Algorithmica (2011) 60: 207–235
DOI 10.1007/s00453-009-9333-0

Constructing the Simplest Possible Phylogenetic Network from Triplets

Leo van Iersel · Steven Kelk

Received: 6 August 2008 / Accepted: 16 June 2009 / Published online: 7 July 2009
© The Author(s) 2009. This article is published with open access at Springerlink.com

Abstract A phylogenetic network is a directed acyclic graph that visualizes an evolutionary history containing so-called *reticulations* such as recombinations, hybridizations or lateral gene transfers. Here we consider the construction of a simplest possible phylogenetic network consistent with an input set T, where T contains at least one phylogenetic tree on three leaves (a *triplet*) for each combination of three taxa. To quantify the complexity of a network we consider both the total number of reticulations and the number of reticulations per biconnected component, called the *level* of the network. We give polynomial-time algorithms for constructing a level-1 respectively a level-2 network that contains a minimum number of reticulations and is consistent with T (if such a network exists). In addition, we show that if T is precisely equal to the set of triplets consistent with some network, then we can construct such a network with smallest possible level in time $O(|T|^{k+1})$, if k is a fixed upper bound on the level of the network.

Keywords Phylogenetics · Polynomial-time algorithm · Phylogenetic networks · Triplets · Minimising reticulations

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

L. van Iersel (✉)
Department of Mathematics and Statistics, University of Canterbury, Private Bag 4800, Christchurch, New Zealand
e-mail: l.j.j.v.iersel@gmail.com

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

1 Introduction

One of the ultimate goals in computational biology is to create methods that can reconstruct evolutionary histories from biological data of currently living organisms. The immense complexity of biological evolution makes this an extremely challenging task. This has motivated researchers to focus first on the simplest possible shape of evolution, the tree-shape [21]. Now that treelike evolution has been well studied, a logical next step is to also consider slightly more complicated evolutionary scenarios, gradually extending the complexity that our models can describe. At the same time, we also wish to take into account the parsimony principle, which tells us that amongst all equally good explanations of our data, one prefers the simplest one.

For a set of taxa (e.g. species or strains), a phylogenetic tree describes (a hypothesis of) the evolution that these taxa have undergone. The taxa form the leaves of the tree while the internal vertices represent events of genetic divergence: one incoming branch splits into two (or more) outgoing branches.

Phylogenetic networks form an extension to this model where it is also possible that two branches combine into one new branch. We call such an event a *reticulation*, which can model any kind of non-treelike (also called "reticulate") evolutionary process such as recombination, hybridization or lateral gene transfer. In addition, reticulations can also be used to display different possible (treelike) evolutions in one figure. In recent years, there has emerged enormous interest in phylogenetic networks and their application [2, 12, 19, 21, 22].

This model of a phylogenetic network allows for many different degrees of complexity, ranging from networks that are equal, or almost equal, to a tree to complex webs of frequently diverging and recombining lineages. Therefore we consider two different measures for the complexity of a network. The first of these measures is the total number of reticulations in the network. Secondly, we consider the *level* of the network, which is an upper bound on the number of reticulations per non-treelike part (i.e. biconnected component) of the network. In this paper we consider two different approaches for constructing networks that are as simple as possible. The first approach minimizes the total number of reticulations for a fixed level (of at most two) and the second approach minimizes both the level and the total number of reticulations, but under more heavy restrictions on the input.

Level-k phylogenetic networks were first introduced by Choy et al. [7] and further studied by various authors [15, 16, 18]. Gusfield et al. gave a biological justification for level-1 networks (which they call "galled trees") [8]. Minimizing reticulations has been very well studied in the framework where the input consists of (binary) sequences [9, 23, 24]. For example, Wang et al. considered the problem of finding a "perfect phylogeny" with a minimum number of reticulations and showed that this problem is NP-hard [25]. Gusfield et al. showed that this problem can be solved in polynomial time if restricted to level-1 networks [8]. There are also several results known already about the version of the problem where the input consists of a set of trees and the objective is to construct a network that is "consistent" with each of the input trees. Baroni et al. give bounds on the minimum number or reticulations needed to combine two trees [3] and Bordewich et al. showed that it is APX-hard to compute this minimum number exactly [4]. However, there exists an exact algorithm [5] that

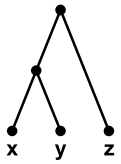

Fig. 1 One of the three possible triplets on the leaves x, y and z. Note that, as with all figures in this article, all arcs are directed downwards

runs reasonably fast in many practical situations. If restricted to level-1 networks, the problem becomes polynomial-time solvable even if there are more than two input trees [13].

In this paper we also consider input sets consisting of trees, but restrict ourselves to small trees with three leaves each, called *triplets*. See Fig. 1 for an example. Triplets can for example be constructed by existing methods, such as Maximum Parsimony or Maximum Likelihood, that work accurately and fast for small numbers of taxa. Triplet-based methods have also been well-studied. Aho et al. [1] gave a polynomial-time algorithm to construct a tree from triplets if there exists a tree that is consistent with all input triplets. Jansson et al. [17] showed that the same is possible for level-1 networks if the input triplet set is *dense*, i.e. if there is a triplet for any set of three taxa. Van Iersel et al. further extended this result to level-2 networks [15]. From non-dense triplet sets it is NP-hard to construct level-k networks for any $k \geq 1$ [16, 17]. From the proof of this result also follows directly that it is NP-hard to find a network consistent with a non-dense triplet set that contains a minimum number of reticulations.[1] It is unknown whether this problem becomes easier if the input triplet set is dense.

In the first part of this paper we consider fixed-level networks and aim to minimize the total number of reticulations in these networks. In Sect. 3 we give a polynomial-time algorithm that constructs a level-1 network consistent with a dense triplet set T (if such a network exists) and minimizes the total number of reticulations over all such networks. This gives an extension to the algorithm by Jansson et al. [17], which can also reconstruct level-1 networks, but does not minimize the number of reticulations. To illustrate this we give in Sect. 2 an example dense triplet set on n leaves for which the algorithm in [17] (and the ones in [15] and [18]) creates a level-1 network with $\frac{n-1}{2}$ reticulations. However, a level-1 network with just one reticulation is also possible and our algorithm MARLON is able to find that network. We have implemented MARLON, tested it and made it publicly available [20]. The worst case running time of the algorithm is $O(n^5)$ for n leaves (and hence $O(|T|^{\frac{5}{3}})$ with $|T|$ the input size).

In Sect. 4 we further extend this approach by giving an algorithm that even constructs a level-2 network consistent with a dense triplet set (if one exists) and again minimizes the total number of reticulations over all such networks. This means that if the level is at most two, we can minimize both the level and the total number of reticulations, giving priority to the criterion that we find most important. The running time is $O(n^9)$ (and thus $O(|T|^3)$). This extends recent results [15] in which an algorithm is described that also constructs level-2 networks, but does not minimize the number of reticulations in such networks.

[1] This follows from the proof of Theorem 7 in [17], since only one reticulation is used in their reduction.

While previous work [14–18] considered the construction of level-k networks for $k \leq 2$, Sect. 5 of this paper considers the case $k > 2$. This is an even more challenging problem, even without minimizing the total number of reticulations. Given a dense set of triplets, it is a major open problem whether one can construct a minimum level phylogenetic network consistent with these triplets in polynomial time. Moreover, it is not even known whether it is possible to construct a level-3 network consistent with a dense input triplet set in polynomial time. In Sect. 5 of this paper we show some significant progress in this direction. As a first step we consider the restriction to "simple" networks, i.e. networks that contain just one nontrivial biconnected component. We show how to construct, in $O(|T|^{k+1})$ time, a minimum level simple network with level at most k from a dense input triplet set (for fixed k).

Subsequently we show that this can be used to also generate general level-k networks if we put an extra restriction on the quality of the input triplets. Namely, we assume that the input set contains exactly all triplets consistent with some network. If that is the case then our algorithm can find such a network that simultaneously minimizes level *and* the total number of reticulations used. The fact that in this case optimal solutions for both measures coincide, is an interesting consequence of the restriction on the input triplets. The algorithm runs in polynomial time $O(|T|^{k+1})$ if the upper bound k on the level of the network is fixed. (For $k = 1, 2$ we can use existing, optimized simple level-1 and simple level-2 algorithms as subroutines to obtain improved running times of $O(|T|)$ and $O(|T|^{\frac{8}{3}})$ respectively.) This result constitutes an important step forward in the analysis of level-k networks, since it provides the first positive result that can be used for all levels k.

2 Preliminaries

A *phylogenetic network* (*network* for short) is defined as a directed acyclic graph in which exactly one 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 *reticulations* for short) or indegree 1 and outdegree 0 (*leaves*), where the leaves are distinctly labelled. A phylogenetic network without reticulations is called a *phylogenetic tree*.

A directed acyclic graph is *connected* (also called "weakly connected") if there is an undirected path between any two vertices and *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. To avoid "redundant" networks we only allow networks in which every nontrivial biconnected component has at least three outgoing arcs. We call an arc $a = (u, v)$ of a network N a *cut-arc* if its removal disconnects N and call it *trivial* if v is a leaf.

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

A level-k network that contains no nontrivial cut-arcs and is not a level-$(k-1)$ network is called a *simple* level-k network.[2] Informally, a simple network thus consists of a nontrivial biconnected component with leaves "hanging" of it.

A *triplet* $xy|z$ is a phylogenetic tree on the leaves x, y and z such that the lowest common ancestor of x and y is a proper descendant of the lowest common ancestor of x and z. The triplet $xy|z$ is displayed in Fig. 1. Denote the set of leaves in a network N by L_N. For any set T of triplets define $L(T) = \bigcup_{t\in T} L_t$ and let $n = |L(T)|$. A set T of triplets is called *dense* if for each $\{x, y, z\} \subseteq L(T)$ at least one of $xy|z$, $xz|y$ and $yz|x$ belongs to T.

For a set of triplets T and a set of leaves $L' \subseteq L(T)$, we denote by $T|L'$ the triplets $t \in T$ with $L_t \subseteq L'$. Furthermore, if $\mathcal{C} = \{S_1, \ldots, S_q\}$ is a collection of leaf-sets we use $T\nabla\mathcal{C}$ to denote the *induced* set of triplets $S_iS_j|S_k$ such that there exist $x \in S_i$, $y \in S_j$, $z \in S_k$ with $xy|z \in T$ and i, j and k all distinct.

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 $v \to z$.

The above definitions enable us to give a formal description of the problems we consider.

Problem: Minimum Reticulation Level-k network on dense triplet sets (DMRL-k).
Input: dense set of triplets T.
Output: level-k network N that is consistent with T (if such a network exists) and has a minimum number of reticulations over all such networks.

