Journal of Bioinformatics and Computational Biology
Vol. 7, No. 4 (2009) 597–623

Imperial College Press
www.icpress.co.uk

UNIQUENESS, INTRACTABILITY AND EXACT ALGORITHMS: REFLECTIONS ON LEVEL-K PHYLOGENETIC NETWORKS*

LEO VAN IERSEL

Department of Mathematics and Computer Science
Technische Universiteit Eindhoven, P. O. Box 513
5600 MB Eindhoven, The Netherlands
l.j.j.v.iersel@tue.nl

STEVEN KELK

Centrum voor Wiskunde en Informatica (CWI)
P. O. Box 94079, 1090 GB Amsterdam, The Netherlands
s.m.kelk@cwi.nl

MATTHIAS MNICH

Department of Mathematics and Computer Science
Technische Universiteit Eindhoven, P. O. Box 513
5600 MB Eindhoven, The Netherlands
m.mnich@tue.nl

Received 10 January 2008
Revised 25 September 2008
Accepted 26 November 2008

Phylogenetic networks provide a way to describe and visualize evolutionary histories that have undergone so-called reticulate evolutionary events such as recombination, hybridization or horizontal gene transfer. The level k of a network determines how non-treelike the evolution can be, with level-0 networks being trees. We study the problem of constructing level-k phylogenetic networks from triplets, i.e. phylogenetic trees for three leaves (taxa). We give, for each k, a level-k network that is uniquely defined by its triplets. We demonstrate the applicability of this result by using it to prove that (1) for all $k \geq 1$ it is NP-hard to construct a level-k network consistent with all input triplets, and (2) for all $k \geq 0$ it is NP-hard to construct a level-k network consistent with a maximum number of input triplets, even when the input is dense. As a response to this intractability, we give an exact algorithm for constructing level-1 networks consistent with a maximum number of input triplets.

Keywords: Phylogenetic networks; NP-hardness; exact algorithms.

*Part of this research has been funded by the Dutch BSIK/BRICKS project.

1. Introduction

A central problem in biology is to accurately reconstruct plausible evolutionary histories. This area of research is called phylogenetics and provides fascinating challenges for both biologists and mathematicians. Throughout most of the history of phylogenetics, researchers have concentrated on constructing phylogenetic *trees*. In recent years however, more and more attention is devoted to phylogenetic *networks*. From a biological point of view, networks are able to explain and visualize more complex evolutionary scenarios, since they take into account biological phenomena that cannot be displayed in a tree. These phenomena are so-called reticulate evolutionary events such as hybridization, recombination and horizontal gene transfer. From a mathematical point of view however, phylogenetic networks pose formidable challenges. Irrespective of the exact model being used, many problems that are computationally tractable for trees (i.e. solvable in polynomial time) become intractable (NP-hard) for networks. Huson and Bryant wrote a detailed discussion of phylogenetic networks and their application.[1]

Here we study the *level* of networks, which restricts how interwoven the reticulations can be. In trees (i.e. level-0 networks), no reticulation events occur; in level-1 networks, all reticulation cycles must be disjoint. The higher the level of the network, the more freedom in reticulation is allowed. Formally, a level-k network is a phylogenetic network in which each biconnected component contains at most k reticulation events. Level-1 networks have also been called *galled trees*,[2] *gt-networks*[3] and *galled networks*.[4] General level-k networks were first introduced by Choy, Jansson, Sadakane and Sung.[5] The focus on level (as opposed to, for example, minimizing the total number of reticulation vertices) is motivated by several factors. Firstly, level induces a hierarchy on the space of networks with lower level networks being more "tree-like" than higher-level networks. Identifying the position of candidate solutions (i.e. networks) within this hierarchy, or finding the minimum level at which candidate solutions exist, communicates important structural information about the solution space. (Level minimization, which derives its legitimacy from the parsimony principle, can also be used in an implicit context e.g. to measure the accuracy of input data. For example, if we expect the solution to be a tree, but only obtain higher level networks, this suggests that data errors lie in the regions corresponding to the biconnected components.) Secondly, from an algorithmic perspective, focussing on lower-level networks can potentially lead to (1) polynomial-time solvability, (2) better running-times and (3) clearer mathematical analysis. Finally, restricting level is necessary for many optimization criteria to avoid trivial solutions, e.g. several of the problems we discuss in this article can be trivially optimized if we choose a solution with high enough level, but (as we shall see) this communicates no useful information.

A great variety of approaches have been proposed for phylogenetic reconstruction. They include methods like Maximum Parsimony, Maximum Likelihood, Bayesian methods (using Monte Carlo Markov Chain), distance-based methods

(using e.g. Neighbor Joining or UPGMA). Holder and Lewis[6] list advantages and disadvantages of each of these methods. Quartet methods use a different approach, which aims to combine a set of unrooted trees on four leaves each (quartets) into a single tree or network for all leaves (i.e. taxa). The relative performance of quartet methods has been a topic of recent discussion. St. John *et al.*[7] considered the weighted version of this method and showed that quartet methods perform poorly for various weight functions. However, Snir *et al.*[8] showed that quartet methods perform much better for different weight functions in which it is allowed that for some subsets of four leaves all quartets over these four leaves are allowed to have weight zero.

Triplet methods are similar to quartet methods but operate in a rooted context. The input consists of rooted phylogenetic trees on three taxa each and the goal is to construct a directed phylogenetic tree or network for all taxa. The input triplets can, for example, be constructed by methods such as Maximum Parsimony or Maximum Likelihood, that work accurately and fast for small numbers of taxa. Another possibility is to infer the triplets from a set of phylogenetic trees, possibly originating from different sources. However, after the triplets are obtained, the next step is to combine them into a single, large phylogenetic network for all taxa. Designing algorithms for the latter task forms the subject of this article. Triplet methods have become popular since they allow us to solve certain problems in polynomial time, as will be elaborated on shortly. Next to that, an advantage of these methods is that they provide the possibility to combine different sorts of biological data. A possible drawback of triplet-based methods is that it is not yet clear whether the negative results of St. John *et al.*[7] for quartet methods are also relevant here.

Triplet-based methods have been extensively studied in the literature. Aho *et al.*[9] gave a polynomial-time algorithm that constructs a tree from triplets if there exists a tree that is consistent with all input triplets. After Mike Steel[10] linked this result to phylogenetics, it provided the stimulus for studying the applicability of triplet-based methods to networks. Unfortunately, it has been shown that for level-1[4] and level-2[11,12] networks the corresponding problem becomes NP-hard. However, the same articles give polynomial-time algorithms for the problem where the input is *dense*, i.e. there is at least one triplet in the input for every size-3 subset of the taxa. A related problem that accommodates errors in the triplets is finding a tree consistent with as many input triplets as possible. This problem is NP-hard,[13–15] and approximation algorithms have been explored both for the construction of trees[16] and level-1 networks.[4,17] For the construction of trees, efficient heuristics have been designed by Semple and Steel,[18] Page,[19] Wu[15] and Snir and Rao.[20] The last algorithm (MAX CUT triplets) outperforms the character-based method Matrix Representation with Parsimony (MRP), which is popular in practice.[21–23]

This article considers the construction of a level-k phylogenetic network consistent with all (or a maximum number of) input triplets. First, we show for which

values of k it is true that for *any* input triplet set on n leaves there exists a level-k network consistent with all input triplets (in Sec. 3). Then we use this analysis to give, for each k, a level-k network that is uniquely defined by the set of triplets it is consistent with, i.e. no other level-k network is consistent with that set of triplets (in Sec. 4). We use these networks to give two NP-hardness results in Sec. 5. We prove that constructing level-k networks consistent with all input triplets is NP-hard for every $k \geq 1$. This complements the known results for $k \in \{1, 2\}$ (see above), of which our result for $k > 2$ is a non-trivial generalization. In addition, we show that constructing a level-k network consistent with a maximum number of input triplets is NP-hard for all $k \geq 0$, even if the input triplet set is dense. This means that it is even NP-hard to construct a phylogenetic tree consistent with a maximum number of triplets from a *dense* triplet set. We respond to the aforementioned intractability results with an exact algorithm for constructing level-1 phylogenetic networks in Sec. 6. This algorithm runs in time $O(m4^n)$ (for n leaves and m triplets) and can also be used for the weighted version of the problem. The above mentioned article by Snir *et al.*[8] shows that quartet methods are particularly powerful when the input quartets are not forced to contain information for each quadruple of leaves. This suggest that triplet methods might perform especially well on non-dense input sets of triplets. The level-1 algorithm we present can tolerate such inputs and for this reason we are optimistic about its biological relevance. We conclude with a discussion of open problems.

2. Preliminaries

A *phylogenetic network* (*network* for short) is defined as a directed acyclic graph in which a single vertex has indegree 0 and outdegree 2 (the *root*) and all other vertices have either indegree 1 and outdegree 2 (*split vertices*), indegree 2 and outdegree 1 (*reticulation vertices*) or indegree 1 and outdegree 0 (*leaves*), where the leaves are distinctly labeled. Let $L(N)$ denote the set of leaves of a network N.

A directed acyclic graph is *connected* (also "weakly connected") if for every pair of vertices x, y there is an undirected path (ignoring arc-orientations) between x and y. A directed acyclic graph is *biconnected* if it contains no vertex whose removal disconnects the graph. A *biconnected component* of a network is a maximal biconnected subgraph and is called *trivial* if it is equal to two vertices connected by an arc. Otherwise, it is *non-trivial*. An arc $a = (u, v)$ of a network N is a *cut-arc* if its removal disconnects N; it is *trivial* if v is a leaf and otherwise *non-trivial*. A vertex w is *below* an arc $a = (u, v)$ (and *below* vertex v) if there is a directed path from v to w.

Definition 1. A network is said to be a *level-k network* if each biconnected component contains at most k reticulation vertices.

