2014) with the GTR model to infer a gene tree from the simulated DNA sequence of 1000 bp. We used an outgroup to root the gene trees inferred by RAxML.

Jaccard score between two phylogenetic networks

We measured the dissimilarity between two phylogenetic networks by considering the symmetric difference of the set of taxa clusters in the networks (Huson et al. 2010). Here, a cluster in a network consists of all taxa below a node in that network. Precisely, for two phylogenetic networks N_1 and N_2 over X, we use $C(N_i)$ to denote the multiset of clusters appearing in N_i for $i=1$, 2, and define the Jaccard score between N_1 and N_2 as $s(N_1, N_2) = 1 - |C(N_1) \cap C(N_2)|/\ |C(N_1) \cup C(N_2)|$.

Tree data preprocessing for wheat-related grass species

Two hundred forty-seven distinct gene trees for 13 wheat-related grass species and four outgroup species were downloaded from the evolutionary study of Glémin et al. (2019). (These trees were inferred from orthologous genes in 47 individual genomes by using RAxML v8.) To infer interspecific reticulate events, we simplified the gene trees by using only one individual sequence for each species and removing all four outgroup sequences, resulting in 227 distinct trees with 13 leaves. To reduce the effect of ILS and gene tree inferring errors, we further selected 33 gene trees for which either of the following two conditions is true: (1) it was inferred on two genes, and (2) every node cluster of it appears in t (=20) or more gene trees. We used the ratio of the number of trees displayed in a network to its HN to measure its expression capacity. The percentage used in the condition (2) was chosen to control the trade-off between the size and expression capacity of the network model. For $t>18$, the inferred networks had a high HN. For $t>22$, the inferred network displayed a low number of gene trees. For $t=18$, 19, 20, 21, 22, the HN of the inferred network was 17, 13, 12, 12, and 12, whereas the network displayed 90, 71, 71, 71, and 63 gene trees, respectively. Because $90/17<71/13<71/12$ and $63/12<71/12$, we selected 20 as the filtering condition, resulting 33 gene trees.

Software availability

The C source code of ALTS can be found as Supplemental Code and at GitHub (https://github.com/LX-Zhang/AAST).

Competing interest statement

The authors declare no competing interests.

Acknowledgments

We thank Cedric Chauve and Aniket Mane for discussion in the beginning of this project. We also thank the anonymous reviewers for constructive comments on the earlier versions of our manuscript submitted to RECOMB'2023 and *Genome Research*. L.Z. was partly supported by Singapore Ministry of Education E Tier 1 grant R-146-000-318-114. Y.W. was partly supported by U.S. National Science Foundation grants CCF-1718093 and IIS-1909425.

References

Albrecht B. 2015. Computing all hybridization networks for multiple binary phylogenetic input trees. *BMC Bioinformatics* **16:** 236. doi:10.1186/s12859-015-0660-7

Albrecht B, Scornavacca C, Cenci A, Huson DN. 2012. Fast computation of minimum hybridization networks. *Bioinformatics* **28:** 191–197. doi:10.1093/bioinformatics/btr618

Bordewich M, Semple C. 2007. Computing the minimum number of hybridization events for a consistent evolutionary history. *Discrete Applied Math* **155:** 914–928. doi:10.1016/j.dam.2006.08.008

Cardona G, Zhang L. 2020. Counting and enumerating tree-child networks and their subclasses. *J Computer Syst Sci* **114:** 84–104. doi:10.1016/j.jcss.2020.06.001

Cardona G, Rosselló F, Valiente G. 2009. Comparison of tree-child phylogenetic networks. *IEEE-ACM Trans Comput Biol Bioinform* **6:** 552–569. doi:10.1109/TCBB.2007.70270

Elworth RL, Ogilvie HA, Zhu J, Nakhleh L. 2019. Advances in computational methods for phylogenetic networks in the presence of hybridization. In *Bioinformatics and phylogenetics* (ed. Warnow T), pp. 317–360. Springer, New York.

Fontaine MC, Pease JB, Steele A, Waterhouse RM, Neafsey DE, Sharakhov IV, Jiang X, Hall AB, Catteruccia F, Kakani E, et al. 2015. Extensive introgression in a malaria vector species complex revealed by phylogenomics. *Science* **347:** 1258524. doi:10.1126/science.1258524