A feasible solution to DMRL-1 can be found by the algorithm in [17] or [18] and the algorithm in [15] finds a feasible solution to DMRL-2. To show that these algorithms do not always minimize the number of reticulations, consider a triplet set over an odd number n of leaves, labelled $1, \ldots, n$, containing all triplets $ab|c$ with $a, b > c$ and the triplets $a(a+1)|n$ with $a = 1, 3, \ldots, n-2$. The aforementioned algorithms find for this input set a level-1 network with $\frac{n-1}{2}$ reticulations. However, a level-1 network with just one reticulation is also possible and our algorithm MARLON, introduced shortly, is able to find that network. See Fig. 2 for an example for $n = 9$.

Given a network N let $T(N)$ denote the set of all triplets consistent with N. We say that a network N *reflects* a triplet set T if $T(N) = T$. If, for a triplet set T, there exists a network N that reflects it, we say that T is *reflective*. The second problem we consider is thus the following:

Problem: MIN-REFLECT-k.
Input: set of triplets T.
Output: level-k network N that reflects T (if such a network exists) and, ranging over all such networks, minimizes both the level and the number of reticulations used.

[2] This definition is equivalent to Definition 4 in [15] by Lemma 2 in [15].

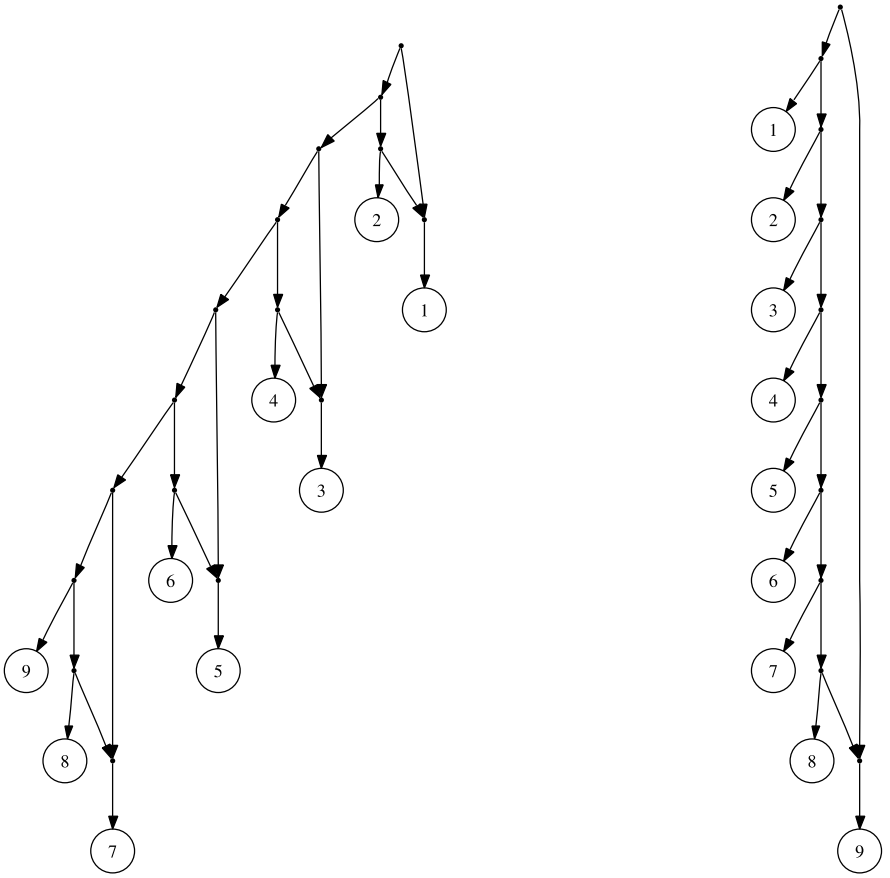

Fig. 2 Example of a situation where previous algorithms (by Jansson et al. [17] and Van Iersel et al. [15]) construct a network like the one to the left with $\frac{n-1}{2}$ reticulations, while MARLON constructs the network to the right, with just one reticulation

The above problem might at first glance seem strangely formulated because, in general, minimizing level and minimizing number of reticulations are two distinct optimization criteria. However, in the case of reflectivity it will turn out that any solution that minimizes the number of reticulations also minimizes level.

Note that this problem is closely related to the mixed triplets problem (MT) studied in [10], which asks for a phylogenetic network consistent with an input triplet set T and *not* consistent with another input triplet set F. Namely, MIN-REFLECT-k is a special case of MT restricted to level-k networks where the set F of forbidden triplets contains all triplets that are not in T.

To describe our algorithms we need to introduce some more definitions. We say that a cut-arc is a *highest cut-arc* if it is not reachable from any other cut-arc. We call a cycle containing the root a *highest cycle* and a reticulation in such a cycle a *highest reticulation*. We say that a leaf x is *below* an arc (u, v) (and *below* vertex v) if x is

reachable from v. In the next section we will frequently use the set $BHR(N)$, which denotes the set of leaves in network N that is below a highest reticulation.

A subset S of the leaves is an *SN-set* (of triplet set T) if there is no triplet $xy|z$ in T with $x, z \in S$, $y \notin S$. An SN-set is called *nontrivial* if it does not contain all leaves. Furthermore, we say that an SN-set S is *maximal* (under restriction X) if there is no nontrivial SN-set (satisfying restriction X) that is a strict superset of S. An SN-set of size one is called a *singleton* SN-set.

Any two SN-sets of a dense triplet set are either disjoint or one is included in the other [18, Lemma 8], which leads to the following definition. The *SN-tree* is a directed tree with vertices with outdegree greater or equal to two, such that the SN-sets of T correspond exactly to the sets of leaves reachable from a vertex of the SN-tree. It follows that there are at most $2(n-1)$ nontrivial SN-sets of a dense triplet set T. All these SN-sets can be found by constructing the SN-tree in $O(n^3)$ time [17]. If a network is consistent with a dense triplet set T, then the set of leaves S below any cut-arc is always an SN-set, since triplets of the form $xy|z$ with $x, z \in S$, $y \notin S$, are not consistent with such a network. Furthermore, each maximal SN-set is equal to the union of leaves below one or more highest cut-arcs [14, Lemma 5].

3 Constructing a Level-1 Network with a Minimum Number of Reticulations

We propose the following dynamic programming algorithm for solving DMRL-1. The algorithm considers all SN-sets from small to large and computes an optimal solution N_S for each SN-set S, based on the optimal solutions for included SN-sets. The algorithm considers both the case where the root of N_S is contained in a cycle and the case where there are two cut-arcs leaving the root. In the latter case there are two SN-sets S_1 and S_2 that are maximal under the restriction that they are a subset of S. If this is the case then the algorithm constructs a candidate for N_S by creating a root connected to the roots of N_{S_1} and N_{S_2}.

The other possibility is that the root of N_S is contained in some cycle. For this case the algorithm tries each SN-set as $BHR(N_S)$: the set of leaves below the highest reticulation. The sets of leaves below other highest cut-arcs can then be found using the property of optimal level-1 networks outlined in Lemma 1. Subsequently, an induced set of triplets is computed, where each set of leaves below a highest cut-arc is replaced by a single meta-leaf. A candidate network is constructed by computing a simple level-1 network and replacing each meta-leaf S_i by an optimal network N_{S_i} for the corresponding subset of the leaves. The optimal network N_S is then the network with a minimum number of reticulations over all candidate networks.

A structured description of the computations is in Algorithm 1. We use $f(L')$ to denote the minimum number of reticulations in any level-1 network consistent with $T|L'$. In addition, $g(L', S')$ denotes the minimum number of reticulations in any level-1 network consistent with $T|L'$ with $BHR(N) = S'$. The algorithm first computes the optimal number of reticulations. Then a network with this number of reticulations is constructed using backtracking.

To show that the algorithm indeed computes an optimal solution we need the following crucial property of optimal level-1 networks.

Algorithm 1 MARLON (Minimum Amount of Reticulation Level One Network)

compute the set SN of SN-sets of T
for $i = 1 \ldots n$ **do**
  for each S in SN of cardinality i **do**
    for each $S' \in SN$ with $S' \subset S$ **do**
      let $\mathcal{C}$ contain S' and all SN-sets that are maximal under the restriction that they are a subset of S and do not contain S'
      if $T \nabla \mathcal{C}$ is consistent with a simple level-1 network **then**
        $g(S, S') := 1 + \sum_{X \in \mathcal{C}} f(X)$
    if there are exactly two SN-sets $S_1, S_2 \in SN$ that are maximal under the restriction that they are a strict subset of S **then**
      $g(S, \emptyset) := f(S_1) + f(S_2)$ $(\mathcal{C} := \{S_1, S_2\})$
    $f(S) := \min g(S, S')$ over all computed values of $g(S, \cdot)$
    store the optimal $\mathcal{C}$ and the corresponding simple level-1 network
construct an optimal network by backtracking.

Lemma 1 *If there exists a solution to DMRL*-1, *then there also exists an optimal solution* N, *where the sets of leaves below highest cut-arcs equal either* (i) $BHR(N)$ *and the SN-sets that are maximal under the restriction that they do not contain* $BHR(N)$, *or* (ii) *the maximal SN-sets* (*if* $BHR(N) = \emptyset$).

Proof If $BHR(N) = \emptyset$ then there are two highest cut-arcs and the sets below them are the maximal SN-sets. Otherwise, the root of N is part of a cycle. We prove the following. □

Claim 1 Let S be an SN-set. Either S equals a (sub)set of the leaves below a highest cut-arc or there exists a directed path P ending in the highest reticulation or in one of its parents, such that S equals the set of leaves that are below a highest cut-arc with its tail on P.

Proof Assume that an SN-set S does not equal a (sub)set of the leaves below a highest cut-arc. It follows that S equals the union of sets of leaves below several highest cut-arcs. Indeed, if there are leaves $x, z \in S$ below distinct highest cut-arcs then for any leaf y below any of these cut-arcs holds that $xy|z \in T$ and hence that $y \in S$. Now observe that no two leaves in S have the root as their lowest common ancestor, since this would imply that *all* leaves are in S. It now follows that there exists a directed path P on the highest cycle such that all leaves in S are below a highest cut-arc with its tail on P. Let P be a minimal such path. We now argue that all leaves below a highest cut-arc with tail on P are in S. If this would not be the case, then there would be leaves x, z, y below highest cut-arcs with tails respectively p_1, p_2, p_3 that are on P (in this order) with $x, y \in S$ and $z \notin S$. However, this would lead to a contradiction because then the triplet $xy|z$ is not consistent with N, whilst $yz|x$ and $xz|y$ are not in T since S is an SN-set. It remains to prove that P ends in either the highest reticulation or in one of its parents. Assume that this is not true, then there exists a vertex v on the interior of a path from the last vertex of P to the highest

reticulation. Consider some leaf $z \notin S$ below the highest cut-arc with v as tail and some leaves $x, y \in S$ below distinct highest cut-arcs with tails on P. Then this again leads to a contradiction because $xy|z$ is not consistent with N. □

To prove the lemma, consider a maximal SN-set S that is not equal to the set of leaves below a single highest cut-arc. By the maximality of S, it cannot equal a strict subset of the leaves below a highest cut-arc. Thus, from the above claim it follows that there exists a path P such that S equals the set of leaves that are below a highest cut-arc with its tail on P. First suppose that P ends in a parent of the highest reticulation. In this case we can modify the network by putting S below a single cut-arc, without increasing the number of reticulations. To be precise, if p and p' are the first and last vertex of P respectively and r is the highest reticulation, then we subdivide the arc entering p by a new vertex v, add a new arc (v, r), remove the arc (p', r) and suppress p', since it now has indegree and outdegree both equal to one. It is not too difficult to see that the resulting network is still consistent with T.