To avoid "redundant" networks, we require every non-trivial biconnected component of a network to have at least three outgoing arcs. A level-k network is a

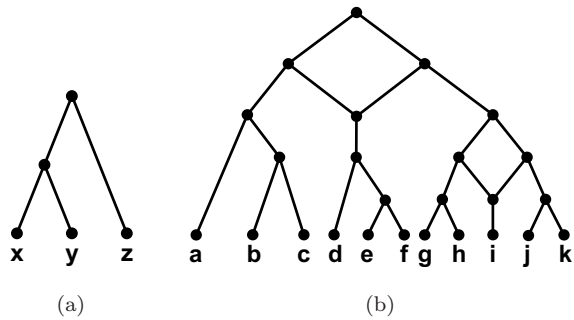


Fig. 1. (a) One of three possible triplets on leaves x, y, z and (b) an example of a level-1 network. As with all figures in this article, all arcs are directed downwards.

strict level-k network if it is not a level-$(k-1)$ network. The class of level-0 networks are *phylogenetic trees* (*trees* for short); they have no reticulation vertices.

A *triplet* $xy|z$ is a tree on leaves x, y, z such that the lowest common ancestor of x and y is a proper descendant of the lowest common ancestor of x and z, see Fig. 1(a). The leaves of triplet t form the set $L(t)$. A set T of triplets has leaf set $L(T) = \bigcup_{t \in T} L(t)$, with size $n = L(T)$. For $L' \subseteq L(T)$, denote by $T|_{L'}$ the set of triplets $t \in T$ with $L(t) \subseteq L'$. A set T of triplets is *dense* if it contains at least one triplet for each size-3 subset of $L(T)$.

Definition 2. A triplet $xy|z$ is *consistent* with a network N (interchangeably: N is consistent with $xy|z$) if N contains a subdivision of $xy|z$, i.e. if N contains vertices $u \neq v$ and pairwise internally vertex-disjoint paths $u \to x$, $u \to y$, $v \to u$ and $v \to z$.

By extension, a set T of triplets is *consistent* with N (interchangeably: N is consistent with T) if every triplet in T is consistent with N and $L(T) = L(N)$. For example, Fig. 1(b) is a level-1 network with two non-trivial biconnected components, each of them containing one reticulation vertex. This network is consistent with (amongst others) the triplets $bc|a$, $bd|h$, $hd|b$ and $gi|k$, but is not consistent with $dg|k$, $ab|c$, $gk|i$ or $cd|f$.

We introduce the class of *simple* level-k networks. Intuitively, these are the basic building blocks of level-k networks in the sense that each non-trivial biconnected component of a level-k network is in essence a simple level-l network, for some $l \leq k$. These simple networks will be built by adding leaves to "generators", which are formally defined as follows.

Definition 3. A *simple level-k generator*, for $k \geq 1$, is a directed acyclic biconnected multigraph, which has a single root (a vertex with indegree 0 and outdegree 2), precisely k reticulation vertices (with indegree 2 and outdegree at most 1) and apart from that only split vertices (with indegree 1 and outdegree 2).

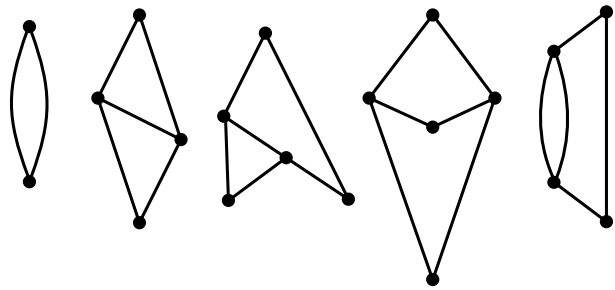


Fig. 2. The unique simple level-1 generator and the four simple level-2 generators.

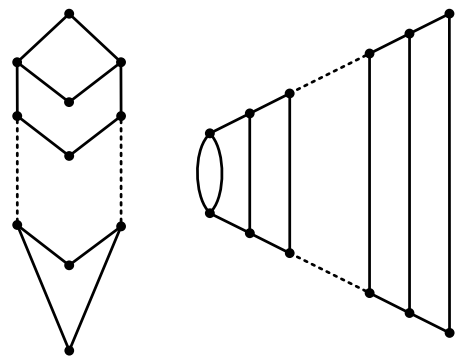


Fig. 3. Two examples of simple level-k generators; the left one has a maximum number of vertices and arcs and the right one a minimum number. Remember that all arcs are directed downwards.

A case analysis shows that there is only one simple level-1 generator and that there are four simple level-2 generators,[12] depicted in Fig. 2. Computer calculations revealed the 65 simple level-3 generators.[24] For high k, the number of different simple level-k generators is huge. In Fig. 3, two examples are shown, one with a maximum and one with a minimum number of vertices and arcs.

Lemma 1. *If $G = (V, A)$ is a simple level-k generator then $2k \leq |V| \leq 3k - 1$ and $3k - 1 \leq |A| \leq 4k - 2$.*

Proof. Suppose G contains r_0 vertices with indegree 2 and outdegree 0, r_1 vertices with indegree 2 and outdegree 1 and s vertices with indegree 1 and outdegree 2. The sum of the indegrees of all vertices is $s+2r_1+2r_0$, while the sum of their outdegrees is $2+2s+r_1$. It is well known that in any directed graph the sum of all outdegrees equals the sum of all indegrees. It follows that $s = r_1+2r_0-2$. Because $r_0+r_1 = k$, it follows that $|V| = 1+s+r_1+r_0 = 2k-1+r_0$ and $|A| = \frac{1}{2}(2+3s+3r_1+2r_0) = 3k-2+r_0$. The number of vertices and arcs in a simple level-k generator thus only depends on k and r_0. Since $1 \leq r_0 \leq k$ the lemma follows. □

Definition 4. A simple level-k network, for $k \geq 1$, is a network obtained by applying the following transformation to some simple level-k generator G:

(1) first, for each pair u, v of vertices in G connected by a single arc (u, v), replace (u, v) by a path with $\ell \geq 0$ internal vertices and for each such internal vertex w, add a new leaf x and an arc (w, x);
(2) second, for each pair u, v of vertices in G connected by multiple arcs, replace one such arc by a path with at least one internal vertex and for each such internal vertex w, add a new leaf x and an arc (v, x); and treat the other arc between u, v as in step (1);
(3) third, for each vertex v of G with indegree 2 and outdegree 0, add a new leaf y and an arc (v, y).

We remark that at least three leaves have to be added to G, to avoid redundancy of the constructed network. A network is *simple* if it is a simple level-k network for some k. There is an elegant characterization of simple level-k networks:

Lemma 2 (Van Iersel *et al.*[12]). *For $k \geq 1$, a network N is a simple level-k network if and only if N is a strict level-k network and every cut-arc is trivial.*

In our proofs, we will frequently remove leaves from a network. This might result in a graph that is not a valid network. Therefore, we define *tidying up* a directed acyclic graph as repeatedly applying the following four operations: (1) delete unlabeled vertices with outdegree 0; (2) suppress vertices with indegree and outdegree 1; (3) replace multiple arcs by single arcs; (4) remove the root if it has outdegree 1 and (5) replace a non-trivial biconnected component by a single vertex if there are at most two arcs going out of this biconnected component. Observe that if N' is the result of removing leaves L' from network N and tidying up the resulting graph then N' is a valid network. In addition, observe that in this case N' is consistent with exactly the same triplets as N is, except for triplets containing leaves from L'.

3. Sufficiency and Necessity of Network Level

In this section, we prove that any triplet set on n leaves is consistent with a level-$(n-1)$ network. Then we show that this bound is tight, by giving a triplet set on n leaves that is not consistent with any network of level smaller than $n-1$.

Let $T_F(n)$ be the set of all $3\binom{n}{3}$ triplets possible on n leaves. Call $T_F(n)$ the *full triplet set* on n leaves.

Proposition 1. *For any triplet set T on n leaves, there exists a level-$(n-1)$ network consistent with T.*

Proof. Let $n \geq 3$ and let $N_F(n)$ be the network in Fig. 4. First, look at triplets $x_h x_i | x_j \in T_F(n)$ with $h, i \neq n$. There exists a unique split vertex v below the left child of the root, from which there are two paths to x_h and x_i that have

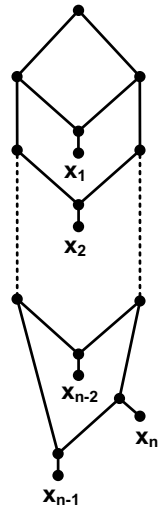


Fig. 4. The level-$(n-1)$ network $N_F(n)$ is consistent with every triplet set on n leaves, and has the minimum number of arcs and vertices among all such networks.

only v in common. On the other hand, there is a path from the root to x_j, via the right child of the root. So the network is consistent with $x_h x_i | x_j$.

Second, look at triplets $x_i x_n | x_j \in T_F(n)$. There exists a unique split vertex v below the right child of the root, from which there are two paths to x_i and x_n that have only v in common. As there is also a path from the root to x_j via the left child of the root, the network is consistent with $x_i x_n | x_j$. Given that $T \subseteq T_F(n)$, the result follows. □

Lemma 3. *Any network consistent with the full triplet set must be simple.*

Proof. Let $n \geq 3$ and let N be consistent with $T_F(n)$. If N is not simple then, by Lemma 2, it contains a non-trivial cut-arc $a = (u, v)$. If there is only one leaf below a, then N is not a valid network because it contains a biconnected component with only one outgoing arc, which we do not allow. If all leaves are below a then again N is not a valid network because it contains a biconnected component with only one outgoing arc. Hence there are leaves x and y below a and a leaf z not below a. This implies that the triplet $xz|y$ is not consistent with N, a contradiction. □

Let a *reticulation leaf* be defined as a leaf whose parent is a reticulation vertex. Simple level-k networks have, by definition, at least one and at most k reticulation leaves ($k \geq 1$).