Fraser CB. 1995. "Subsequences and supersequences of strings." PhD thesis, University of Glasgow, United Kingdom.
Garey MR, Johnson DS. 1979. *Computers and intractability: a guide to the theory of NP-completeness*. WH Freeman and Company, San Francisco.
Glémin S, Scornavacca C, Dainat J, Burgarella C, Viader V, Ardisson M, Sarah G, Santoni S, David J, Ranwez V. 2019. Pervasive hybridizations in the history of wheat relatives. *Sci Adv* **5:** eaav9188. doi:10.1126/sciadv.aav9188
Gogarten JP, Townsend JP. 2005. Horizontal gene transfer, genome innovation and evolution. *Nature Reviews Microbiol* **3:** 679–687. doi:10.1038/nrmicro1204
Hudson RR. 2002. Generating samples under a Wright–Fisher neutral model of genetic variation. *Bioinformatics* **18:** 337–378. doi:10.1093/bioinformatics/18.2.337
Huson DH, Rupp R, Scornavacca C. 2010. *Phylogenetic networks: concepts, algorithms and applications*. Cambridge University Press, Cambridge.
Kihara H. 1954. Considerations on the evolution and distribution of *Aegilops* species based on the analyser-method. *Cytologia (Tokyo)* **19:** 336–357. doi:10.1508/cytologia.19.336
Koblmüller S, Duftner N, Sefc KM, Aibara M, Stipacek M, Blanc M, Egger B, Sturmbauer C. 2007. Reticulate phylogeny of gastropod-shell-breeding cichlids from Lake Tanganyika: the result of repeated introgressive hybridization. *BMC Evol Biol* **7:** 7. doi:10.1186/1471-2148-7-7
Koonin EV, Makarova KS, Aravind L. 2001. Horizontal gene transfer in prokaryotes: quantification and classification. *Annual Rev Microbiol* **55:** 709–742. doi:10.1146/annurev.micro.55.1.709
Levy AA, Feldman M. 2022. Evolution and origin of bread wheat. *Plant Cell* **34:** 2549–2567. doi:10.1093/plcell/koac130
Li L-F, Liu B, Olsen KM, Wendel JF. 2015. A re-evaluation of the homoploid hybrid origin of *Aegilops tauschii*, the donor of the wheat D-subgenome. *New Phytologist* **208:** 4–8. doi:10.1111/nph.13294
Linz S, Semple C. 2019. Attaching leaves and picking cherries to characterise the hybridisation number for a set of phylogenies. *Adv Applied Math* **105:** 102–129. doi:10.1016/j.aam.2019.01.004
Lutteropp S, Scornavacca C, Kozlov AM, Morel B, Stamatakis A. 2022. NetRAX: accurate and fast maximum likelihood phylogenetic network inference. *Bioinformatics* **38:** 3725–3733. doi:10.1093/bioinformatics/btac396
Marcussen T, Sandve SR, Heier L, Spannagl M, Pfeifer M, International Wheat Genome Sequencing Consortium, Jakobsen KS, Wulff BB, Steuernagel B, Mayer KF, et al. 2014. Ancient hybridizations among the ancestral genomes of bread wheat. *Science* **345:** 1250092. doi:10.1126/science.1250092
Mirzaei S, Wu Y. 2015. Fast construction of near parsimonious hybridization networks for multiple phylogenetic trees. *IEEE-ACM Trans Comput Biol Bioinform* **13:** 565–570. doi:10.1109/TCBB.2015.2462336
Molloy EK, Durvasula A, Sankararaman S. 2021. Advancing admixture graph estimation via maximum likelihood network orientation. *Bioinformatics* **37(**Suppl_1**):** i142–i150. doi:10.1093/bioinformatics/btab267
Müller NF, Stolz U, Dudas G, Stadler T, Vaughan TG. 2020. Bayesian inference of reassortment networks reveals fitness benefits of reassortment in human influenza viruses. *Proc Natl Acad Sci* **117:** 17104–17111. doi:10.1073/pnas.1918304117
Müller NF, Kistler KE, Bedford T. 2022. A Bayesian approach to infer recombination patterns in coronaviruses. *Nat Commun* **13:** 4186. doi:10.1038/s41467-022-31749-8
Pickrell J, Pritchard J. 2012. Inference of population splits and mixtures from genome-wide allele frequency data. *Nat Prec (2012)* doi:10.1038/npre.2012.6956.1
Rimbaud A, Grass NC. 1997. Seq-Gen: an application for the Monte Carlo simulation of DNA sequence evolution along phylogenetic trees. *Bioinformatics* **13:** 235–238. doi:10.1093/bioinformatics/13.3.235
Stamatakis A. 2014. RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. *Bioinformatics* **30:** 1312–1313. doi:10.1093/bioinformatics/btu033
van Iersel L, Janssen R, Jones M, Murakami Y, Zeh N. 2022. A practical fixed-parameter algorithm for constructing tree-child networks from multiple binary trees. *Algorithmica* **84:** 917–960. doi:10.1007/s00453-021-00914-8
Whidden C, Beiko RG, Zeh N. 2013. Fixed-parameter algorithms for maximum agreement forests. *SIAM J Computing* **42:** 1431–1466. doi:10.1137/110845045
Wu Y. 2010. Close lower and upper bounds for the minimum reticulate network of multiple phylogenetic trees. *Bioinformatics* **26:** i140–i148. doi:10.1093/bioinformatics/btq198
Wu Y. 2020. Inference of population admixture network from local gene genealogies: a coalescent-based maximum likelihood approach. *Bioinformatics* **36**(Suppl_1): i326–i334. doi:10.1093/bioinformatics/btaa465
Yamada K, Chen Z-Z, Wang L. 2020. Improved practical algorithms for rooted subtree prune and regraft (rSPR) distance and hybridization number. *J Comput Biol* **27:** 1422–1432. doi:10.1089/cmb.2019.0432
Zhang L. 2019a. Generating normal networks via leaf insertion and nearest neighbor interchange. *BMC Bioinform* **20:** 642. doi:10.1186/s12859-019-3209-3
Zhang L. 2019b. Clusters, trees, and phylogenetic network classes. In *Bioinformatics and phylogenetics* (ed. Warnow T), pp. 277–315. Springer, New York.

Received January 6, 2023; accepted in revised form May 16, 2023.