Now suppose that P ends in the highest reticulation. The sets of leaves below highest cut-arcs are all SN-sets (as is always the case). One of these sets is equal to $BHR(N)$. Suppose that another such set X is *not* maximal under the restriction that it does not contain $BHR(N)$. Then X is a strict subset of a nontrivial SN-set S' that is maximal under the restriction that it does not contain $BHR(N)$. We apply Claim 1 to S'. Observe that S' cannot be equal to a (sub)set of the leaves below a highest cut-arc, since it is a strict superset of X. Thus, S' equals the set of leaves that are below a highest cut-arc with a tail on a path P' on the highest cycle. Moreover, since S' does not contain $BHR(N)$, P' does not end in the highest reticulation, but in one of its parents. Thus, the procedure from the previous paragraph can be used to put S' below a highest cut-arc.

The lemma now follows from the following. If there exists a solution to DMRL-1, then there exists an optimal solution to DMRL-1. After applying the modifications described above to this optimal solution, for each maximal SN-set S, the sets of leaves below highest cut-arcs in the resulting network N' are indeed equal to $BHR(N')$ and the SN-sets that are maximal under the restriction that they do not contain $BHR(N')$.

An example is given in Fig. 3. In the network on the left one maximal SN-set equals the set of leaves below the grey path. In the middle is the same network, but

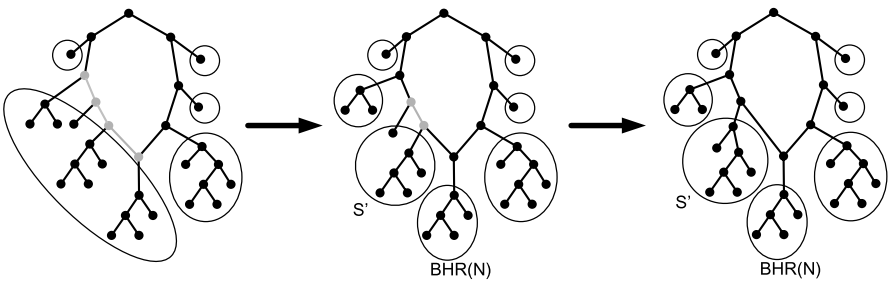

Fig. 3 Visualisation of the proof of Lemma 1. From the maximal SN-sets (*encircled* in the network on the *left*) to the sets of leaves below highest cut-arcs (*encircled* in the network on the *right*). Remember that all arcs are directed downwards

now we encircled $BHR(N)$ and the SN-sets that are maximal under the restriction that they do not contain $BHR(N)$. There is still an SN-set (S') below a path on the cycle (again in grey). However, in this case the network can be modified by putting S' below a single cut-arc, without increasing the number of reticulations. This gives the network to the right, where the sets of leaves below highest cut-arcs are indeed equal to $BHR(N)$ and the SN-sets that are maximal under the restriction that they do not contain $BHR(N)$.

Theorem 1 *Given a dense set of triplets T, algorithm MARLON constructs a level-1 network that is consistent with T (if such a network exists) and has a minimum number of reticulations in $O(n^5)$ time.*

Proof The proof is by induction on the size i of S. Suppose that N is an optimal level-1 network consistent with $T|S$. If $BHR(N) = \emptyset$ then the sets of leaves below highest cut-arcs are the two maximal SN-sets S_1 and S_2. In this case $f(S)$ can be computed by adding up the $f(S_1)$ and $f(S_2)$. Otherwise, it follows from Lemma 1 and the observation that $BHR(N)$ has to be an SN-set, that at some iteration the algorithm will consider the set $\mathcal{C}$ equal to the sets of leaves below the highest cut-arcs of N. In this case the number of reticulations can be computed by adding one to the sum of the values $f(X)$ over all $X \in \mathcal{C}$. This is because the network N consists of a (highest) cycle, connected to optimal networks for the different $X \in \mathcal{C}$. By induction, all values of $f(X)$ for $|X| < i$ have been computed correctly and correctness of the algorithm follows. The number of SN-sets is $O(n)$ because any two SN-sets are either disjoint or one is included in the other [18, Lemma 8]. These SN-sets can be found in $O(n^3)$ time by computing the SN-tree [17]. Simple level-1 networks can be found in $O(n^3)$ time [17] and $T\nabla\mathcal{C}$ can be computed in $O(n^3)$ time. These computations are repeated $O(n^2)$ times: for all $S \in SN$ and all $S' \in SN$ with $S' \subset S$. Therefore, the total running time is $O(n^5)$. □

In Fig. 4 we present an example of a network constructed by MARLON. This network (which has 80 leaves and 13 reticulations) could be constructed by MARLON in less than six minutes on a Pentium IV 3 GHz PC with 1 GB of RAM.

4 Constructing a Level-2 Network with a Minimum Number of Reticulations

This section extends the approach from Sect. 3 to level-2 networks. We describe how one can find a level-2 network consistent with a dense input triplet set containing a minimum number of reticulations, or decide that such a network does not exist.

The general structure of the algorithm is the same as in the level-1 case. We loop though all SN-sets S from small to large and compute an optimal solution N_S for that SN-set, based on previously computed optimal solutions for included SN-sets. For each SN-set we still consider, like in the level-1 case, the possibility that there are two cut-arcs leaving the root of N_S and the possibility that this root is in a biconnected component with one reticulation. However, now we also consider a third possibility, that the root of N_S is in a biconnected component containing two reticulations.

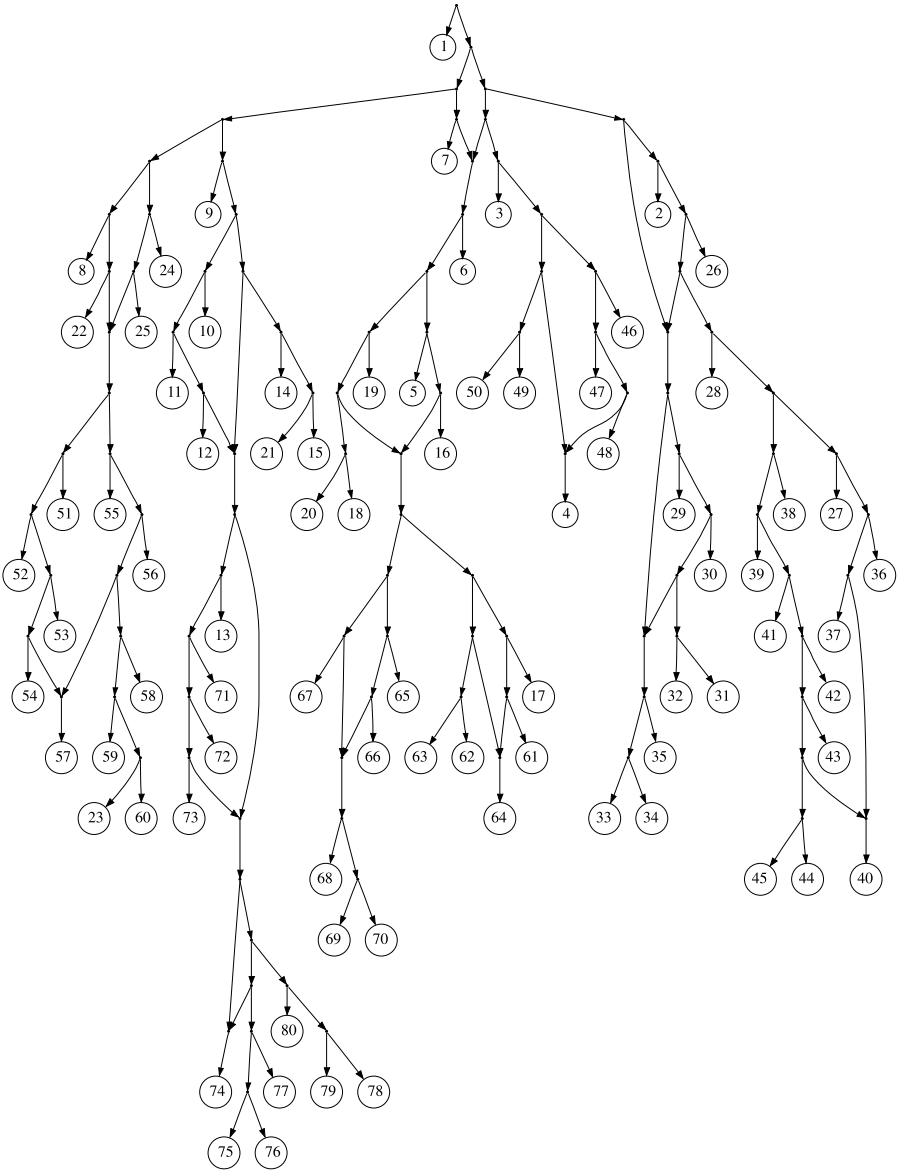

Fig. 4 Example of a network constructed by MARLON

In the construction of biconnected components with two reticulations, we use the notion of "non-cycle-reachable"-arc, or n.c.r.-arc for short, introduced in [16]. We call an arc $a = (u, v)$ an *n.c.r.-arc* if v is not reachable from any vertex in a cycle. These n.c.r.-arcs will be used to combine networks without increasing the network level. In addition, we use the notion *highest biconnected component* to denote the biconnected component containing the root.

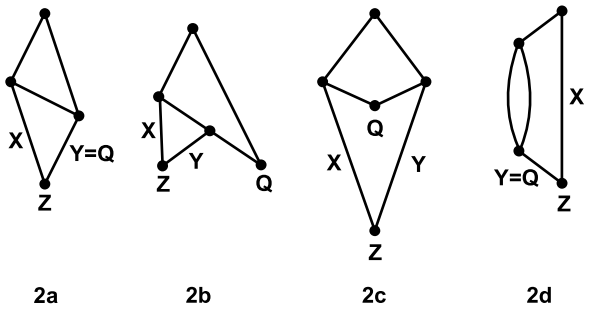

Fig. 5 The four possible structures of a biconnected component containing two reticulations

Our complete algorithm is described in detail in Algorithm 2. To get an intuition of why the algorithm works, consider the four possible structures of a biconnected component containing two reticulations displayed in Fig. 5. In particular, consider the graph that displays the form of the highest biconnected component of N_S. Let X (respectively Y, Z, Q) be the set of leaves below all cut-arcs leaving the side labelled X (respectively Y, Z, Q) in the figure. Observe that after removing Z in each case X, Y and Q become a set of leaves below a single cut-arc and hence an SN-set (w.r.t. $T|(S \setminus Z)$). In cases 2a, 2b and 2c the highest biconnected component becomes a cycle, Q the set of leaves below the highest reticulation and X and Y sets of leaves below highest cut-arcs. We will first describe the approach for these cases and show later how a similar technique is possible for case 2d.