Proposition 2. *The full triplet set on $n \geq 3$ leaves is not consistent with any level-k network with $k < n - 1$.*

Proof. The proof is by induction on n. The theorem holds for $n = 3$: any network consistent with $T_F(3)$ must be simple by Lemma 3, and any simple level-1 network on three leaves is consistent with only two triplets. Since there are three possible triplets on a set of three leaves, there is no level-1 network consistent with $T_F(3)$. Let $n > 3$; the induction hypothesis is that for all $n' < n$ the full triplet set $T_F(n')$ is not consistent with any level-k' network for $k' < n' - 1$. Suppose for contradiction that the theorem does not hold for n, thence there exists a level-k network N consistent with $T_F(n)$ and $k < n - 1$. By Lemma 3, N must be a simple level-k network and thus contains a reticulation leaf x. Delete x and tidy up the resulting graph. This decreases the level of the network since the parent of x is a reticulation vertex and gets removed when the graph is tidied up. This thus yields a level-$(n-3)$ network consistent with $T_F(n-1)$, contradicting the induction hypothesis. We thus conclude that there exists no level-k network consistent with $T_F(n)$ for $k < n - 1$. □

The network $N_F(n)$ of Fig. 4 is much smaller than the network proposed by Jansson and Sung,[25] that is consistent with $T_F(n)$ and was obtained from a complicated sorting network.

Lemma 4. *For $n \geq 3$, the network $N_F(n)$ has the minimum number of arcs and vertices over all networks consistent with the full triplet set $T_F(n)$.*

Proof. We first show that any simple level-k network $N = (V, A)$ on n leaves has $2n + 2k - 1$ vertices and $2n + 3k - 2$ arcs. Let s be the number of split vertices. The sum of the indegrees of all vertices is $s+2k+n$, while the sum of their outdegrees is $2 + 2s + k$. Again, we use that the sum of all outdegrees equals the sum of all indegrees. It follows that $s = n + k - 2$. Using this formula, we obtain that the total number of vertices equals:

$$|V| = s + k + n + 1 = (n + k - 2) + k + n + 1 = 2n + 2k - 1.$$

The total number of arcs in N is $|A| = \frac{1}{2}(3s + 3k + n + 2) = \frac{1}{2}(3(n + k - 2) + 3k + n + 2) = 2n + 3k - 2$.

Let N_n be a network consistent with $T_F(n)$. By Lemma 3 and Proposition 2, N_n is simple and has level at least $n - 1$. Then the calculation above yields that N_n has at least $2n + 2(n - 1) - 1 = 4n - 3$ vertices and $2n + 3(n - 1) - 2 = 5(n - 1)$ arcs. The proof is complete by noting that $N_F(n)$ has exactly $4n - 3$ vertices and $5(n - 1)$ arcs. □

4. A Unique Level-*k* Network

In the construction and analysis of triplet methods, it is often important to know that a certain network is uniquely defined by a set of triplets. Characterizing such networks is an important open problem. In this section, we present a partial solution to this question by giving, for each k, a level-k network N^k that is unique in the

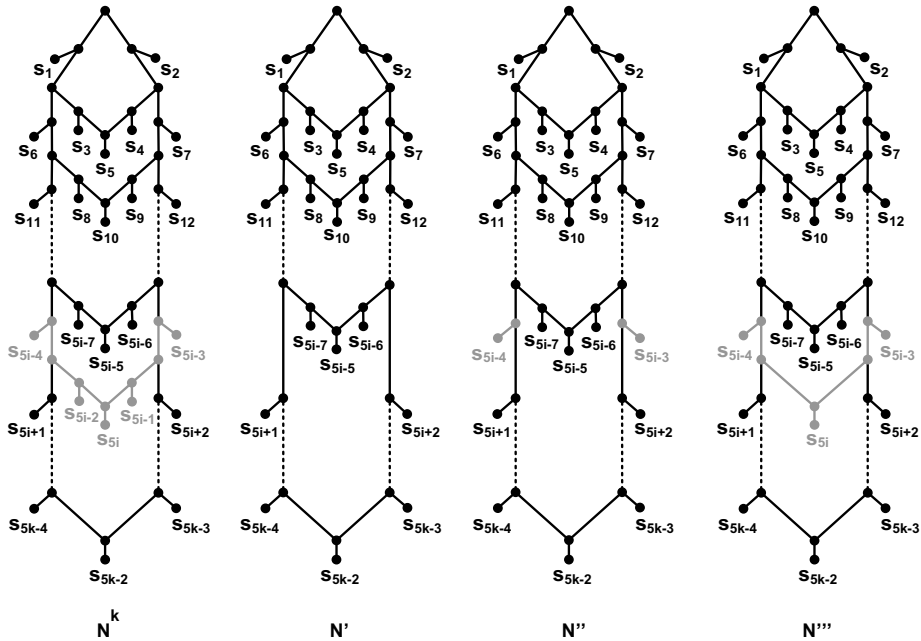


Fig. 5. Removing leaves $s_{5i-4}, \ldots, s_{5i}$ from N^k and tidying up the graph gives N'. Adding s_{5i-4} and s_{5i-3} to N' gives N''; adding s_{5i} to N'' gives N'''; subsequently adding s_{5i-2} and s_{5i-1} gives the original network N^k to the left again.

sense that it is the only level-k network that is consistent with all triplets that are consistent with N^k. In the next section, we will demonstrate how useful this unique network is, by using it to show the intractability of constructing level-k networks from triplets.

Let N^k be the network to the left in Fig. 5 and let T^k be the set of triplets that are consistent with N^k. By *hanging leaves* $x_1, \ldots, x_p$ on an arc (u, v) (for some $p \geq 1$) we mean replacing (u, v) by a path $u, w_1, \ldots, w_p, v$ and adding arcs (w_i, x_i) for all $i = 1, \ldots, p$.

Theorem 1. *For each $k \geq 2$, the network N^k is the unique level-k network consistent with T^k.*

Proof. Let R be the set of reticulation leaves of N^k, that is $R = \{s_5, s_{10}, \ldots, s_{5k-5}, s_{5k-2}\}$. We start by proving the following claims.

Claim 1. *Any level-k network consistent with T^k is a simple level-k network.*

Proof of Claim 1. First observe that all triplets over the leaves $R \cup \{s_{5k-4}\}$ are in T^k. Let N be a level-k network consistent with T^k. From Proposition 2 it follows that N is a strict level-k network. Now suppose for contradiction that N is not simple. Then by Lemma 2, N contains a non-trivial cut-arc a. Let $B \subseteq L(N)$

be the set of leaves below a and let $A = L(N)\backslash B$. Because a is non-trivial, B contains at least two leaves. For every two leaves x, y in B and every leaf z in A, there is only one triplet in T^k on leaves x, y, z that is consistent with N. However, for s_{5k-2} there are no two leaves x', y' such that there is only one triplet in T^k with leaves s_{5k-2}, x', y'. It follows that s_{5k-2} belongs to neither A nor B, a contradiction. □

Claim 2. *In any level-k network consistent with T^k, at least one of the leaves in R is a reticulation leaf.*

Proof of Claim 2. Let N be a network consistent with T^k. Recall that T^k contains all possible triplets over leaves $R \cup \{s_{5k-4}\}$. Proposition 2 says that any network consistent with all triplets over $R \cup \{s_{5k-4}\}$ cannot have level smaller than k, so N is a strict level-k network. By Claim 1, N is a simple level-k network and hence contains a reticulation leaf x. First suppose x does not belong to $R \cup \{s_{5k-4}\}$. Then removing x and tidying up the resulting graph yields a level-$(k-1)$ network consistent with all triplets over $R \cup \{s_{5k-4}\}$. This is a contradiction, thus N contains no leaves outside $R \cup \{s_{5k-4}\}$ as reticulation leaf. Symmetrically, no leaf outside $R \cup \{s_{5k-3}\}$ is a reticulation leaf of N. It follows that only leaves from R can be reticulation leaves of N, so x belongs to R. □

We are now ready to prove the theorem. The proof is by induction on k; the base case $k = 2$ has been proven by van Iersel *et al.* [Ref. 12, Lemma 18]. Let $k > 2$ and assume the theorem holds for all $k' = 2, \ldots, k-1$. In the induction step, we will show that any level-k network consistent with T^k and with reticulation leaf s_{5i} (for any $i \in \{1, \ldots, k-1\}$), equals the network N^k. The case that s_{5k-2} is a reticulation leaf is symmetric to the case that s_{5k-5} is a reticulation leaf. Since by Claim 2 at least one leaf from R must be a reticulation leaf, the theorem will follow.

Let N be a simple level-k network consistent with T^k and let $i \in \{1, \ldots, k-1\}$ be such that s_{5i} is a reticulation leaf in N. Let T' be the triplet set obtained from T^k by removing all triplets containing some leaf from $\{s_{5i-4}, \ldots, s_{5i}\}$, i.e. $T' = T^k|_{(L\backslash\{s_{5i-4},\ldots,s_{5i}\})}$. Then T' is consistent with network N', the second network from the left in Fig. 5. Because T' equals the set of all triplets that are consistent with N' (which is a relabeling of N^{k-1}), by the induction hypothesis N' is the unique level-$(k-1)$ network consistent with T'.

Consider the network obtained from N by removing the leaves s_{5i-1}, s_{5i-2}, s_{5i}, s_{5i-3} and s_{5i-4} (in this order) from N and tidying up the resulting graph. This decreases the level of the network, since the parent of s_{5i} was a reticulation vertex and gets removed when tidying up the graph. Hence this gives a level-$(k-1)$ network consistent with T', which by the induction hypothesis equals N'.

To show that N equals N^k, consider the network N' and apply the reverse of the operation that removed the leaves $s_{5i-1}, s_{5i-2}, s_{5i}, s_{5i-3}$ and s_{5i-4} from N. This process is illustrated in Fig. 5, and we will show that the such obtained network will equal N^k. Process the leaves in reverse order, so add s_{5i-4} to N' first.

Since N' has $k-1$ reticulation leaves and s_{5i} also has to become a reticulation leaf, s_{5i-4} must be a leaf below a split vertex. Hence s_{5i-4} is added to the network by hanging s_{5i-4} on some arc of N'. The same holds for s_{5i-3}. Since s_{5i} was a reticulation leaf in N, it is added to the network choosing two arcs $(u_1, v_1), (u_2, v_2)$, subdividing them into $(u_1, w_1), (w_1, v_1)$ and $(u_2, w_2), (w_2, v_2)$, respectively, and adding a new reticulation vertex x and arcs $(w_1, x), (w_2, x), (x, s_{5i})$. Subsequently, s_{5i-2} and s_{5i-1} are added to the network by hanging them on arcs to be specified. It remains to determine which arcs to subdivide, as to add the leaves $s_{5i-1}, s_{5i-2}, s_{5i}, s_{5i-3}$ and s_{5i-4}.

