Constructing the Simplest Possible Phylogenetic Network from Triplets[*]

Leo van Iersel[1] and Steven Kelk[2]

[1] 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

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

Abstract. A phylogenetic network is a directed acyclic graph that visualises an evolutionary history containing so-called *reticulations* such as recombinations, hybridisations 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, which minimises both the level and the total number of reticulations, in time $O(|T|^{k+1})$, if k is a fixed upper bound on the level.

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 task almost a hopeless one [17]. This has motivated researchers to focus first on the simplest possible pattern of evolution. This least complicated shape of an evolutionary history is the tree-shape. Now that treelike evolution has been extremely well studied, a logical next step is to 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 (see e.g. [8]).

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

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

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, hybridisation 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 [3,9,15,17,18].

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 minimises the total number of reticulations for a fixed level (of at most two) and the second approach minimises 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. [5] and further studied by different authors [11,12,14]. Gusfield et al. gave a biological justification for level-1 networks (which they call "galled trees") [6]. Minimising reticulations has been very well studied in the framework where the input consists of (binary) sequences [6,8,19]. 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 [2] and Bordewich et al. showed that it is APX-hard to compute this minimum number exactly [4]. If restricted to level-1 networks the problem becomes polynomial-time solvable even if there are more than two input trees [10].

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. 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. [13] 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 [11]. From non-dense triplet sets it is NP-hard to construct level-k networks for any $k \geq 1$ [12,13]. From the proof of this result also follows directly that it is NP-hard to find a network consistent with a non-dense triplet

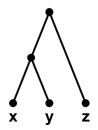

Fig. 1. One of the three possible triplets on the leaves x, y and z. Note that, as with all figures in this article, all arcs are directed downwards.

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 minimise 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 minimises the total number of reticulations over all such networks. We have implemented MARLON, tested it and made it publicly available [16]. 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 minimises the total number of reticulations over all such networks. This means that if the level is at most two, we can minimise 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 $O(|T|^3)$).

Constructing level-k phylogenetic networks becomes even more challenging when the level can be larger than two, even without minimising 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 minimises 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, optimised simple level-1

[1] This follows from the proof of Theorem 7 in [13], since only one reticulation is used in their reduction.

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. 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 $L' \subseteq L(T)$, 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 $S_i S_j | S_k$ 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 $u \rightarrow x$, $u \rightarrow y$, $v \rightarrow u$ and $v \rightarrow z$.*

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

[2] This definition is equivalent to Definition 4 in [11] by Lemma 2 in [11].

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. Any two SN-sets of a dense triplet set T are either disjoint or one is included in the other [14, Lemma 8], which implies that there are at most $2(n-1)$ nontrivial SN-sets. All SN-sets can be found in $O(n^3)$ time [13]. If a network is consistent with 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 [11].

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

Given a dense set of triplets T, the problem DMRL-k asks for a level-k network consistent with T 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 (in $O(n^3)$ time [13]) 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.

Algorithm 1. Minimum Amount of Reticulation Level One Network

```
1: compute the set $SN$ of SN-sets of $T$
2: for $i = 1 \dots n$ do
3:     for each $S$ in $SN$ of cardinality $i$ do
4:         for each $S' \in SN$ with $S' \subset S$ do
5:             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 $S'$
6:             if $T \nabla \mathcal{C}$ is consistent with a simple level-1 network then
7:                 $g(S, S') := 1 + \sum_{X \in \mathcal{C}} f(X)$
8:         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
9:             $g(S, \emptyset) := f(S_1) + f(S_2)$ ($\mathcal{C} := \{S_1, S_2\}$)
10:        $f(S) := \min g(S, S')$ over all computed values of $g(S, \cdot)$
11:        store the optimal $\mathcal{C}$ and the corresponding simple level-1 network
12: construct an optimal network by backtracking.
```

The following property of optimal level-1 networks shows that the algorithm computes an optimal solution.

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.

Claim (1). Each maximal SN-set S equals either the set of leaves below a highest cut-arc or the set of leaves below a directed path P ending in the highest reticulation or in one of its parents.