Our algorithm loops through all SN-sets that are a subset of S and will hence at some iteration consider the SN-set Z. The algorithm removes the set Z and computes the SN-sets of $T|(S \setminus Z)$. The sets of leaves below highest cut-arcs (in some optimal solution, if one exists) are now equal to X, Y, Q and the SN-sets that are maximal under the restriction that they do not contain X, Y or Q (by the same arguments as in the proof of Lemma 1). Therefore, the algorithm tries each possible SN-set for X, Y and Q and in one of these iterations it will correctly determine the sets of leaves below highest cut-arcs. Then the algorithm computes the induced set of triplets, where each set of leaves below a highest cut-arc is replaced by a single meta-leaf. All simple level-1 networks consistent with this induced set of triplets are obtained by the algorithm in [17]. Our algorithm loops through all these networks and does the following for each simple level-1 network N_1. Each meta-leaf V, not equal to X or Y, is replaced by an optimal network N_V, which has been computed in a previous iteration. To include leaves in Z, X and Y, we compute an optimal network N_2 consistent with $T|(X \cup Z)$ and an optimal network N_3 consistent with $T|(Y \cup Z)$ where in both networks Z is the set of leaves below an n.c.r.-arc. Then we combine these three networks into a single network like in Fig. 6. A new reticulation is created and Z becomes the set of leaves below this reticulation. Finally, we check for each constructed network whether it is consistent with $T|S$. The network with the minimum number of reticulations over all such networks is the optimal solution N_S for this SN-set.

Now consider case 2d. Suppose we remove Z and replace X, Y $(= Q)$ and each SN-set of $T|(S \setminus Z)$ that is maximal under the restriction that it does not contain X or Y by a single leaf. Then the resulting network consists of a path ending in a simple

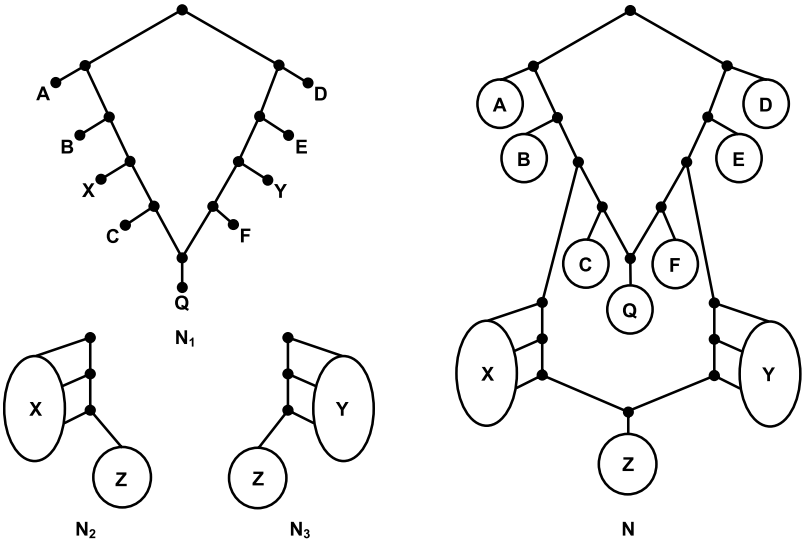

Fig. 6 Example of the construction of network N from N_1, N_2 and N_3

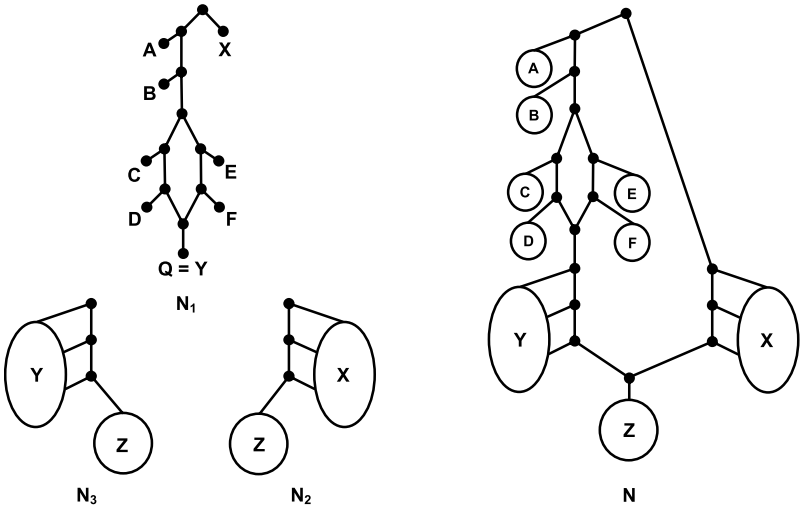

Fig. 7 Example of the construction of network N from N_1, N_2 and N_3 in case 2d

level-1 network, with X a child of the root and Q the child of the reticulation; and each vertex of the path has a leaf as child. Such a network can easily be constructed and subsequently one can use the same approach as in cases 2a, 2b and 2c. See Fig. 7 for an example of the construction in case 2d.

Theorem 2 *Given a dense triplet set T, Algorithm MARLTN constructs a level-2 network consistent with T (if such a network exists) that has a minimum number of reticulations in $O(n^9)$ time.*

Proof Consider some SN-set S and assume that there exists an optimal solution N_S consistent with $T|S$. The proof is by induction on the size of S. If the highest biconnected component of N_S contains one reticulation then the algorithm constructs an optimal solution by the proof of Theorem 1. Hence we assume from now on that the highest biconnected component of N_S contains two reticulations.

Consider the four graphs in Fig. 5. Any biconnected component containing two reticulations is a subdivision of one of these graphs [14, Lemma 13]. These graphs are called simple level-2 *generators* in [15] and X, Y, Z and Q each label, in each generator, a *side* of the generator, i.e. either an arc or a vertex with indegree 2 and outdegree 0. Suppose that the highest biconnected component of a network N is a subdivision of a generator G. We say that a leaf x *is on side* S of G if there exists a cut-arc (u, v) in N such that u is on the subdivision of S (if S is an arc) or u is a reticulation (if S is a reticulation), and there is a directed path from v to x (possibly $v = x$). Furthermore, we identify the side S with the set of leaves that are on side S.

In each generator in Fig. 5, X, Y, Z and Q are sides of the generator. Thus, if the highest biconnected component of N_S is a subdivision of a generator $G \in \{2a, 2b, 2c, 2d\}$ then we identify X (Y, Z, Q respectively) with the set of leaves in N_S on side X (Y, Z, Q respectively) of G.

To find an optimal network consistent with $T|(X \cup Z)$ (or $T|(Y \cup Z)$) such that Z is below an n.c.r.-arc we can use the following approach. If there are more than two maximal SN-sets then it is not possible. Otherwise, we create a root and connect it to two networks for the two maximal SN-sets. If one of these maximal SN-sets contains Z as a strict subset then we create a network for this set recursively. For other maximal SN-sets we use the optimal networks computed in earlier iterations.

Given a network N' and a set of leaves L' below a cut-arc (u, v) we denote by $N' \setminus L'$ the network obtained by removing v and all vertices reachable from v from N', deleting all vertices with outdegree zero and suppressing all vertices with indegree and outdegree both one.

Claim 2 There exists an optimal solution N' such that the sets of leaves below highest cut-arcs of $N' \setminus Z$ are X, Y, Q and the SN-sets of $T|(S \setminus Z)$ that are maximal under the restriction that they do not contain X, Y or Q.

Proof The highest biconnected component of $N' \setminus Z$ contains just one reticulation and the same arguments can be used as in the proof of Lemma 1. □

Let N' be a network with the property described in the claim above and $\mathcal{C}$ the collection of sets of leaves below highest cut-arcs of $N' \setminus Z$. At some iteration the algorithm will consider this set $\mathcal{C}$. Let T' equal $T|(S \setminus Z)$. If we replace in $N' \setminus Z$ each set of leaves below a highest cut-arc by a single leaf, then we obtain a network consistent with $T'\nabla\mathcal{C}$ which is either a simple level-1 network or a path ending in a simple level-1 network, with X a child of the root, Q the child of the reticulation; and

Algorithm 2 MARLTN (Minimum Amount of Reticulation Level Two Network)

– compute the set SN of SN-sets of T
for $i = 1 \ldots n$ **do**
  for each S in SN of cardinality i **do**
    \\ **try to construct a network for** S **with one reticulation in its highest biconnected component**
    for each $S' \in SN$ with $S' \subset S$ **do**
      – let $\mathcal{C}$ contain S' and all SN-sets that are maximal under the restriction that they are a subset of S and do not contain S'
      if $T \nabla \mathcal{C}$ is consistent with a simple level-1 network N_1 **then**
        – construct N^* from N_1 by replacing each leaf V by an optimal network N_V constructed in a previous iteration
        – $g(S, S')$ is the number of reticulations in N^*
    if there are exactly two SN-sets $S_1, S_2 \in SN$ that are maximal under the restriction that they are a strict subset of S **then**
      – N^* consists of a root connected to the roots of optimal networks N_{S_1} and N_{S_2} that have been constructed in previous iterations
      – $g(S, \emptyset)$ is the number of reticulations in N^*
    \\ **try to construct a network for** S **with two reticulations in its highest biconnected component**
    for each $Z \in SN$ with $Z \subset S$ **do**
      – $T' := T|(S \setminus Z)$
      – compute the set SN' of SN-sets of T'
      for each $X, Y, Q \in SN'$ **do**
        – $\mathcal{C}$ is the collection consisting of X, Y, Q and all SN-sets in SN' that are maximal under the restriction that they do not include X, Y or Q
        – construct an optimal network N_2 consistent with $T|(X \cup Z)$ such that Z is the set of leaves below an n.c.r.-arc (u, v)
        – construct an optimal network N_3 consistent with $T|(Y \cup Z)$ such that Z is the set of leaves below an n.c.r.-arc (u', v')
        – construct all simple level-1 networks consistent with $T' \nabla \mathcal{C}$
        – construct all networks consistent with $T' \nabla \mathcal{C}$ that consist of a path ending in a simple level-1 network, with X a child of the root, Q the child of the reticulation; and with a leaf below each internal vertex of the path
        for each network N_1 from the networks constructed in the above two lines **do**
          – construct N^* from N_1 by doing the following: replace X by N_2, Y by N_3 and each other leaf V by an optimal network N_V constructed in a previous iteration, then subdivide (u, v) into (u, w) and (w, v), delete everything below u' and add an arc (u', w)
          if N^* is consistent with $T|S$ **then**
            – $h(S, X, Y, Z, Q)$ is the number of reticulations in N^*
    – $f(S)$ is the minimum of all computed values of $g(S, S')$ and $h(S, X, Y, Z, Q)$
    – store network N_S, which is a network N^* attaining the minimum number $f(S)$ of reticulations

each vertex of the path has a leaf as child. The algorithm considers all networks of these types, so in some iteration it will consider the right one. Let N^* be the network constructed by the algorithm in this iteration. It remains to prove that N^* (i) is consistent with $T|S$, (ii) contains a minimum number of reticulations and (iii) is a level-2 network.