First consider the case $i > 1$. Because $s_{5k-4}s_{5i+1}|s_{5i-4}$ and $s_{5i-4}s_{5i+1}|s_{5i-7}$ are triplets in T^k, it follows that s_{5i-4} is added to N' by hanging it on the arc entering the parent of s_{5i+1}. Symmetrically, s_{5i-3} is hung on the arc entering the parent of s_{5i+2}. This leads to network N'' in Fig. 5. Next we discuss how to add s_{5i} to network N''. Triplets $s_{5i}s_{5i+1}|s_{5i-4}$ and $s_{5k-4}s_{5i+1}|s_{5i}$ force a subdivision of the arc between the parents of s_{5i-4} and s_{5i+1}. For symmetric reasons, also the arc between the parents of s_{5i+2} and s_{5i-3} has to be subdivided. So subdivide these arcs and make s_{5i} a reticulation leaf below them (as described in detail in the previous paragraph). This leads to the network N''' in Fig. 5. Now s_{5i-2} and s_{5i-1} can only be added to the network by hanging them on the arcs entering the parent of s_{5i}, since $s_{5i+1}s_{5i-2}|s_{5i}, s_{5i}s_{5i-2}|s_{5i+1} \in T^k$ and $s_{5i+2}s_{5i-1}|s_{5i}, s_{5i}s_{5i-1}|s_{5i+2} \in T^k$. This leads to the leftmost network in Fig. 5, which is the network N^k.

The case $i = 1$ is slightly different, since a leaf s_{5i-7} does not exist. However, the triplets $s_{5k-4}s_6|s_1$ and $s_6s_1|s_7$ enforce that $s_1 = s_{5i-4}$ is added to N' by hanging it on the arc entering the parent of s_6. Symmetrically, $s_2 = s_{5i-3}$ must be hung on the arc entering the parent of s_7. The same arguments as in the case $i > 1$ show how to add the leaves $s_{5i}, s_{5i-1}, s_{5i-2}$. Also in this case we obtain the network N^k.

It follows that N equals N^k, completing the proof of Theorem 1. □

For level 1, the network N^1 is not the only network consistent with T^1. Figure 6(a) shows the three networks that are consistent with T^1. However, there does exist a level-1 network that is unique in this sense. It is not too difficult to argue

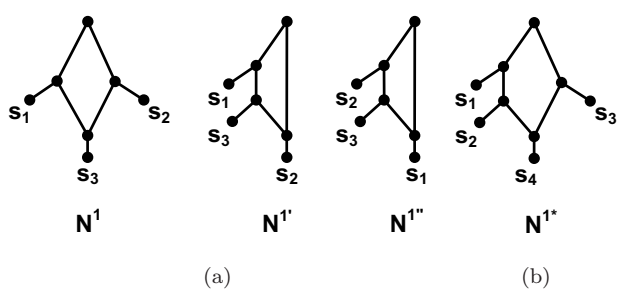


Fig. 6. (a) The three networks $N^1, N^{1\prime}$ and $N^{1\prime\prime}$ that are consistent with $T^1 = \{s_1s_3|s_2, s_2s_3|s_1\}$ and (b) network N^{1*}, which is the unique network that is consistent with the set of triplets consistent with N^{1*}.

that the network N^{1^*} in Fig. 6(b) is the only level-1 network that is consistent with all triplets that are consistent with N^{1^*}. For level 0, the only tree consistent with a single triplet is the triplet itself.

5. From Uniqueness to Intractability of Constructing Level-k Networks

In this section, we show how to use the unique networks from the previous section in the complexity analysis of network reconstruction methods from triplets. We demonstrate this in two NP-hardness proofs. First, we show that it is NP-hard, for each $k \geq 1$, to decide whether a given triplet set is consistent with some level-k network. Secondly, we show that the maximization variant of this problem is NP-hard for each $k \geq 0$ even for dense triplet sets.

We start with the proof that it is NP-hard to construct a level-k network consistent with all input triplets. Hardness was already known for $k = 1$[4] and $k = 2$.[12] Note that the uniqueness result from the previous section plays a crucial role in the subsequent NP-hardness proof for level k. Indeed, the NP-hardness proof is by no means an immediate consequence of the hardness for levels 1 and 2. This is because, although the core of the reduction is the same as the level 1 and 2 cases (i.e. reduction from SET SPLITTING), identifying a unique structure upon which the reduction can be constructed is non-trivial.

In the proofs, we will often say we "hang" a leaf or "caterpillar" from a "side" of a simple level-k generator. A network is a *caterpillar* if deleting all leaves gives a directed path. In simple level-k generators, a *side* is either an arc or a vertex with outdegree zero (cf. Ref. 12). *Hanging a caterpillar from arc* S_i means subdividing S_i and connecting the new vertex to the root of the caterpillar. Similarly defined is *hanging a caterpillar from a vertex with outdegree zero*, which gets connected to the root of the caterpillar. Hanging a leaf from a side is defined similarly. In addition, a leaf x *is on side* S_i if there exists a cut-arc (u, v) such that u is on a subdivision of S_i (if S_i is an arc) or u is a reticulation vertex (if S_i is a reticulation vertex), and there is a directed path from v to x (possibly $v = x$). A leaf x is said to *hang between* vertices w and q if there is a cut-arc (u, v) such that u is on a directed path from w to q and there is a directed path from v to x.

Theorem 2. *For each* $k \geq 2$, *it is NP-hard to decide whether for a triplet set* T *there exists some level-*k *network* N *consistent with* T.

Proof. Reduce from the following NP-hard problem[26]:

Set Splitting

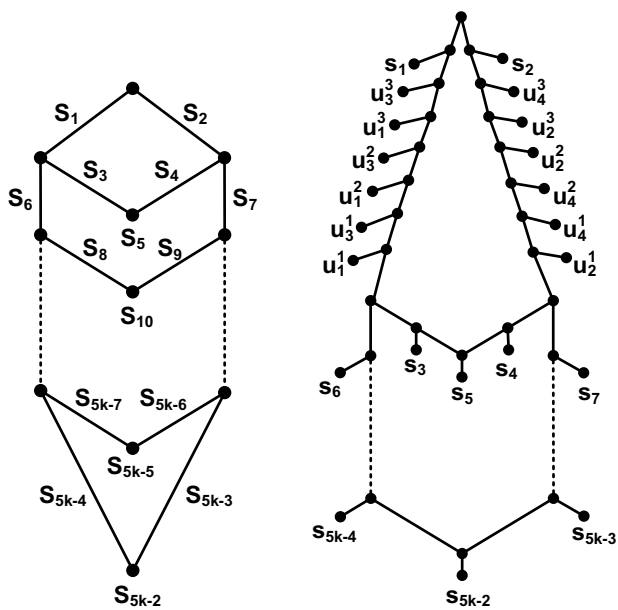
Instance: A set $U = \{u_1, \ldots, u_n\}$ and a collection $\mathcal{C} = \{C_1, \ldots, C_m\}$ of size-3 subsets of U.

Question: Can U be partitioned into sets U_1 and U_2 (a set splitting) such that $C_j \not\subseteq U_1$ and $C_j \not\subseteq U_2$, for all $1 \leq j \leq m$?

From an instance $(U, \mathcal{C})$ of Set Splitting construct a set T of triplets as follows. Start with triplet set T^k (see previous section), and for each set $C_j = \{u_a, u_b, u_c\} \in \mathcal{C}$ (with $1 \leq a < b < c \leq n$) add triplets $u_a^j s_5 | u_b^j$, $u_b^j s_5 | u_c^j$ and $u_c^j s_5 | u_a^j$. In addition, for every $u_i \in U$ and $1 \leq j \leq m$ add triplets $s_5 u_i^j | s_1$, $s_5 u_i^j | s_2$, $s_5 s_6 | u_i^j$, $s_5 s_7 | u_i^j$ and (if $j \neq m$) $u_i^j u_i^{j+1} | s_5$. This completes the construction of T. We will prove that T is consistent with some level-k network if and only if there exists a set splitting $\{U_1, U_2\}$ of $(U, \mathcal{C})$.

First suppose that there exists a set splitting $\{U_1, U_2\}$ of $(U, \mathcal{C})$. Construct the network N by starting with the network N^k, which is obtained from the simple level-k generator G^k in Fig. 7 by hanging a leaf s_i on each side S_i. For each element $u_i \in U_1$, hang all leaves $u_i^1, \ldots, u_i^m$ on side S_1 below the parent of s_1; for each element $u_i \in U_2$ hang all leaves $u_i^1, \ldots, u_i^m$ on side S_2 below the parent of s_2. To determine the order in which to put these leaves, consider a set $C_j = \{u_a, u_b, u_c\} \in \mathcal{C}$. If u_a and u_b are in the same class of the partition then put leaf u_a^j below u_b^j; if u_b and u_c are in the same class of the partition put u_b^j below u_c^j; and if u_a and u_c are in the same class put u_c^j below u_a^j. The rest of the ordering is arbitrary. It is easy to check that N is consistent with all triplets in T. For an example of this construction, see the network to the right in Fig. 7.