Proof. If S equals the set of leaves below a single highest cut-arc then we are done. From now on assume that S equals the set of leaves below different highest cut-arcs. First 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, because S is an SN-set. From this follows that all leaves in S are below some directed path P on the highest cycle. First assume that not all leaves reachable from vertices in P are in S. Then there are leaves x, z, y reachable respectively from vertices p_1, p_2, p_3 that are on P (in this order) with $x, y \in S$ and $z \notin S$. But this leads to a contradiction because then the triplet $xy|z$ is not consistent with N, whilst $yz|x$ and $xz|y$ cannot be 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 $z \notin S$ reachable from v and some leaves $x, y \in S$ below different highest cut-arcs. Then this again leads to a contradiction because $xy|z$ is not consistent with N. □

First suppose that some maximal SN-set S equals the set of leaves below a directed path P ending 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 the resulting vertex with indegree and outdegree both equal to one. It is not too difficult to see that the resulting network is still consistent with T.

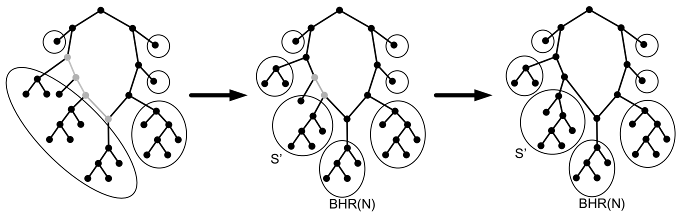

Fig. 2. Visualisation of the proof of Lemma 1. From the maximal SN-sets (encircled in the network on the left) to the sets of leaves below highest cut-arcs (encircled in the network on the right). Remember that all arcs are directed downwards.

Now suppose that some maximal SN-set S equals the set of leaves below a directed path P ending in the highest reticulation. The sets of leaves below highest cut-arcs are all SN-sets (as is always the case). One of them is equal to $BHR(N)$. If any of the others is contained in a nontrivial SN-set S' that does not contain $BHR(N)$, then the procedure from the previous paragraph can again be used to put S' below a highest cut-arc. In the resulting network 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 $BHR(N)$.

An example is given in Fig. 2. 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 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 $BHR(N)$. $\square$

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.*

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. Details have been omitted due to space constraints.

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.

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 [12]. 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.

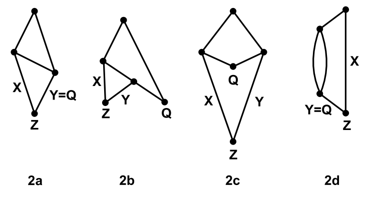

Fig. 3. The four possible structures of a biconnected component containing two reticulations

To get an intuition of the approach, consider the four possible structures of a biconnected component containing two reticulations displayed in Fig. 3. Let X, Y, Z and Q be the sets of leaves indicated in the graph that displays the form of the highest biconnected component of N_S. Observe that after removing Z in each case X, Y and Q become a set of leaves below a 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

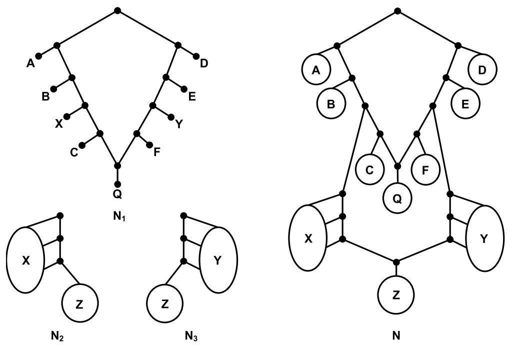

Fig. 4. Example of the construction of network N from N_1, N_2 and N_3

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 [13]. 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. 4. 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 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.

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.*

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 minimises both the level and number of reticulations used. We will show that this problem is polynomial-time solvable for each fixed k.

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.* 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 $\{xy|z, xz|y, zy|x\}$.

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, minimises both the level and the number of reticulations.

This problem might at first glance seem strangely formulated because, in general, minimising level and minimising number of reticulations are two distinct optimisation criteria. However, in the case of reflectivity it turns out that any solution that minimises the number of reticulations also minimises level.

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

We note that it is already known how to generate all simple level-1 networks consistent with T in time $O(|T|)$ [13] and all simple level-2 networks consistent with T in time $O(|T|^{\frac{8}{3}})$ [11].