To prove that N^* is consistent with $T|S$, consider any triplet $xy|z \in T|S$. First suppose x, y and z are all in Z or all in the same set of $\mathcal{C} \setminus \{X, Y\}$. Then x, y and z are elements of some SN-set S' with $|S'| < |S|$. Triplet $xy|z$ is consistent with the subnetwork $N_{S'}$ by the induction hypothesis and hence with N^*.

Now suppose that x, y and z are all in $X \cup Z$ (or all in $Y \cup Z$). Consider the construction of the network consistent with $X \cup Z$ such that Z is below an n.c.r.-arc. First suppose that at some level of the recursion there are two maximal SN-sets, each containing leaves from $\{x, y, z\}$. Then it follows that x and y are in one maximal SN-set and z in the other one, by the definition of SN-set, and hence that $xy|z$ is consistent with the constructed network. Otherwise, x, y and z are all in some subnetwork N'_S with $|S'| < |S|$ and is $xy|z$ consistent with this subnetwork (by the induction hypothesis) and hence with N^*.

Now consider any other triplet $xy|z \in T|S$, which thus contains leaves that are below at least two different highest cut-arcs. Observe that the highest biconnected components of N^* and N' are identical; the only differences between these networks occur in the subnetworks below highest cut-arcs. Therefore $xy|z$ is consistent with N^* since it is consistent with N'.

To show that N^* contains a minimum number of reticulations consider any set S' of leaves below a highest cut-arc $a = (u, v)$ of N^*. The subnetwork $N_{S'}$ rooted at v contains a minimum number of reticulations by the induction hypothesis. Hence N^* contains at most as many reticulations as N', which is an optimal solution.

In the networks N_2 and N_3 is Z the set of leaves below an n.c.r.-arc. This implies that none of the potential reticulations in these networks end up in the highest biconnected component of N'. Therefore, this biconnected component contains exactly two reticulations. All other biconnected components of N' also contain at most two reticulations by the induction hypothesis. We thus conclude that N' is a level-2 network.

To conclude the proof we analyze the running time of the algorithm. The number of SN-sets is $O(n)$ and hence there are $O(n)$ choices for each of S, X, Y, Z and Q. For each combination of S, X, Y, Z and Q there will be $O(n)$ networks N^* constructed and for each of them it takes $O(n^3)$ time to check if it is consistent with $T|S$ (in line 23). Hence the overall time complexity is $O(n^9)$. □

5 Constructing Networks Consistent with Precisely the Input Triplet Set

In this section we consider the problem MIN-REFLECT-k. Given a triplet set T, this problem asks for a level-k network N that is consistent with precisely those triplets in T (if such a network exists) and amongst all such solutions minimizes both the level and number of reticulations used. We will show that this problem is polynomial-time solvable for each fixed k.

Recall that we use $T(N)$ to denote the set of all triplets consistent with a network N. Furthermore, we say that a network N *reflects* a triplet set T if $T(N) = T$. Note that, if N reflects T, that N is in general not uniquely defined by T. There are, for example, several distinct simple level-2 networks that reflect the triplet set $\{xy|z, xz|y, zy|x\}$.

Theorem 3 *Given a dense set of triplets T, it is possible to construct all simple level-k networks consistent with T in time $O(|T|^{k+1})$.*

This will be proven in Sect. 5.1. We note that it is already known how to generate all simple level-1 networks consistent with T in time $O(|T|)$ [17] and all simple level-2 networks consistent with T in time $O(|T|^{\frac{8}{3}})$ [14].

Lemma 3 *Let N be any simple network. Then all the nontrivial SN-sets of $T(N)$ are singletons.*

This lemma, as well as Theorem 4, will be proven in Sect. 5.2. As we will show, Lemma 3 allows us to solve the problem MIN-REFLECT-k by recursively constructing simple level-k networks, which we can do by Theorem 3. This leads to the algorithm MINPITS (MINimum network consistent with Precisely the Input Triplet Set).

Theorem 4 *Given a set of triplets T, Algorithm MINPITS solves MIN-REFLECT-k in time $O(|T|^{k+1})$, for any fixed k.*

For $k = 1, 2$ we can actually do slightly better: running time $O(|T|)$ and $O(|T|^{\frac{8}{3}})$ respectively.

5.1 Constructing All Simple Level-k Networks in Polynomial Time

We start by proving the following utility lemma.

Lemma 2 *For fixed k, any level-k network on n leaves contains $O(n)$ vertices and $O(n)$ arcs.*

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 $2 + 2s + k$. It is well known that in any directed graph 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:

$$|V'| = s + k + n + 1 = (n + k - 2) + k + n + 1 = 2n + 2k - 1. \tag{1}$$

Split vertices and reticulation vertices have total degree 3, leaves have total degree 1, and the root of N' has total degree 2. Thus the total number of arcs in N' is:

$$|A'| = \frac{3s + 3k + n + 2}{2} = \frac{3(n + k - 2) + 3k + n + 2}{2} = 2n + 3k - 2.$$

Now consider a general level-k network $N = (V, A)$ with q nontrivial biconnected components. Let $B(N)$ be the result of replacing each nontrivial biconnected component C by a single vertex (i.e. contracting all arcs in C). Thus $B(N)$ is a tree, q of the internal vertices of $B(N)$ represent biconnected components of N and the other internal vertices represent split-vertices of N. Let $B(N)$ contain b internal vertices with $n_1, \ldots, n_b$ outgoing arcs respectively. Denote by k_i the number of reticulations in the biconnected component of N represented by the i-th internal vertex of $B(N)$ and let $k_i = 0$ if this internal vertex represents a split-vertex of N. Then $B(N)$ contains $b+n$ vertices in total and $b+n-1$ arcs.

Assume the i-th internal vertex of $B(N)$ represents a biconnected component C_i of N. Consider C_i, the cut-arcs leaving C_i and the n_i vertices that have a parent on C_i. This forms a simple level-k_i network with n_i leaves. By (1) it has $2n_i + 2k_i - 1$ vertices. Replacing C_i by a single vertex thus reduces the number of vertices by $n_i + 2k_i - 2$.

The number of vertices of N is equal to the number of vertices in $B(N)$ plus the number of vertices that have been deleted while replacing each C_i by a single vertex:

$$\begin{aligned} |V| &= b + n + \sum_{i=1}^{b} (n_i + 2k_i - 2) \\ &\leq b + n + (b + n - 1) + 2q \cdot k - 2b \leq 2n - 1 + k(n-1). \end{aligned}$$

For the first inequality we use that $n_1 + \ldots + n_b$ is equal to the number of arcs of $B(N)$, which is $b+n-1$. For the second inequality we use that each nontrivial biconnected component of N has at least three outgoing arcs. This implies that $b \leq n - 1 - q$ and hence that $q \leq n - 1 - q$ and hence $q \leq (n-1)/2$. Similarly, the number of arcs of N is at most:

$$\begin{aligned} |A| &= b + n - 1 + \sum_{i=1}^{b} (n_i + 3k_i - 2) \\ &\leq b + n - 1 + (b + n - 1) + 3q \cdot k - 2b \leq 2n - 2 + \frac{3}{2} k(n-1). \end{aligned}$$ □

Let N be a network with at least one reticulation vertex, and let v be the child of a reticulation vertex in N. If v has no reticulation vertex as a descendant, then we call the subnetwork rooted at v a *Tree hanging Below a Reticulation vertex* (*TBR*). We additionally introduce the notion of the *empty* TBR, which corresponds to the situation when a reticulation vertex has no outgoing arcs. This cannot happen in a normal network but as explained shortly it will prove a useful abstraction.

Observation 1 *Every network N containing a reticulation vertex contains at least one TBR.*

Proof Suppose this is not true. Let v be the child of a reticulation vertex in N maximizing the longest path from the root to v. There must exist some vertex $v' \neq v$ which

is a child of a reticulation vertex and which is a descendent of v. But then the longest path from the root to v' is greater than to v, contradiction. □

Note that, because a TBR is (as a consequence of its definition) below a cut-arc, there exists an SN-set S of T such that $T|S$ is consistent with (only) the TBR. An SN-set S such that $T|S$ is consistent with a tree, we call a *CandidateTBR* SN-Set. Every TBR of N corresponds to some CandidateTBR SN-Set of T, but the opposite is not necessarily true. For example, a singleton SN-set is a CandidateTBR SN-Set, but it might not be the child of a reticulation vertex in N.

We abuse definitions slightly by defining the empty CandidateTBR SN-Set, which will correspond to the empty TBR. (This is abusive because the empty set is not an SN-set.) Furthermore we define that every triplet set T has an empty CandidateTBR SN-Set.

Observation 2 *Let T be a dense set of triplets on n leaves. There are at most $O(n)$ CandidateTBR SN-sets. All such sets, and the tree that each such set represents, can be found in total time $O(n^3)$.*

Proof First we construct the SN-tree for T, this takes time $O(n^3)$. There is a bijection between the SN-sets of T and the vertices of the SN-tree. (In the SN-tree, the children of an SN-set S are the maximal SN-sets of $T|S$.) Observe that a vertex of the SN-tree is a CandidateTBR SN-set if and only if it is a singleton SN-set *or* it has in total two children, and both are CandidateTBR SN-sets. We can thus use depth first search to construct all the CandidateTBR SN-sets; note that this is also sufficient to obtain the trees that the CandidateTBR SN-sets represent, because (for trees) the structure of the tree is identical to the nesting structure of its SN-sets. Given that there are only $O(n)$ SN-sets, the running time is dominated by construction of the SN-tree. □

We claim that algorithm SL-k constructs all simple level-k networks consistent with a dense input triplet set T. The high-level idea is as follows. Consider a simple level-k network N. From Observation 1, we know that N contains at least one TBR. (Given that N is simple we know that all TBRs are equal to single leaves. That is why the outermost loop of the algorithm can restrict itself to considering only single-leaf TBRs.) By looping through all CandidateTBR SN-sets we will eventually find one that corresponds to a real TBR. If we remove this TBR and the reticulation vertex from which it hangs, and then suppress any resulting vertices with both indegree and outdegree equal to 1, we obtain a new network (not necessarily simple) with one fewer reticulation vertex than N. Note that this new network might not be a "real" network in the sense that it might have reticulation vertices with no outgoing arcs. Repeating this k times in total we eventually reach a tree which we can construct using the algorithm of Aho et al. (and is unique, as shown in [18]). From this tree we can reconstruct the network N by reintroducing the TBRs back into the network (each TBR below a reticulation vertex) in the reverse order to which we found them. We don't, however, know exactly where the reticulation vertices were in N, so every time we reintroduce a TBR back into the network we exhaustively try every pair of arcs (as the arcs which will be subdivided to hang the reticulation vertex, and

thus the TBR, from.) Because we try every possible way of removing TBRs from the network N, and every possible way of adding them back, we will eventually reconstruct N.