Conversely, suppose that T is consistent with some level-k network N. Since $T^k \subset T$, Theorem 1 says that N must be equal to N^k with the leaves not



(a) (b)

Fig. 7. Auxiliary networks in the proof of Theorem 2.

in $L(N^k)$ added. Triplets $s_5u_i^j|s_1$ and $s_5u_i^j|s_2$ imply that none of the leaves u_i^j can hang between the root and s_1, or between the root and s_2. Further, triplets $s_5s_6|u_i^j$ and $s_5s_7|u_i^j$ imply that u_i^j must be on either side S_1 or S_2. Triplets $u_i^ju_i^{j+1}|s_5$ yield that for each $1 \leq i \leq n$, all leaves $u_i^1, \ldots, u_i^m$ have to hang on the same side. For $h \in \{1, 2\}$, let U_h be the set of elements $u_i \in U$ for which all leaves $u_i^1, \ldots, u_i^m$ hang on side S_h. It remains to prove that (U_1, U_2) is a set splitting of $(U, \mathcal{C})$. Consider a set $C_j = \{u_a, u_b, u_c\}$ and suppose for contradiction that $u_a, u_b, u_c \in U_h$ for some $h \in \{1, 2\}$. It follows that all leaves u_a^j, u_b^j, u_c^j hang between s_h and the root. This is impossible, as T contains triplets $u_a^js_5|u_b^j$, $u_b^js_5|u_c^j$, and $u_c^js_5|u_a^j$. □

For dense triplet sets, it can be decided in polynomial time whether there exists a level-1[4] or level-2[11] network consistent with all input triplets. Using the uniqueness result from the previous section, we will prove that the maximization versions of these problems are NP-hard, even for dense triplet sets and for all $k \geq 0$.

MaxCL-k-Dense

Instance: A dense triplet set T.
Output: A level-k network consistent with the maximum number of triplets in T that any level-k network is consistent with.

Theorem 3. *The problem* MaxCL-k-Dense *is NP-hard, for all* $k \geq 0$.

Proof. Reduce from the following NP-hard problem.[27,28]

Feedback Arc Set in Tournaments (FAST)

Instance: A complete directed graph $G = (V, A)$ and an integer $q \in \mathbb{N}$.
Question: Is there a set $A' \subseteq A$ of q arcs such that $G' = (V, A \backslash A')$ is acyclic?

For $k = 0$, we imitate the reduction of the non-dense case.[13,15] The difference is that the constructed instance of MaxCL-0-Dense contains more triplets, to become dense. Given an instance $G = (V, A)$ and $q \in \mathbb{N}$ of FAST, construct an instance T of MaxCL-0-Dense as follows. Introduce a vertex $x \notin V$ and for each arc $(z, y) \in A$, add a triplet $xy|z$ to T. In addition, for each combination of three leaves $v_1, v_2, v_3 \in V$ (thus $v_1, v_2, v_3 \neq x$), add all three triplets $v_1v_2|v_3$, $v_1v_3|v_2$ and $v_2v_3|v_1$ to T. The differences with the reduction of Wu[15] are that (1) we reduce from FAST instead of FAS and that (2) we add all triplets containing three leaves from V. The combination of these two modifications makes the instances dense. The extra triplets do not change the reduction since any level-0 network is consistent with exactly one triplet for every combination of three leaves. The intuition of the reduction is as follows: the vertices of an acyclic graph can be uniquely labeled such that arcs point only from vertices with higher label to vertices with lower label. In a phylogenetic tree, this ordering of the vertices corresponds to an ordering of the leaves on the unique path from the tree root to leaf x. Along the lines of the proof for the non-dense case,[13,15] it can be argued that G contains a feedback arc set

of size q if and only if there exists a tree consistent with $|T| - q - 2\binom{|V|}{3}$ triplets from T. This completes the proof that MaxCL-0 is NP-hard for dense triplet sets.

For $k \geq 2$, use a similar reduction but start from the simple level-k generator G^k in Fig. 7(a). Use the following property of G^k, implied by Theorem 1: Let N^k be a network obtained by hanging a leaf from each side of G^k. If T^k is the triplet set consistent with N^k then N^k is the unique level-k network consistent with T^k.

Given a tournament $G = (V, A)$ and integer $q \in \mathbb{N}$, construct a corresponding instance T of MaxCL-k-Dense as follows. First construct a network N' from G^k. From each side S_i of G^k hang a caterpillar with leaves $S_i^1, \ldots, S_i^p$, with $p = 2(q + 2\binom{|V|}{3}) + 1$. The intuition being that p is "large" to force a specific structure of the networks consistent with many triplets in T. For simplicity, denote S_{5k-2}^1 by x. Hang $|V|$ leaves on side S_{5k-4}, distinctly labeled by the vertices of V, between the root of the caterpillar and the reticulation vertex on that side, in arbitrary order. This gives the network N'. For an example, see the network on the right in Fig. 8. Let T' be the set of triplets consistent with N', except for triplets $ab|c$ with $a, c \in V$ and $b \notin V$. For each arc $(z, y) \in A$, add a triplet $xy|z$ to T', informally encoding the arc (z, y) as a constraint "z hangs between the root of the caterpillar and y". Finally, for each 3-set of vertices from V add all three triplets over the three leaves labeled by the vertices, that are not yet present. Denote the resulting (dense) triplet

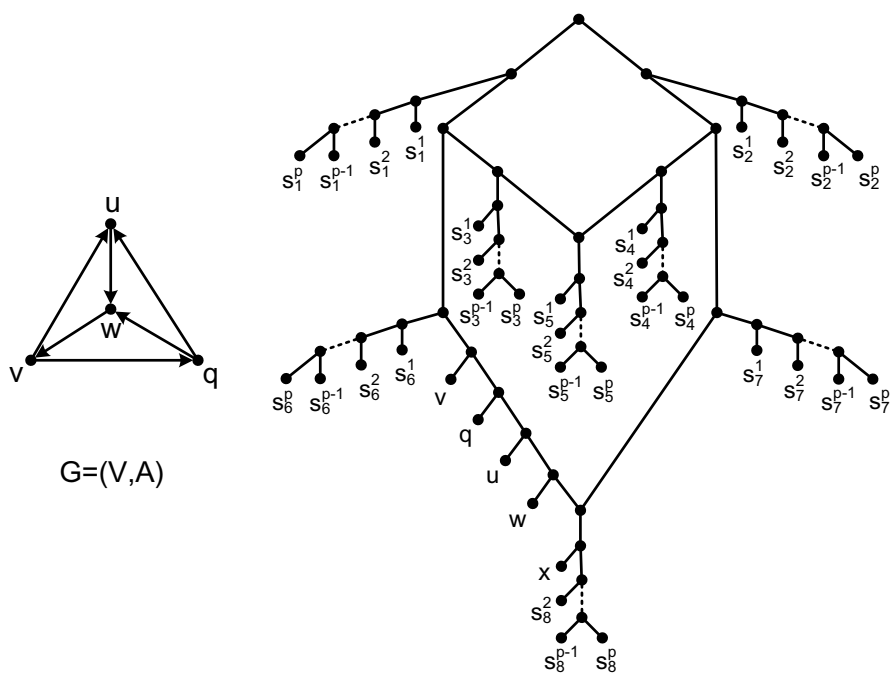


Fig. 8. An example input $G = (V, A)$ of FAST on the left and the network N constructed in the proof of Theorem 3, for $k = 2$, to the right.

set by T, which forms an instance of MaxCL-k-Dense. We will show that there exists a level-k network N consistent with $|T| - q - 2\binom{|V|}{3}$ triplets from T if and only if there exists a feedback arc set A' of size q.

First suppose G has a feedback arc set A' of size q. Thus the graph $G' = (V, A \backslash A')$ is acyclic, and each vertex $v \in V$ can receive a label $f(v)$ such that there are no arcs $(z, y) \in A \backslash A'$ with $f(y) \geq f(z)$. Construct the network N from N' by rearranging the leaves from V by sorting them with respect to their labels such that the highest leaf has the largest label. For any arc $(z, y) \in A \backslash A'$, it holds that $f(y) < f(z)$ and hence the triplet $xy|z$ is consistent with N. For every vertex pair $\{z, y\}$, the triplet $yz|x$ is consistent with N. For each combination of three leaves from V, there is exactly one triplet over these leaves consistent with N. It follows that the only triplets in T that are not consistent with N are (1) the triplets corresponding to the arcs in A', and (2) exactly two-thirds of the triplets that have only leaves in V. It means that in total $|T| - q - 2\binom{|V|}{3}$ triplets from T are consistent with N.

For the converse, suppose there exists some level-k network N consistent with $|T| - q - 2\binom{|V|}{3}$ triplets from T. For all $1 \leq j \leq p$, there exists a unique network with leaf set $L_j = \{S_i^j \mid 1 \leq i \leq 5k-2\}$ that is consistent with all triplets from $T_j = T|_{L_j}$. There are at most $q + 2\binom{|V|}{3}$ triplets not consistent with N, and the sets T_j are pairwise disjoint, so at least one of the sets L_j is placed on a simple level-k network of type G^k. Take any i and observe that for each j such that S_i^j is not on side S_i of N, there exists a triplet $t \in T$ that is not consistent with N and $L(t) = \{S_i^j, \ell_1, \ell_2\}$ for $\ell_1, \ell_2 \notin \{S_i^1, \ldots, S_i^p\}$. If there would be more than $q + 2\binom{|V|}{3}$ such j then there would be more than $q + 2\binom{|V|}{3}$ distinct triplets from T not consistent with N. Hence, for each i there are at least $p' = q + 2\binom{|V|}{3} + 1$ indices j such that S_i^j is on side S_i. Let $L_1^*, \ldots, L_{p'}^*$ be pairwise disjoint sets each containing exactly one leaf S_i^j that is on side S_i, for each $1 \leq i \leq 5k-2$.

The next claim is that all leaves labeled by vertices from V have to be on side S_{5k-4}, between the root of the caterpillar and the reticulation vertex on that side. Suppose for contradiction this were not the case for some leaf labeled by $v \in V$. Then for each of the leaf sets $L_1^* \cup \{v\}, \ldots, L_{p'}^* \cup \{v\}$ there exists a triplet in T not consistent with N. Since the sets $L_1^*, \ldots, L_{p'}^*$ are pairwise disjoint and $p' > q + 2\binom{|V|}{3}$, we obtain a contradiction.

Since the leaves corresponding to vertices from V all hang on the same side S_{5k-4}, they can be uniquely labeled by their order on side S_{5k-4}, such that the highest leaf has the largest label. If some leaves are below the same cut-arc then they receive the same label. Let A' be the set of arcs (z, y) corresponding to the triplets $xy|z$ that are not consistent with N, and for every $v \in V$ let $f(v)$ be the label of the leaf corresponding to v. Then the graph $G' = (V, A \backslash A')$ is acyclic, because all arcs $(z, y) \in A \backslash A'$ satisfy the relation $f(y) < f(z)$.