Lemma 2. *Let N be any simple network. Then all the nontrivial SN-sets of $T(N)$ are singletons.*

The high-level idea behind solving MIN-REFLECT-k is that, if a network N reflects a triplet set T, the sets of leaves below highest cut-arcs are in 1:1 correlation with the maximal SN-sets of T. This is a consequence of Lemma 2. We thus use the maximal SN-sets to induce a new triplet set T', which must be reflected by some simple level-ℓ network, with $\ell \leq k$. Using Theorem 3 we can easily find such a (minimum level) simple network: if we can generate all networks *consistent* with a triplet set T, it is easy to identify which of those *reflect* T. This leads ultimately to the algorithm MINPITS (MINimum network consistent with Precisely the Input Triplet Set). The key to understanding why MINPITS produces solutions that simultaneously minimise level and the number of reticulations, lies in the fact that in any two networks N and N' that reflect T, the leaf partition induced by the highest cut-arcs of N, and the leaf partition induced by the highest cut-arcs of N', are identical. It follows that the only way in which N and N' can differ from each other, lies in the choice of simple network at each recursive step of the algorithm. But we always choose the simple network of minimum level, and it is not too difficult to see that this leads to the global minimisation of both level and the number of reticulations. Proofs and algorithms have been omitted due to space restraints.

Theorem 4. *Problem MIN-REFLECT-k can be solved 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.

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 minimises 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 minimisation 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.

Acknowledgements

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

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 Journal on Computing 10(3), 405–421 (1981)

2. Baroni, M., Grünewald, S., Moulton, V., Semple, C.: Bounding the Number of Hybridisation Events for a Consistent Evolutionary History. Mathematical Biology 51, 171–182 (2005)
3. Baroni, M., Semple, C., Steel, M.: A Framework for Representing Reticulate Evolution. Annals of Combinatorics 8, 391–408 (2004)
4. Bordewich, M., Semple, C.: Computing the Minimum Number of Hybridization Events for a Consistent Evolutionary History. Discrete Applied Mathematics 155(8), 914–928 (2007)
5. Choy, C., Jansson, J., Sadakane, K., Sung, W.-K.: Computing the Maximum Agreement of Phylogenetic Networks. Theoretical Computer Science 335(1), 93–107 (2005)
6. Gusfield, D., Eddhu, S., Langley, C.: Optimal, Efficient Reconstructing of Phylogenetic Networks with Constrained Recombination. Journal of Bioinformatics and Computational Biology 2(1), 173–213 (2004)
7. He, Y.-J., Huynh, T.N.D., Jansson, J., Sung, W.-K.: Inferring Phylogenetic Relationships Avoiding Forbidden Rooted Triplets. Journal of Bioinformatics and Computational Biology 4(1), 59–74 (2006)
8. Hein, J.: Reconstructing Evolution of Sequences Subject to Recombination Using Parsimony. Mathematical Biosciences 98, 185–200 (1990)
9. Huson, D.H., Bryant, D.: Application of Phylogenetic Networks in Evolutionary Studies. Molecular Biology and Evolution 23(2), 254–267 (2006)
10. Huynh, T.N.D., Jansson, J., Nguyen, N.B., Sung, W.-K.: Constructing a Smallest Refining Galled Phylogenetic Network. In: Miyano, S., Mesirov, J., Kasif, S., Istrail, S., Pevzner, P.A., Waterman, M. (eds.) RECOMB 2005. LNCS (LNBI), vol. 3500, pp. 265–280. Springer, Heidelberg (2005)
11. 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: Vingron, M., Wong, L. (eds.) RECOMB 2008. LNCS (LNBI), vol. 4955, pp. 450–462. Springer, Heidelberg (2008)
12. 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)
13. Jansson, J., Nguyen, N.B., Sung, W.-K.: Algorithms for Combining Rooted Triplets into a Galled Phylogenetic Network. SIAM Journal on Computing 35(5), 1098–1121 (2006)
14. Jansson, J., Sung, W.-K.: Inferring a Level-1 Phylogenetic Network from a Dense Set of Rooted Triplets. Theoretical Computer Science 363, 60–68 (2006)
15. Makarenkov, V., Kevorkov, D., Legendre, P.: Phylogenetic Network Reconstruction Approaches. In: Applied Mycology and Biotechnology. International Elsevier Series 6, Bioinformatics, pp. 61–97 (2006)
16. MARLON: Constructing Level One Phylogenetic Networks with a Minimum Amount of Reticulation, `http://homepages.cwi.nl/~kelk/marlon.html`
17. Morrison, D.A.: Networks in Phylogenetic Analysis: New Tools for Population Biology. International Journal for Parasitology 35(5), 567–582 (2005)
18. 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 Transactions on Computational Biology and Bioinformatics 1(1), 13–23 (2004)
19. Song, Y.S., Hein, J.: On the Minimum Number of Recombination Events in the Evolutionary History of DNA Sequences. Journal of Mathematical Biology 48, 160–186 (2004)