The role of the dummy leaves in SL-k is linked to the empty TBRs (and their corresponding empty CandidateTBR SN-Sets). When a TBR is removed, it can happen (as mentioned above) that a network is created containing reticulation vertices with no outgoing arcs. (For example: when one of the parents of a reticulation vertex from which the TBR hangs, is also a reticulation vertex.) Conceptually we say that there *is* a TBR hanging below such a reticulation vertex, but that it is empty. Hence the need in the algorithm to also consider removing the empty TBR. If this happens, we will also encounter the phenomenon in the second phase of the algorithm, when we are re-introducing TBRs into the network. What do we insert into the network when we reintroduce an empty TBR? We use a dummy leaf as a place-holder, ensuring that every reticulation vertex always has an outgoing arc. The dummy leaves can be removed once that outgoing arc is subdivided later in the algorithm, or at the end of the algorithm, whichever happens sooner. The dummy root, finally, is needed for when there are no leaves on a side connected to the root.

Theorem 3 *Given a dense set of triplets T, it is possible to construct all simple level-k networks consistent with T in time $O(|T|^{k+1})$.*

Proof Correctness follows from the discussion above. We now analyze the running time. For $k \in \{1, 2\}$ we can actually generate all simple level-1 networks in time $O(n^3)$ using the algorithm in [17], and all simple level-2 networks in time $O(n^8)$ using the algorithm in [14]. For $k \geq 3$ we use SL-k. From Observation 2 we know that each execution of *FindCandidateTBRs* (which computes all TBRs in a dense triplet set plus the empty TBR) takes $O(n^3)$ time and returns at most $O(n)$ TBRs. Operations such as computing T'_i, and the construction of the tree N'_k, all require time bounded above by $O(n^3)$. The for loops when we "guess" the TBRs are nested to a depth of k. The for loops when we "guess" pairs of arcs from which to hang TBRs, are also nested to a depth of k. (There will only be $O(n)$ arcs to choose from by Lemma 2.) Checking whether N' is consistent with T, which we do inside the innermost loop of the entire algorithm, takes time $O(n^3)$ [6, Lemma 2]. So the running time is $O(n(n^3 + n(n^3 \ldots n(n^3 + n^{2k+3}))$ which is $O(n^{3k+3})$. □

Corollary 1 *For fixed k and a triplet set T it is possible to generate in time $O(n^{3k+3})$ all simple level-k networks N that reflect T.*

Proof The algorithm SL-k (or, for that matter, the algorithms from [15, 17]) can easily be adapted for this purpose: we change in line 43 "network consistent with T" to "network that reflects T". The running time is unchanged because, whether we are checking consistency or reflection, the implementation of [6, Lemma 2] implicitly generates $T(N')$. □

Algorithm 3 SL-k (Construct all Simple Level-k networks)

$Net := \emptyset$
$TBR_1 := L(T)$
for each leaf $b_1 \in TBR_1$ **do**
  $L'_1 := L(T) \setminus \{b_1\}$
  $T'_1 := T|L'_1$
  $TBR_2 := FindCandidateTBRs(T'_1)$
  for each $b_2 \in TBR_2$ **do**
    ...
    {Continue nesting **for** loops to a depth of k.}
    ...
    $TBR_k := FindCandidateTBRs(T'_{k-1})$
    for each $b_k \in TBR_k$ **do**
      $L'_k := L'_{k-1} \setminus b_k$
      $T'_k := T'_{k-1}|L'_k$
      {At this point we have finished "guessing" where the TBRs are,}
      {and $(\{b_1\}, b_2, \ldots, b_k)$ is a vector of (possibly empty) subsets of $L(T)$.}
      {We now "guess" all possible ways of hanging the TBRs back in.}
      if L'_k contains 2 or more leaves **then**
        build the unique tree $N'_k = (V, A)$ consistent with T'_k if it exists (see [18])
      else
        If L'_k contains 1 leaf $\{x\}$, let N'_k be the network comprising the single leaf $\{x\}$
        If L'_k contains 0 leaves, let N'_k be the network comprising a single, new dummy leaf
      $V := V \cup \{r'\}$; $A := A \cup \{(r', r)\}$ {with r the root of N'_k and r' a new dummy root}
      Let $H(b_k)$ be the unique tree consistent with b_k
      {Note: $H(b_k)$ is a single vertex if $|b_k| = 1$ and empty if $|b_k| = 0$.}
      for every two arcs a_k^1, a_k^2 in N'_k (not necessarily distinct) **do**
        Let p (respectively q) be a new vertex obtained by subdividing a_k^1 (respectively a_k^2)
        Connect p and q to a new reticulation vertex ret_k
        Hang $H(b_k)$ (or a new dummy leaf if $H(b_k)$ is empty) from ret_k
        if a_k^1 (or a_k^2) was the arc above a dummy leaf d **then**
          Remove d and if its former parent has indegree and outdegree 1, suppress that
          Let N'_{k-1} be the resulting network
          Let $H(b_{k-1})$ be the unique tree consistent with b_{k-1}
          for every two arcs a_{k-1}^1, a_{k-1}^2 in N'_{k-1} (not necessarily distinct) **do**
            ...
            {Continue nesting **for** loops to a depth of k.}

```
31:                 ...
32:                 Let N'_1 be the resulting network
33:                 Let H(b_1) be the tree consisting of only the single vertex b_1
                    for every two arcs a^1_1, a^2_1 in N'_1 (not necessarily distinct) do
34:                   Let p (respectively q) be a new vertex obtained by
                      subdividing a^1_1 (respectively a^2_1)
35:                   Connect p and q to a new reticulation vertex ret_1
36:                   Hang H(b_1) from ret_1
                      if a^1_1 (or a^2_1) was the arc above a dummy leaf d then
37:                     Remove d and if its former parent has indegree and
                        outdegree 1, suppress that
                    {This is the innermost loop of the algorithm.}
38:                 Let N' be the resulting network
39:                 Remove the dummy root r' from N'
40:                 Remove (and if needed suppress former parents of) any
                    remaining dummy leaves in N'
                    if N' is a simple level-k network consistent with T  then
41:                   Net := Net ∪ {N'}
42: return Net
```

5.2 From Simple Networks that Reflect, to General Networks that Reflect

For a triplet $xy|z$ and a network N, we define an *embedding* of $xy|z$ in N as any set of four paths $(q \to x, q \to y, p \to q, p \to z)$ which, except for their endpoints, are mutually vertex disjoint, and where $p \neq q$. We say that the vertex p is the *summit* of the embedding. Clearly, $xy|z$ is consistent with N if and only if there is at least one embedding of $xy|z$ in N.

Lemma 3 *Let N be any simple network. Then all the nontrivial SN-sets of $T(N)$ are singletons.*

Proof We prove the lemma by contradiction. Assume thus that there is some SN-set S of $T(N)$ such that $1 < |S| < |L(N)|$.

Let r be the root of N. An *in-out root embedding* is an embedding of any triplet $xz|y$ with $x, z \in S$ and $y \notin S$ that has r as its summit. We begin by proving that an in-out root embedding exists in N. Suppose by contradiction this is not true. For all $x, z \in S$ and $y \notin S$, the triplet $xz|y$ must be in $T(N)$ because $T(N)$ is dense. Consider a triplet embedding $(q \to x, q \to z, p \to q, p \to y)$ with $p \neq r$, $x, z \in S$, $y \notin S$ that minimizes (amongst all such embeddings) the length of the shortest directed path from r to p. Let P be any shortest directed path from r to p. Now, consider a path Q beginning somewhere on the path P. First observe that Q cannot intersect with the path $p \to y$ or $p \to q$, because this would contradict the minimality of the length of P. In addition, Q may not intersect with the path $q \to x$ $(q \to z)$ because this would mean $y \in S$. We conclude that such a path Q either terminates at a leaf l, or re-intersects with the path P. It cannot terminate at a leaf $l \notin S$ because then we obtain an

embedding of $xz|l$ that has a summit closer to the root than p, thus contradicting the minimality of P. It can also not end at a leaf $l \in S$, because then we have that $y \in S$. We conclude that all outgoing paths from P must re-intersect with P. However, given that P includes the root, and that in a network every vertex is reachable by a directed path from the root, it follows that the last arc on the path P must be a cut-arc. But this violates the biconnectivity of N, contradiction. We conclude that there exists at least one in-out root embedding in N.

Let $(q \to x, q \to z, r \to q, r \to y)$ be any in-out root embedding. We observe that the path $r \to y$ must contain at least one internal vertex, by biconnectivity. Also, at least one of $q \to x$ and $q \to y$ must contain an internal vertex, because it is not possible for a vertex in a simple network to have two leaf children.

We now argue that there must exist a *twist cover* of the path $r \to q$. This is defined as a non-empty set C of undirected paths (undirected in the sense that not all arcs need to have the same orientation) where (i) all paths in C are arc-disjoint from the in-out root embedding, (ii) exactly one path starts at an internal vertex s of (without loss of generality) $q \to z$, (iii) exactly one path ends at an internal vertex t of $r \to y$, (iv) all other start and endpoints of the paths in C lie on $r \to q$ and (v) for every vertex v of the path $r \to q$ (including r and q), there is at least one path in C that has its startpoint to the left of v, and its endpoint to the right. Property (v) is crucial because it says (informally) that every vertex on $r \to q$ is "covered" by some path that begins and ends on either side of it and is arc-disjoint from the embedding. The length of C is defined to be the sum of the number of arcs in each path in C.