An example for $k = 2$ is displayed in Fig. 8. The graph on the left is an example instance $G = (V, A)$ of FAST. The arcs of G are encoded as triplets $xw|u$, $xw|q$, $xu|v$, $xv|w$, $xu|q$ and $xq|v$. The network N to the right is consistent with all these

triplets except $xv|w$. The arc (w, v) is indeed a feedback arc set of the graph G. Other triplets in T enforce this specific level-2 network N and make T dense.

For $k = 1$, the same reduction as for $k \geq 2$ works, when hanging two caterpillars from side S_1. □

6. An Exact Algorithm for Constructing Level-1 Networks

Given the intractability results from the previous section for constructing networks consistent with a maximum number of input triplets, there is no hope (unless P = NP) for algorithms solving these problems exactly and in polynomial time. Still, these problems need to be solved in practice, so algorithms for MaxCL-k-Dense and its relaxation MaxCL-k to general triplet sets are either not guaranteed to give an optimal solution, or require superpolynomial time. In this section, we consider the latter approach.

Wu described an exact algorithm[15] that finds a tree consistent with a maximum number of input triplets in $O(3^n(n^2+m))$ time and $O(2^n)$ space complexity, with m the number of triplets and n the number of leaves. We extend this approach for reconstructing evolutions that are not tree-like, but where reticulation cycles are disjoint. We do this by describing an exact algorithm that runs in $O(m4^n)$ time and solves the MaxCL-1 problem, which is NP-hard by Theorem 3.

Note that the problem MaxCL-1 does not only ask if there exists a level-1 network consistent with *all* input triplets; it asks us to find a level-1 network that is consistent with a maximum number of them. Hence an algorithm for MaxCL-1 always outputs a solution, no matter how bad the data are that the algorithm is confronted with. This contrasts with existing algorithms[9,11,25] that only find a solution if a network exists that is consistent with all triplets of a (dense) input. The algorithm described in this section is also more powerful in that it also works for non-dense triplet sets. It can thus be used even if for some combinations of three taxa it is difficult to find the right triplet, which is very likely to be the case in practice. The very same algorithm works for the weighted version of the problem. In addition, it can also be used to choose, among all level-1 networks consistent with a maximum number of input triplets, a network with a minimum number of reticulation vertices. However, its exponential running time means that it can only be used for a relatively small number of leaves at a time.

The intuition behind our algorithm is the following. There are three different shapes possible for the optimal network. Either the arcs leaving the root are cut-arcs, like in Fig. 11(b), or the root is part of a cycle, which can be "skew" like the cycle in Fig. 12(a) or "non-skew" like in Fig. 12(b). We construct a candidate network of each type separately. Given the tripartition (X', Y', Z') (with possibly $Z = \emptyset$) of the leaves indicated in the figures, it turns out to be possible to reconstruct the optimal network by combining optimal smaller networks for X', Y', $X' \cup Z'$ and $Y' \cup Z'$. It is critical that these smaller networks for $X' \cup Z'$ and $Y' \cup Z'$ must be such that combining two networks does not create biconnected components with more than one reticulation vertex.

To achieve this, we introduce the notion of "non-cycle-reachable"-arc, or ncr-arc for short. An arc $a = (u, v)$ is an *ncr-arc* if there is no directed path from any vertex w in an (undirected) cycle to u. Also, for some arc $a = (u, v)$ write $R[a]$ to denote the set of leaves below v. Use $f(N)$ to denote the number of triplets in T consistent with N; and $g(N, Z)$ to denote the number of triplets in T that are consistent with N and that are not of the form $xy|z$ with $z \in Z$ and $x, y \notin Z$. It will become clear later that the definition of g ensures that combining networks that are optimal with respect to g leads to networks optimal with respect to f. The three candidate networks for a certain partition $\pi = (X, Y, Z)$ are denoted N^1_π, N^2_π, N^3_π. For partitions with $Z = \emptyset$, only one candidate network N^2_π is created.

A description of the algorithm is displayed in Algorithm 1. As will be shown below, for $L' \subseteq L$, the computed network $N(L')$ is a network maximizing $f(N)$ over all networks with $L(N) = L'$. In addition, for $Z \subset L' \subseteq L$, the computed network $N^2(L', Z)$ will be shown to maximize $g(N, Z)$ over all level-1 networks N with $L(N) = L'$ that contain an ncr-arc a with $Z = R[a]$. If there are various such networks then the algorithm picks one arbitrarily.

Because the arcs $a = (u, v)$ and $a' = (u', v')$ in lines 9 and 11 are ncr-arcs, we know that [in $N^2(X \cup Z, Z)$ and $N^2(Y \cup Z, Z)$] neither u, nor u', nor one of their ancestors is contained in a cycle. It follows that the newly created cycles do not overlap with any of the original cycles and hence that the constructed networks are indeed level-1 networks. It now also becomes intuitively clear why networks $N^2(X \cup Z, Z)$ and $N^2(Y \cup Z, Z)$ are used that are optimal with respect to g (rather than f). Consider for example the construction of N^1_π from $N^2(X \cup Z, Z)$ and $N^2(Y \cup Z, Z)$ in Fig. 9. It does not matter whether a triplet $x_1x_2|z$ with $x_1, x_2 \in X$ and $z \in Z$ is consistent with $N^2(X \cup Z, Z)$ or not, since such a triplet will be consistent with N^1_π anyhow. Similarly, it does not matter whether triplets of the form $y_1y_2|z$ with $y_1, y_2 \in Y$ and $z \in Z$ are consistent with $N^2(Y \cup Z, Z)$, since the creation of a new cycle as in Figs. 9 and 10, causes all triplets of this form to become consistent with the network.

Correctness of the algorithm is shown by induction on i. The induction basis for $i = 1$ is trivial. For the induction step, consider a leaf set L' with $|L'| > 1$ and assume that the algorithm works correctly for all leaf sets of smaller size. Thus, for each tripartition (X, Y, Z) of L' with $X, Y \neq \emptyset$ the algorithm has correctly computed networks $N(X)$, $N(Y)$, $N^2(X \cup Z, Z)$ and $N^2(Y \cup Z, Z)$. The following lemma shows that the algorithm correctly computes $N^2(L', \bar{Z})$, for all $\bar{Z} \subset L'$. Subsequently, Lemma 6 shows that the algorithm correctly computes $N(L')$.

Lemma 5. *For every $\bar{Z} \neq \emptyset$, the computed network $N^2(L', \bar{Z})$ maximizes $g(N, \bar{Z})$ over all level-1 networks N with $L(N) = L'$ that contain an ncr-arc a with $\bar{Z} = R[a]$.*

Proof. Let N' be a network with $L(N) = L'$ and some ncr-arc a such that $\bar{Z} = R[a]$. We show that $g(N', \bar{Z}) \leq g(N^2(L', \bar{Z}), \bar{Z})$. Because N' contains the ncr-arc a, the root of N' is not in a cycle. Let a_1 and a_2 be the two cut-arcs leaving the root

Algorithm 1. Exact Algorithm for MaxCL-1

1. **for** each $\ell \in L$ **do**
2. Let $N(\ell)$ be a single leaf labeled ℓ.
3. **for** $i = 2 \dots n$ **do**
4. **for** each $L' \subseteq L$ of cardinality i **do**
5. **for** each tripartition $\pi = (X, Y, Z)$ of L' with $X, Y \neq \emptyset$ **do**
6. **if** $Z = \emptyset$ **then**
7. Combine $N(X)$ and $N(Y)$ into a new network N^2_π by adding a new root and connecting it to the roots of $N(X)$ and $N(Y)$.
8. **else**
9. **Construct first candidate.** Combine $N^2(X \cup Z, Z)$ and $N^2(Y \cup Z, Z)$ into a new network N^1_π by creating a "non-skew" cycle as follows. Add a new root and connect it to the roots of $N^2(X \cup Z, Z)$ and $N^2(Y \cup Z, Z)$. Let $a = (u, v)$ and $a' = (u', v')$ be the (unique) ncr-arcs such that $Z = R[a]$ in $N^2(X \cup Z, Z)$ and $Z = R[a']$ in $N^2(Y \cup Z, Z)$. Subdivide a into (u, w) and (w, v), delete v' and all arcs and vertices reachable from v', and add an arc (u', w) (see Fig. 9).
10. **Construct second candidate.** Combine $N(X)$ and $N^2(Y \cup Z, Z)$ into a new network N^2_π by adding a new root and connecting it to the roots of $N(X)$ and $N^2(Y \cup Z, Z)$.
11. **Construct third candidate.** Create N^3_π from N^2_π by creating a "skew" cycle as follows: let $a = (u, v)$ be the (unique) ncr-arc with $Z = R[a]$. Subdivide a into (u, w) and (w, v), add a new root and connect it to the old root and to w (see Fig. 10).
12. Let $N(L')$ be a network that maximizes $f(N)$ over all computed networks N^1_π, N^2_π and N^3_π over all tripartitions π of L'.
13. **for** each $\bar{Z} \subset L'$ **do**
14. Let $N^2(L', \bar{Z})$ be a network that maximizes $g(N^2_\pi, \bar{Z})$ over the networks N^2_π over all tripartitions $\pi = (X, Y, Z)$ of L' with $Z = \bar{Z}$.
15. **output** $N(L)$

such that the leaves in $\bar{Z}$ are reachable from a_2. Let $X' = R[a_1]$ and $Y' = R[a_2] \backslash \bar{Z}$, see Fig. 11(a). Because $N^2(L', \bar{Z})$ maximizes $g(N^2_\pi, \bar{Z})$ over all tripartitions $\pi = (X, Y, Z)$ of L' with $Z = \bar{Z}$, it is certainly at least as good as $N^2_{(X', Y', \bar{Z})}$. That is, $g(N^2(L', \bar{Z}), \bar{Z}) \geq g(N^2_{(X', Y', \bar{Z})}, \bar{Z})$. Write $N^{2\prime}$ as short for $N^2_{(X', Y', \bar{Z})}$. Compare triplets consistent with N', with those consistent with $N^{2\prime}$.

- There are at least as many triplets in $T|_{X'}$ consistent with $N^{2\prime}$ as with N', because $N(X')$ is a subgraph of $N^{2\prime}$ and $N(X')$ maximizes $f(N)$ over all networks N with $L(N) = X'$.
- There are at least as many triplets in $T|_{(Y' \cup \bar{Z})}$ that are not of the form $y_1 y_2 | z$ for $y_1, y_2 \in Y'$ and $z \in \bar{Z}$ that are consistent with $N^{2\prime}$ as with N',

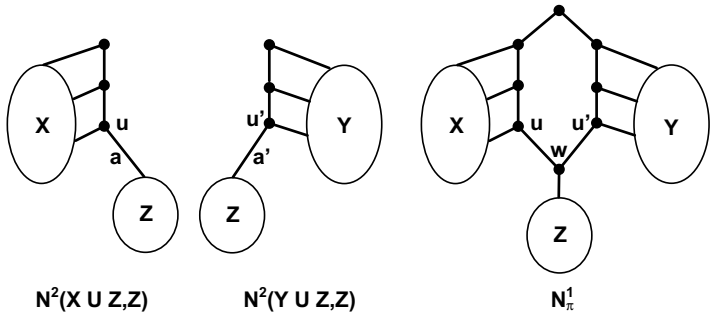


Fig. 9. Construction of N^1_π from $N^2(X \cup Z, Z)$ and $N^2(Y \cup Z, Z)$.

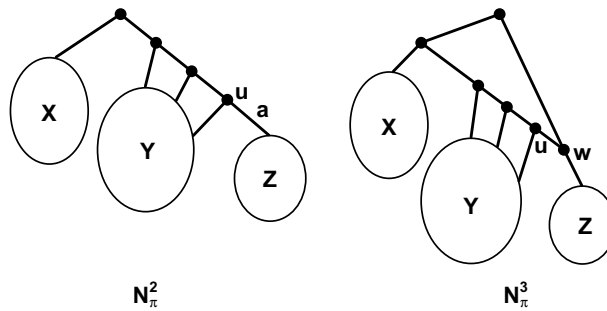


Fig. 10. Construction of N^3_π from N^2_π.

because $N^2(Y' \cup \bar{Z}, \bar{Z})$ is a subgraph of $N^{2\prime}$ and $N^2(Y' \cup \bar{Z}, \bar{Z})$ maximizes $g(N, \bar{Z})$ over all networks N with $L(N) = Y' \cup \bar{Z}$ that contain an ncr-arc a with $\bar{Z} = R[a]$.

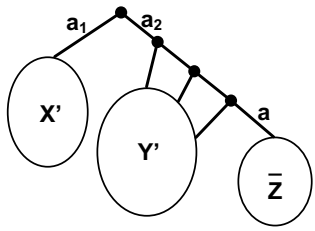
- All triplets of the form $ab|c$ with $a, b \in X'$, $c \in Y' \cup \bar{Z}$ or $a, b \in Y' \cup \bar{Z}$, $c \in X'$ are consistent with both $N^{2\prime}$ and N'.
- All triplets of the form $ab|c$ with $a, c \in X'$, $b \in Y' \cup \bar{Z}$ or $a, c \in Y' \cup \bar{Z}$, $b \in X'$ are consistent with neither $N^{2\prime}$ nor N'.

Thus $g(N', \bar{Z}) \leq g(N^{2\prime}, \bar{Z}) \leq g(N^2(L', \bar{Z}), \bar{Z})$. □

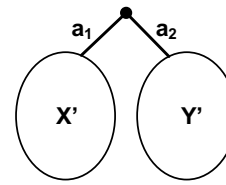
Lemma 6. *The computed network $N(L')$ maximizes $f(N)$ over all level-1 networks N with $L(N) = L'$.*

Proof. For contradiction, suppose that some network $N' \neq N(L')$ with $L(N') = L'$ is consistent with more triplets in T than $N(L')$. Distinguish three cases, depending on the shape of N'.

The first case is that the two arcs leaving the root of N' are cut-arcs a_1 and a_2. Let $X' = R[a_1]$, $Y' = R[a_2]$ and $Z' = \emptyset$, see Fig. 11(b), and compare N' to $N^2_{(X',Y',Z')}$.

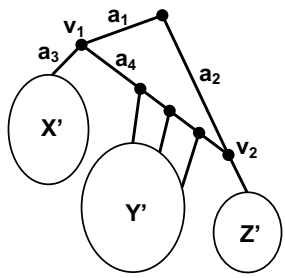


(a) In the proof of Lemma 5

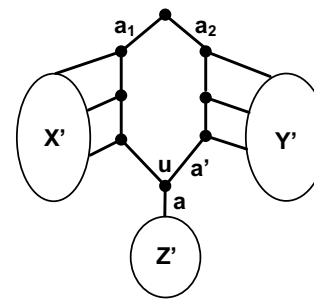


(b) The first case in the proof of Lemma 6

Fig. 11. Networks N' in the proofs of Lemmas 5 and 6.



(a) Skew cycle; the second case in Lemma 6



(b) Non-skew cycle; the third case in Lemma 6

Fig. 12. Networks N' in the proof of Lemma 6.

The latter network is consistent with at least as many triplets from $T|_{X'}$ because it contains $N(X')$ as a subnetwork, and $N(X')$ maximizes $f(N)$ over all networks N with $L(N) = X'$. Similarly, the network $N^2_{(X',Y',Z')}$ is consistent with as least as many triplets from $T|_{Y'}$ as N'. All other triplets are either consistent with both or with none of these networks. Hence $N^2_{(X',Y',Z')}$ is consistent with at least as many triplets as N'. Because $N(L')$ is consistent with at least as many triplets as $N^2_{(X',Y',Z')}$, it follows that $N(L')$ is also consistent with at least as many triplets as N'; a contradiction.

The second case is that one child of the root of N' is a reticulation vertex. Let $a_1 = (r, v_1)$ and $a_2 = (r, v_2)$ be the two arcs leaving the root of N' and suppose that v_2 is a reticulation vertex. Let a_3 and a_4 be the two arcs leaving v_1. Because N' is a level-1 network, one of a_3 and a_4 is a cut-arc, say a_3. Let $X' = R[a_3]$, $Y' = R[a_4] \backslash R[a_2]$ and $Z' = R[a_2]$, see Fig. 12(a). Compare the networks N' and $N^3_{(X',Y',Z')}$ with respect to the number of triplets in T these networks are consistent with. First, consider triplets in $T|_{X'}$: Network $N^3_{(X',Y',Z')}$ is consistent with at least as many of these as N', because it contains $N(X')$ as a subgraph. Second, consider triplets in $T|_{(Y' \cup Z')}$ that are not of the form $y_1 y_2 | z$ for $y_1, y_2 \in Y'$ and $z \in Z'$. We will show that $N^3_{(X',Y',Z')}$ is consistent with at least

as many of these triplets as N'. First recall that $N^3_{(X',Y',Z')}$ contains a subdivision of $N^2(Y' \cup Z', Z')$, which maximizes $g(N, Z')$ over all networks with $L(N) = Y' \cup Z'$ containing an ncr-arc a with $Z' = R[a]$. The network N' does not contain such an ncr-arc, but we will modify it to a network that does contain such an ncr-arc and is consistent with the same number of the considered triplets. Let N'' be the result of removing the arc a_2 from N' (and tidying up the resulting graph). Observe that $g(N'', Z') = g(N', Z')$, and so it follows that $N^3_{(X',Y',Z')}$ is consistent with at least as many of the considered triplets as N'. All other triplets are either consistent with both $N^3_{(X',Y',Z')}$ and N' or with none, since both networks have the structure from Fig. 12(a): only the internal structure inside X', Y' and Z' might be different in the two networks. Hence $N^3_{(X',Y',Z')}$ is consistent with at least as many triplets as N'. Because $N(L')$ is consistent with at least as many triplets as $N^3_{(X',Y',Z')}$ it follows that $N(L')$ is also consistent with at least as many triplets as N'; a contradiction.

The last case is that the two arcs a_1 and a_2 leaving the root of N' are not cut-arcs and are also not leading to reticulation vertices. Let $X' = R[a_1] \backslash R[a_2]$, $Y' = R[a_2] \backslash R[a_1]$ and $Z' = R[a_1] \cap R[a_2]$, see Fig. 12(b). Compare the networks N' and $N^1_{(X',Y',Z')}$, with respect to the number of triplets in T these networks are consistent with. First, consider triplets in $T|_{(X' \cup Z')}$ that are not of the form $x_1x_2|z$ for $x_1, x_2 \in X'$ and $z \in Z'$. We will show that $N^1_{(X',Y',Z')}$ is consistent with at least as many of these triplets as N'. Recall that $N^1_{(X',Y',Z')}$ contains a subdivision of $N^2(X' \cup Z', Z')$, which maximizes $g(N, Z')$ over all networks with $L(N) = X' \cup Z'$ containing an ncr-arc a with $Z' = R[a]$. The network N' does not contain such an ncr-arc, but we will modify it to a network that does contain such an ncr-arc and is consistent with the same number of the considered triplets. Let $a = (u, v)$ be the cut-arc in N' with $Z' = R[a]$, and let a' be the arc that leads to u and is reachable from a_2. Let N'' be the result of removing arc a' from N' (and tidying up the resulting graph). Now N'' is consistent with the same number of the considered triplets as N', and so it follows that $N^1_{(X',Y',Z')}$ is consistent with at least as many of the considered triplets as N'. In a similar way, it follows that $N^1_{(X',Y',Z')}$ is consistent with as least as many triplets in $T|_{(Y' \cup Z')}$ that are not of the form $y_1y_2|z$ for $y_1, y_2 \in Y'$ and $z \in Z'$). All other triplets are either consistent with both networks or with none. Hence $N^1_{(X',Y',Z')}$ is consistent with at least as many triplets as N'. Because $N(L')$ is consistent with at least as many triplets as $N^1_{(X',Y',Z')}$ it follows that $N(L')$ is also consistent with at least as many triplets as N'; a contradiction. □