Suppose for contradiction that a twist cover does not exist. We define a *partial* twist cover as one that satisfies all properties of a twist cover except property (v). Partial twist covers have thus at least one vertex on $r \to q$ that is not covered. (To see that there always exists at least one partial twist cover note that properties (ii) and (iii) in particular are satisfied by the fact that neither the removal of q nor r is allowed to disconnect N.) Let C be a partial twist cover with a minimum number of uncovered vertices. Let d be the uncovered vertex that is closest to q. If we removed d we would, by definition, disconnect the union of the paths in C with the in-out root embedding into a left part G and a right part H. The vertex d does not, however, disconnect N, so there must be some path P not in C that begins somewhere in G and ends somewhere in H. If P has its startpoint on a path $X \in C$ (where X will be in G) and/or an endpoint on a path $Y \in C$ (where Y will be in H) then these paths can be "merged" into a new path that strictly increases the number of vertices covered. The merging occurs as follows. We take the union of the arcs in P with those in X and/or Y and discard superfluous arcs until we obtain a path that covers a strict superset of the union of the vertices covered by X and/or Y. (In particular, the fact that P begins in G and ends in H means that the vertex d becomes covered.) In this way we obtain a new partial twist-cover with fewer uncovered vertices, contradiction. If P has both its startpoint and endpoint on vertices of $r \to q$ that are not on paths in C, then P can be added to the set C and this extends the number of covered vertices, contradiction. If P begins and/or ends elsewhere on the embedding then P can be added to C which again increases the number of vertices covered, contradiction. (If P begins on, without loss of generality, $q \to z$ then it becomes the new property-(ii) path and the old property-(ii) path should be discarded. Symmetrically, if P ends

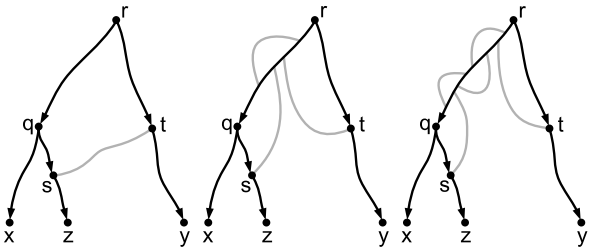

Fig. 8 Several examples of twist covers (the *grey*, undirected paths) from the proof of Lemma 3. Note that these exhibit the regular, interleaved structure associated with minimum-length twist covers

on $r \to y$ then it becomes the new property-(iii) path and the old property-(iii) path should be discarded.) We conclude that for every in-out root embedding there thus exists a twist cover, and in particular a minimum length twist cover.

We observe that a minimum-length twist cover C has a highly regular, interleaved structure. This regularity follows because it cannot contain paths that completely contain other paths (simply discard the inner path) and if two paths $X, Y \in C$ have start-points that are both covered by some other path $Z \in C$, and (without loss of generality) X reaches further right than Y, then we can simply discard Y. For similar reasons minimum-length twist covers are vertex- and arc-disjoint. In Fig. 8 we show several simple examples of twist covers exhibiting this regular structure (although it should be noted that minimum-length twist covers can contain arbitrarily many paths.)

Let C be a twist cover of minimum length ranging over all in-out root embeddings of triplets $xz|y$ with $x, z \in S$ and $y \notin S$. Note that if C contains exactly one path, which is a directed path, then (irrespective of the path orientation) $y \in S$, contradiction. The high-level idea is to show that we can always find, by "walking" along the paths in C, a new in-out root embedding and twist cover that is shorter than C, yielding a contradiction. Let X be the path in C that begins at s, and consider the arc on this path incident to s. The first subcase is that this arc is directed away from s. Note that in that case X cannot be a directed path since then there would be a directed cycle. Continuing along the path we will thus eventually encounter an arc with opposite orientation. Let v be the vertex between these two arcs. There must exist a directed path Q leaving v which eventually reaches a leaf m. If Q intersects with $r \to y$ then we have that $y \in S$, contradiction. If Q intersects with either $q \to x$ or $q \to z$ then we obtain a new in-out root embedding of $xz|y$ and a new twist cover for that embedding that is shorter than C, contradiction. If Q does not intersect with the embedding at all, then it must be true that $m \in S$ (because the triplet $mz|x$ is consistent with the network). But then we have an in-out root embedding of the triplet $zm|y$ with twist cover shorter than C, contradiction.

The second subcase (see Fig. 9) is when the first arc of X is entering s. There must exist some directed path R from r to s that uses this arc. The fact that r is the summit of the embedding means that at some point this directed path departs from the embedding. Let w be the vertex where R departs from the embedding for the last time. If w is on the path $r \to y$ then it follows that $y \in S$, a contradiction. If w is on one of the paths $q \to x$ or $q \to z$ then this leads to a new in-out root embedding with shorter twist cover, a contradiction. The last case is when w lies on the path $r \to q$. We create a new in-out root embedding by using the part of R reachable from w, as an alternative route to z. In this way w becomes the "q" vertex of the

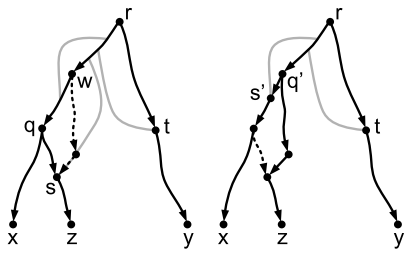

Fig. 9 The case in the proof of Lemma 3 where the arc incident to s is incoming

new embedding (denoted q' in the figure). To see that we also obtain a new twist cover, note principally that paths in C that covered w become legitimate candidates for property-(ii) paths in the new twist cover; in the figure s' denotes the beginning of the property-(ii) path in the new cover. (Such a path can however partly overlap with R. In this case it is necessary to first remove the part that overlaps with R.) We can furthermore discard all paths from C that covered w except for the one with endpoint furthest to the right. Even if this means that no paths from C are discarded (this happens when w is to the left of all the paths in C that have their beginning points on $r \rightarrow q$) we still get a twist cover at least one arc smaller than C, because (in particular) the first arc of X is no longer needed in C. In any case, contradiction. □

Corollary 2 *Let T be a set of triplets, and suppose there exists a simple network N that reflects T. Let N' be any network that reflects T. Then N' is also simple.*

Proof If N' is not simple then it contains a cut-arc below which at least two leaves can be found. Consider the set A of leaves below this cut-arc. This is an SN-set since triplets of the form $xy|z$ with $x, z \in A$ and $y \notin A$ are not consistent with such a network. This is a contradiction because all the SN-sets of T are singletons. □

Let T be a reflective set of triplets and N a network that reflects T. Observe that this implies that T is dense. Let $a_1, \ldots, a_q$ be the highest cut-arcs of N, let $S_1, \ldots, S_q$ be the sets of leaves below these highest cut-arcs and let $\mathcal{C}_N$ denote the collection $\{S_1, \ldots, S_q\}$. Denote by $Collapse(N)$ the result of replacing everything reachable from a_i by a single leaf S_i, for $i = 1, \ldots, q$. The following observation will be critical in the proof of Lemma 4, which shows correctness of our algorithm MINPITS, displayed in Algorithm 4.

Observation 3 (1) *$Collapse(N)$ is a simple network reflecting $T\nabla\mathcal{C}_N$ and* (2) *$S_1, \ldots, S_q$ are the maximal SN-sets of T.*

Proof To show that $Collapse(N)$ is consistent with $T\nabla\mathcal{C}_N$, consider triplets $XY|Z \in T\nabla\mathcal{C}_N$. For each such triplet there exists at least one triplet $xy|z \in T$ with $x \in X$, $y \in Y$ and $z \in Z$. Since N is consistent with $xy|z$ it follows that $Collapse(N)$ is consistent with $XY|Z$. To show that only triplets in $T\nabla\mathcal{C}_N$ are consistent wit $Collapse(N)$, consider a triplet $XY|Z$ consistent with $Collapse(N)$. It follows that all triplets $xy|z$ with $x \in X$, $y \in Y$ and $z \in Z$ are consistent with N. Since N reflects T this implies that all such triplets $xy|z$ are in T and hence that $XY|Z \in T\nabla\mathcal{C}_N$. By

the construction of $Collapse(N)$ it clearly contains no cut-arcs and is thus a simple network reflecting $T\nabla\mathcal{C}_N$. For (2) first observe that each set S_i is an SN-set of T since the set of leaves below any cut-arc is always an SN-set. Assume for contradiction that S_i is not maximal. Then there exists a maximal SN-set S of T that is a strict superset of S_i. Any maximal SN-set can be written as the union of sets of leaves below highest cut-arcs [14, Lemma 5]. Suppose (without loss of generality) that $S = S_1 \cup \ldots \cup S_r$ with $1 < r < q$ and $1 \leq i \leq r$. It follows that $\{S_1, \ldots, S_r\}$ is an SN-set of $T\nabla\mathcal{C}_N$, since the existence of a triplet $XY|Z \in T\nabla\mathcal{C}_N$ with $X, Z \subset S$, $Y \not\subset S$ would imply the existence of a triplet $xy|z \in T$ with $x, z \in S$ and $y \notin S$. However, this is a contradiction since by Lemma 3 all SN-sets of $T\nabla\mathcal{C}_N$ are singletons. □

Lemma 4 *Let T be a reflective set of triplets and let SN be the set of maximal SN-sets of T. Let N' be a simple network of minimum level that reflects $T\nabla SN$. Then replacing each leaf V of N' by a network that reflects $T|V$ and which* (*ranging over all networks that reflect $T|V$*) *simultaneously minimizes both level and number of reticulations*, *yields a network N that reflects T and which simultaneously minimizes both level and number of reticulations* (*ranging over all networks that reflect T*).

Proof Let N^0 be any network that reflects T. The sets of leaves below highest cut-arcs of N^0 are the maximal SN-sets of T by Observation 3. It follows that N and N^0 have the same sets of leaves below highest cut-arcs, i.e. $\mathcal{C}_N = \mathcal{C}_{N^0} = SN$. Thus $T\nabla SN = T\nabla\mathcal{C}_{N^0} = T\nabla\mathcal{C}_N$. Note also that for any maximal SN-set V the set of triplets $T|V$ is reflective: the subnetwork of N^0 below the highest cut-arc corresponding to V reflects $T|V$. This ensures that the recursive step does find some network.

To show that N reflects T, consider a triplet $xy|z$. First assume that x, y and z are all in the same maximal SN-set V, i.e. below the same highest cut-arc a of N. Then is $xy|z$ consistent with N if and only if $xy|z \in T$, since the subnetwork of N below the cut-arc a reflects $T|V$.

Now consider a triplet $xy|z$ with two leaves in the same maximal SN-set and the third leaf in a different maximal SN-set. Such a triplet is consistent with N if and only if x and y are below the same highest cut-arc and z below a different one. By the definition of SN-set (and using that T is dense), such a triplet is in T if and only if x and y are in the same maximal SN-set and z in a different one. Consequently, $xy|z$ is consistent with N if and only if $xy|z \in T$.

Finally, consider triplets $xy|z$ where x, y and z are all in different maximal SN-sets X, Y and Z respectively. First suppose $xy|z \in T$. Then it follows that $XY|Z \in T\nabla SN$ and hence that $XY|Z$ is consistent with N'. From this it follows that $xy|z$ is consistent with N, since an embedding of $XY|Z$ in N' can easily be extended to an embedding of $xy|z$ in N. To show the other direction, assume that $xy|z$ is consistent with N. Then is $XY|Z$ consistent with N' and hence $XY|Z \in T\nabla SN = T\nabla\mathcal{C}_{N^0}$. From the fact that N^0 reflects T it follows that $Collapse(N^0)$ reflects $T\nabla\mathcal{C}_{N^0}$. It follows that $XY|Z$ is consistent with $Collapse(N^0)$. It follows that for any $x' \in X$, $y' \in Y$ and $z' \in Z$ the triplet $x'y'|z'$ is consistent with N^0, implying that $x'y'|z' \in T$. This thus means that also $xy|z \in T$.

It is left to show that N is optimal, i.e. that it has a minimum number of reticulations and a minimum level over all networks that reflect T. Remember that any

Algorithm 4 MINPITS (MINimum network consistent with Precisely the Input Triplet Set)

```
1:  N := ∅
2:  compute the set SN of maximal SN-sets of T
3:  if |SN| = 2 then
4:      N consists of a root connected to two leaves: the elements of SN
5:  else
6:      if there exists a simple level-≤ k network that reflects T∇SN then
7:          let N be such a network of minimum level
8:      else
9:          N := ∅
10: for each leaf V of N do
11:     recursively create a level-k network N_V of minimal level (and which uses a
        minimum number of reticulations) that reflects T|V
12: if N ≠ ∅ and all N_V ≠ ∅ then
13:     replace each leaf V of N by the recursively created N_V.
14:     return N
15: else
16:     return ∅
```

network reflecting T has the maximal SN-sets of T as its sets of leaves below highest cut-arcs. Given that the subnetworks of N below its highest cut-arcs are optimal it follows that N is optimal if and only if N' is optimal. Finally, N' is optimal since it has minimum level and simple networks with minimum level also contain a minimum number of reticulations. □

Theorem 4 *Given a set of triplets T, Algorithm MINPITS solves MIN-REFLECT-k in time $O(|T|^{k+1})$, for any fixed k.*

Proof For $k = 0$ we can simply use the algorithm of Aho et al., which (with an advanced implementation [11]) can be implemented to run in time $O(n^3)$, which is $O(|T|)$. For $k \geq 1$ we use algorithm MINPITS. Correctness of the algorithm follows from Lemma 4. It remains to analyze the running time. A simple level-k network (that reflects the input) can be found (if it exists) in time $O(n^{3k+3})$ using algorithm SL-k. (To find the simple network of minimum level we execute in order SL-1, SL-2, ..., SL-k until we find such a network. This adds a multiplicative factor of k to the running time but this is absorbed by the $O(.)$ notation for fixed k.) Therefore, lines 6 and 7 of MINPITS take $O(|SN|^{3k+3})$ time. At every level of the recursion the computation of the maximal SN-sets of T can be done in time $O(n^3)$, and computation of $T \nabla SN$ also takes $O(n^3)$. The critical observation is that (by Observation 3) every SN-set in T appears exactly once as a leaf inside an execution of SL-k. The overall running time is thus of the form $O(\sum_i (n^3 + s_i^{3k+3}))$ where $\sum_i s_i$ is equal to the total number of SN-sets in T. Noting that $\sum_i s_i^{3k+3} \leq (\sum_i s_i)^{3k+3}$, and that there are at most $O(n)$ SN-sets in T, we obtain for $k \geq 1$ an overall running time of $O(n^{3k+3})$, which is $O(|T|^{k+1})$ because T is dense. Note that for $k \in \{1, 2\}$ we can actually do slightly

better by using the faster simple level-1 and simple level-2 algorithms from [14, 17]. This yields for $k = 1, 2$ overall running times of $O(|T|)$ and $O(|T|^{\frac{8}{3}})$ respectively. □

6 Conclusions and Open Questions

In this article we have shown that, for level 1 and 2, constructing a phylogenetic network consistent with a dense set of triplets that minimizes the number of reticulations, is polynomial-time solvable. We feel that, given the widespread use of the principle of parsimony within phylogenetics, this is an important development, and testing on simulated data has yielded promising results. However, the complexity of finding a *feasible* solution for level-3 and higher, let alone a minimum solution, remains unknown, and this obviously requires attention. Perhaps the feasibility and minimization variants diverge in complexity for higher k. It would be fascinating to explore this.

We have also shown, for *every* fixed k, how to generate in polynomial time all simple level-k networks consistent with a dense set of triplets. This could be an important step towards determining whether the aforementioned feasibility question is tractable for every fixed k. We have used this algorithm to show how MIN-REFLECT-k is polynomial-time solvable for fixed k. Clearly the demand that a set of triplets is exactly equal to the set of triplets in some network is an extremely strong restriction on the input. However, for small networks and high accuracy triplets such an assumption might indeed be valid, and thus of practical use. In any case, the concept of reflection is likely to have a role in future work on "support" for edges in phylogenetic networks generated via triplets. Also, the complexity of some fundamental questions like "does *any* network N reflect T?" remains unclear.

The complexity of constructing a minimum level network consistent with a dense triplet set is still an important open problem. The same holds for constructing a network with a minimum number of reticulations, without restrictions on the level. When restricting to simple networks, these two problems coincide. However, also in this restricted case, their complexity remains open.

Acknowledgements We thank Judith Keijsper, Matthias Mnich and Leen Stougie for many helpful discussions during the writing of the paper.

Open Access This article is distributed under the terms of the Creative Commons Attribution Noncommercial License which permits any noncommercial use, distribution, and reproduction in any medium, provided the original author(s) and source are credited.

References

1. Aho, A.V., Sagiv, Y., Szymanski, T.G., Ullman, J.D.: Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions. SIAM J. Comput. **10**(3), 405–421 (1981)
2. Baroni, M., Semple, C., Steel, M.: A framework for representing reticulate evolution. Ann. Comb. **8**, 391–408 (2004)
3. Baroni, M., Grünewald, S., Moulton, V., Semple, C.: Bounding the number of hybridisation events for a consistent evolutionary history. Math. Biol. **51**, 171–182 (2005)

4. Bordewich, M., Semple, C.: Computing the minimum number of hybridization events for a consistent evolutionary history. Discrete Appl. Math. **155**(8), 914–928 (2007)
5. Bordewich, M., Linz, S., John, K. St., Semple, C.: A reduction algorithm for computing the hybridization number of two trees. Evol. Bioinform. **3**, 86–98 (2007)
6. Byrka, J., Gawrychowski, P., Kelk, S.M., Huber, K.T.: Worst-case optimal approximation algorithms for maximizing triplet consistency within phylogenetic networks. arXiv:0710.3258v3 [q-bio.PE] (2008)
7. Choy, C., Jansson, J., Sadakane, K., Sung, W.-K.: Computing the maximum agreement of phylogenetic networks. Theor. Comput. Sci. **335**(1), 93–107 (2005)
8. Gusfield, D., Eddhu, S., Langley, C.: Optimal, efficient reconstructing of phylogenetic networks with constrained recombination. J. Bioinform. Comput. Biol. **2**(1), 173–213 (2004)
9. Gusfield, D., Hickerson, D., Eddhu, S.: An efficiently computed lower bound on the number of recombinations in phylogenetic networks: theory and empirical study. Discrete Appl. Math. **155**(6–7), 806–830 (2007)
10. He, Y.-J., Huynh, T.N.D., Jansson, J., Sung, W.-K.: Inferring phylogenetic relationships avoiding forbidden rooted triplets. J. Bioinform. Comput. Biol. **4**(1), 59–74 (2006)
11. Henzinger, M.R., King, V., Warnow, T.: Constructing a tree from homeomorphic subtrees, with applications to computational evolutionary biology. Algorithmica **24**(1), 113 (1999)
12. Huson, D.H., Bryant, D.: Application of phylogenetic networks in evolutionary studies. Mol. Biol. Evol. **23**(2), 254–267 (2006)
13. Huynh, T.N.D., Jansson, J., Nguyen, N.B., Sung, W.-K.: Constructing a smallest refining galled phylogenetic network. In: Proceedings of Research in Computational Molecular Biology (RECOMB 2005). LNCS, vol. 3500, pp. 265–280 (2005)
14. van Iersel, L.J.J., Keijsper, J.C.M., Kelk, S.M., Stougie, L.: Constructing level-2 phylogenetic networks from triplets. arXiv:0707.2890v1 [p-bio.PE] (2007)
15. van Iersel, L.J.J., Keijsper, J.C.M., Kelk, S.M., Stougie, L., Hagen, F., Boekhout, T.: Constructing level-2 phylogenetic networks from triplets. In: Proceedings of Research in Computational Molecular Biology (RECOMB 2008). LNBI, vol. 4955, pp. 450–462 (2008)
16. van Iersel, L.J.J., Kelk, S.M., Mnich, M.: Uniqueness, intractability and exact algorithms: reflections on level-k phylogenetic networks. arXiv:0712.2932v3 [q-bio.PE] (2008)
17. Jansson, J., Nguyen, N.B., Sung, W.-K.: Algorithms for combining rooted triplets into a galled phylogenetic network. SIAM J. Comput. **35**(5), 1098–1121 (2006)
18. Jansson, J., Sung, W.-K.: Inferring a level-1 phylogenetic network from a dense set of rooted triplets. Theor. Comput. Sci. **363**, 60–68 (2006)
19. Makarenkov, V., Kevorkov, D., Legendre, P.: Phylogenetic network reconstruction approaches. In: Applied Mycology and Biotechnology. Bioinformatics, vol. 6, pp. 61–97. Elsevier, Amsterdam (2006)
20. MARLON: constructing level one phylogenetic networks with a minimum amount of reticulation. http://homepages.cwi.nl/~kelk/marlon.html
21. Morrison, D.A.: Networks in phylogenetic analysis: new tools for population biology. Int. J. Parasitol. **35**(5), 567–582 (2005)
22. Moret, B.M.E., Nakhleh, L., Warnow, T., Linder, C.R., Tholse, A., Padolina, A., Sun, J., Timme, R.: Phylogenetic networks: modeling, reconstructibility, and accuracy. IEEE/ACM Trans. Comput. Biol. Bioinform. **1**(1), 13–23 (2004)
23. Song, Y.S., Hein, J.: On the minimum number of recombination events in the evolutionary history of DNA sequences. J. Math. Biol. **48**, 160–186 (2004)
24. Song, Y.S., Wu, Y., Gusfield, D.: Efficient computation of close lower and upper bounds on the minimum number of recombinations in biological sequence evolution. Bioinformatics **21**(1), i413–i422 (2005)
25. Wang, L., Zhang, K., Zhang, L.: Perfect phylogenetic networks with recombination. J. Comput. Biol. **8**(1), 69–78 (2001)