Theorem 4. *Given a set T of m triplets over n leaves, a level-1 network consistent with a maximum number of triplets in T can be constructed in $O(m4^n)$ time and $O(n3^n)$ space.*

Proof. Correctness of the algorithm follows from the above. To achieve a small polynomial factor in the complexity, we use dynamic programming to compute

the optimal value of the solution as well as the partitions we have to choose in each step. Then a traceback algorithm constructs a network consistent with the maximum number of triplets. To be precise, the dynamic programming algorithm finds, for all $L' \subseteq L$, the maximum number $\hat{f}(L')$ of triplets in T consistent with a level-1 network with leaves $L' \subseteq L$. It also computes, for all $\bar{Z} \subset L'$, the maximum value $\hat{g}(L', \bar{Z})$ of $g(N, \bar{Z})$ over all level-1 networks N with leaves L' that contain an ncr-arc a with $\bar{Z} = R[a]$. The algorithm loops through all the subsets $L' \subseteq L$ from small to large and considers all tripartitions $\pi = (X, Y, Z)$ of L'. For each such partition, the values $\hat{f}(X)$, $\hat{f}(Y)$, $\hat{f}(Z)$, $\hat{g}(X \cup Z, Z)$ and $\hat{g}(Y \cup Z, Z)$ are readily available from previous iterations. To compute the values $\hat{f}(L')$ and $\hat{g}(L', Z)$ it only remains to count certain triplets in T, whose consistency with a network only depends on the tripartition (X, Y, Z) and the network type (N^1_π, N^2_π or N^3_π). This can be done by first checking membership of X, Y and Z for each leaf in L' (in $O(n)$ time) and then looping through all triplets only once. Hence this counting can be done in $O(n + m) = O(m)$ time. The algorithm's overall running time is thus bounded by $O(m) \sum_{\ell=1}^{n} \binom{n}{\ell} O(3^\ell) = O(m4^n)$.

For each leaf set $L' \subseteq L$, store the optimal tripartition and the optimal type of network (N^1_π, N^2_π or N^3_π). In addition, store an optimal bipartition for all $L' \subseteq L$ and $\bar{Z} \subset L'$. This yields a total space complexity of $O(n3^n)$.

Once the values $\hat{f}(L')$ and $\hat{g}(L', \bar{Z})$ have been computed and all optimal tripartitions and bipartitions have been stored, a level-1 network N consistent with $\hat{f}(L)$ many triplets can be constructed by traceback, in polynomial time. □

The algorithm can be easily adapted to the case of weighted triplets. Specifically: in those parts of the algorithm where we currently count the number of triplets with a certain property, we should instead compute the total weight of triplets with that property. The running time is not affected by the weights on the triplets themselves.

7. Open Problems

The obvious question to ask, is whether the $O(m4^n)$ running time of our exact algorithm in Sec. 6 can be improved. The same question can be asked about the $O(3^n(n^2 + m))$ algorithm[15] for maximum consistent trees. It would also be interesting to extend the exact approach to the construction of level-2 networks, provided that reasonable running times can be achieved.

Positive results for level-3 and higher networks have so far remained out of reach. In light of 65 simple level-3 generators,[24] we fear that algorithms for constructing level-3 networks (and higher) will almost certainly not be possible by using approaches similar to the ones in Sec. 6. A similar statement holds for the dense level-2 case,[11] since the devised algorithms explicitly distinguish between the structures of different level-k generators. Tantalisingly, however, it remains a possibility that for each $k \geq 0$ it is polynomial-time solvable to determine whether there is a level-k network consistent with a dense set of input triplets.

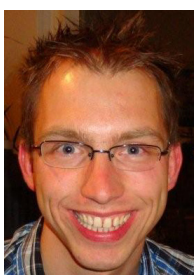
Approximability of the MaxCL-k problem needs to be further explored. APX-completeness of MaxCL-0 is known,[17] hence no *Polynomial Time Approximation Scheme* for MaxCL-0 is possible unless P = NP. It would be interesting to extend this result to $k > 0$. On the other hand, the best known approximation ratios are $\frac{1}{3}$ for MaxCL-0[16] and 0.48 for MaxCL-1,[17] leaving (potentially) much room for improvement.

From a more practical point of view, it is worthwhile to study the actual level of real evolutionary histories. This will tell for which values of k it remains important to design algorithms that construct level-k networks.

References

1. Huson DH, Bryant D, Application of phylogenetic networks in evolutionary studies, *Mol Biol Evol* **23**(2):254–267, 2006.
2. Gusfield D, Eddhu S, Langley C, Optimal, efficient reconstruction of phylogenetic networks with constrained recombination, *J Bioinform Comput Biol* **2**:173–213, 2004.
3. Nakhleh L, Warnow T, Linder CR, St. John K, Reconstructing reticulate evolution in species — theory and practice, *J Comput Biol* **12**(6):796–811, 2005.
4. Jansson J, Nguyen B, Sung WK, Algorithms for combining rooted triplets into a galled phylogenetic network, *SIAM J Comput* **35**(5):1098–1121, 2006.
5. Choy C, Jansson J, Sadakane K, Sung WK, Computing the maximum agreement of phylogenetic networks, *Theor Comput Sci* **335**(1):93–107, 2005.
6. Holder M, Lewis PO, Phylogeny estimation: Traditional and bayesian approaches, *Nat Rev Genet* **4**:275–284, 2003.
7. St. John K, Warnow T, Moret BME, Vawter L, Performance study of phylogenetic methods: (unweighted) quartet methods and neighbor-joining, *J Algorithm* **48**(1):173–193, 2003.
8. Snir S, Warnow T, Rao S, Short quartet puzzling: A new quartet-based phylogeny reconstruction algorithm, *J Comput Biol* **15**(1):91–103, 2008.
9. Aho AV, Sagiv Y, Szymanski TG, Ullman JD, Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions, *SIAM J Comput* **10**(3):405–421, 1981.
10. Steel M, The complexity of reconstructing trees from qualitative characters and subtrees, *J Classif* **9**(1):91–116, 1992.
11. van Iersel LJJ, Keijsper JCM, Kelk SM, Stougie L, Hagen F, Boekhout T, Constructing level-2 phylogenetic networks from triplets, *Research in Comp Mol Biol (RECOMB 2008)*, LNCS 4955, pp. 450–462, 2008.
12. van Iersel LJJ, Keijsper JCM, Kelk SM, Stougie L, Constructing level-2 phylogenetic networks from triplets, arXiv:0707.2890v1 [p-bio.PE], July 2007.
13. Bryant D, *Building Trees, Hunting for Trees, and Comparing Trees: Theory and Methods in Phylogenetic Analysis*, Thesis, University of Canterbury, Christchurch, New Zealand, 1997.
14. Jansson J, On the complexity of inferring rooted evolutionary trees, *Brazilian Symposium on Graphs, Algorithms and Combinatorics* in *Electron Notes Discrete Math.* 7, p. 4, 2001.
15. Wu BY, Constructing the maximum consensus tree from rooted triples, *J Comb Optim* **8**(1):29–39, 2004.
16. Gąsieniec L, Jansson J, Lingas A, Östlin A, On the complexity of constructing evolutionary trees, *J Comb Optim* **3**(2–3):183–197, 1999.

17. Byrka J, Gawrychowski P, Huber KT, Kelk SM, Worst-case optimal approximation algorithms for maximizing triplet consistency within phylogenetic networks, to appear in *J Discrete Algorithm* doi:10.1016/j.jda.2009.01.004.
18. Semple C, Steel M, A supertree method for rooted trees, *Discrete Appl Math* **105**(1–3):147–158, 2000.
19. Page RDM, Modified Mincut Supertrees, *Proc Workshop on Algorithms in Bioinformatics (WABI 2002)*, LNCS 2452, pp. 537–551, 2002.
20. Snir S, Rao S, Using Max Cut to enhance rooted trees consistency, *IEEE/ACM Trans Comp Biol Bioinform* **3**(4):323–333, 2006.
21. Baum BR, Combining trees as a way of combining data sets for phylogenetic inference, *Taxon* **4**:3–10, 1992.
22. Ragan MA, Matrix representation in reconstructing phylogenetic relationships among the eukaryotes, *Biosystems* **28**:47–55, 1992.
23. Sanderson MJ, Purvis A, Henze C, Phylogenetic supertrees: Assembling the trees of life, *Trends Ecol Evol* **13**:105–109, 1998.
24. Kelk SM, Level-3 generators, 2008, ⟨http://homepages.cwi.nl/∼kelk/lev3gen/⟩.
25. Jansson J, Sung WK, Inferring a level-1 phylogenetic network from a dense set of rooted triplets, *Theor Comput Sci* **363**(1):60–68, 2006.
26. Garey MR, Johnson DS, *Computers and Intractability*, W. H. Freeman and Co., San Francisco, California, 1979.
27. Alon N, Ranking tournaments, *SIAM J Discrete Math* **20**(1):137–142, 2006.
28. Charbit P, Thomassé S, Yeo A, The minimum feedback arc set problem is NP-hard for tournaments, *Combin Probab Comput* **16**(1):1–4, 2007.



Leo van Iersel received his M.S. degree in Applied Mathematics from the Universiteit Twente in The Netherlands, in 2004. He obtained his Ph.D. degree from the Technische Universiteit Eindhoven, also in the Netherlands, in January 2009. He is currently a postdoctoral researcher at the University of Canterbury (Christchurch, New Zealand). His research is concerned with the design of algorithms for phylogenetics and other areas of computational biology.

Steven Kelk is a postdoc at the Centrum voor Wiskunde en Informatica in Amsterdam, working on applications of combinatorics to problems arising in computational biology.

His current research interests are phylogenetic networks, elementary flux mode analysis, approximation algorithms and computational complexity.

Matthias Mnich is a Ph.D. student at Eindhoven University of Technology, where he works on the design of exact exponential-time and parameterized algorithms for combinatorial optimization problems.

His research interests are phylogenetic networks, parameterized complexity theory and extremal graph theory.