Constructing Level-2 Phylogenetic Networks from Triplets*

Leo van Iersel[1], Judith Keijsper[1], Steven Kelk[2], Leen Stougie[1,2], Ferry Hagen[3], and Teun Boekhout[3]

[1] Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, Den Dolech 2, 5612 AX Eindhoven, The Netherlands
{l.j.j.v.iersel,j.c.m.keijsper,l.stougie}@tue.nl
[2] Centrum voor Wiskunde en Informatica (CWI), Kruislaan 413, 1098 SJ Amsterdam, The Netherlands
{s.m.kelk,leen.stougie}@cwi.nl
[3] Centraalbureau voor Schimmelcultures (CBS), Fungal Biodiversity Center, Uppsalalaan 8, 3584 CT Utrecht, The Netherlands
{f.hagen,t.boekhout}@cbs.knaw.nl

Abstract. Jansson and Sung showed that, given a dense set of input triplets T (representing hypotheses about the local evolutionary relationships of triplets of taxa), it is possible to determine in polynomial time whether there exists a *level-1 network* consistent with T, and if so to construct such a network [18]. Here we extend this work by showing that this problem is even polynomial-time solvable for the construction of *level-2* networks. This shows that, assuming density, it is tractable to construct plausible evolutionary histories from input triplets even when such histories are heavily non-tree like. This further strengthens the case for the use of triplet-based methods in the construction of phylogenetic networks. We also implemented the algorithm and applied it to yeast data.

1 Introduction

Phylogenetics is the field at the interface of biology, mathematics and computer-science which studies the (re-)construction of plausible evolutionary scenarios when confronted with incomplete and/or error-prone biological data. Until recently almost all research effort was directed at finding evolutionary trees. However, biologically, especially for lower order species, evolution does not necessarily exhibit a tree structure. Thus, the quest for models and methods for more general evolutionary structures than trees has emerged naturally. This forms the subject of this paper.

Many algorithmic strategies for constructing evolutionary trees have been proposed in the literature. The most well-known techniques are Maximum Parsimony (MP), Maximum Likelihood (ML), Bayesian methods, Distance-based

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

M. Vingron and L. Wong (Eds.): RECOMB 2008, LNBI 4955, pp. 450–462, 2008.

methods (such as Neighbour Joining and UPMGA) and quartet-based methods, as well as various (meta-)combinations of these [3][10][19][24].

Quartet methods apply to the construction of unrooted evolutionary trees; less well studied is the problem of constructing *rooted* evolutionary trees, where the edges of the tree are directed to reflect the direction of evolution. The analogue of quartet methods in the case of rooted evolutionary trees are *triplet* methods: here we are given not unrooted trees on four leaves, but rooted binary trees on three leaves, as in Figure 1. The unique *rooted triplet* (*triplet* for short) on a leaf set $\{x, y, z\}$ in which the lowest common ancestor of x and y is a proper descendant of the lowest common ancestor of x and z is denoted by $xy|z$ (which is identical to $yx|z$). The triplet in the figure is $xy|z$.

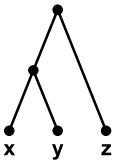

Fig. 1. One of the three possible triplets on the leaves x, y and z. Note that, as with all figures in this article, all arcs are directed downwards, away from the root.

Aho et al. studied the problem of constructing a tree from a set of triplets. They showed that, given a set of triplets, it is possible to construct in polynomial time a rooted tree consistent with all the input triplets, or decide that no such tree exists [1]. This contrasts favourably with the corresponding quartet problem, which is NP-hard [25]. Various authors [2][6][14][15][26] have studied variations of the problem in cases where the algorithm of Aho et al. fails to return a tree. A well-studied one, albeit NP-hard [14], is to find a tree that maximises the number of input triplets it is consistent with.

In recent years attention has turned towards the construction of evolutionary scenarios that are not tree-like. This has been motivated by the fact that biological phenomena such as hybridisation, horizontal gene transfer, recombination, and gene duplication can cause lineages, which earlier in time diversified from a common ancestor, to once again intersect with each other later in time. These kind of evolutionary events are called *reticulation events* and lead to evolutionary scenarios where the underlying undirected graph potentially contains cycles. Rather than attempt to summarise this extremely varied area we refer the reader to [11], [22] and [23], all outstanding survey articles.

Informally, a level-k network is an evolutionary network where each biconnected component of the network contains at most k reticulation events. Jansson, Sung and Nguyen considered the problem of deciding whether, given a set of input triplets, it is possible to construct a *level-1* network (otherwise known as a *galled tree* [8] or a *galled network* [4][9][16]) consistent with all those triplets [16][18]. They showed that, in general, the level-1 problem is NP-hard. (In contrast, the algorithm of Aho et al. always runs in polynomial time.) However,

they show that the problem can be solved in polynomial time when the input is *dense*, meaning that for each set of three taxa there is at least one triplet in the input. Density is a reasonable assumption if high-quality triplets can be constructed for all subsets of three taxa. The authors also give various upper bounds, lower bounds and approximation algorithms for the general case [16]. Related problems studied comprise the construction of level-1 networks from ultrametric distance matrices [4] and building level-1 networks where certain input triplets are *forbidden* [9]. Huson and Klöpper [12] consider a generalisation of level-1 networks that they call *galled networks* (defined differently than in [4][9][16]).

In this paper we extend considerably the work of Jansson and Sung by showing that, when the input set is dense, we can construct in polynomial time a *level-2* network consistent with all input triplets or decide that no such network exists. In case of a general (possibly non-dense) input triplet set, we claim that it is NP-complete to decide whether a level-2 network consistent with the input exists [13]. The proof of this claim, omitted here, is a nontrivial extension of the proof that this problem is NP-hard for level-1 networks [16]. In Section 3 we present our algorithm for constructing level-2 networks from dense triplet sets that runs in (slightly better than) cubic time in the number of input triplets. This significantly extends the power of triplet methods because it further extends the complexity of the evolutionary scenarios that can be constructed. For example, networks of the complexity shown in Figure 2 can be constructed by our algorithm. Our result and the way it is proved make it tempting to conjecture that, for fixed k, the level-k problem with dense triplet sets is polynomial-time solvable. However, it is not yet clear that the pivotal theorem in Section 3, Theorem 2, generalises easily to level-3 networks and higher.

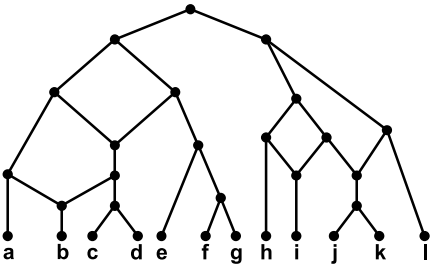

Fig. 2. An example of a level-2 network

An implementation of our algorithm in Java has been made publicly available [21]. We applied it to sequences from the yeast *Cryptococcus gattii*. We constructed a level-2 network for different isolates of this, potentially dangerous, yeast and are planning to use this network to find the origin of a *C. gattii* outbreak on the Westcoast of Canada. The results are reported in Section 4. In

Section 5 we discuss our conclusions and the many fascinating open problems that still remain in this area.

2 Preliminaries

A *phylogenetic tree* is a rooted binary tree with directed edges (arcs) and distinctly labelled leaves. A triplet $xy|z$ is thus a phylogenetic tree on three leaves. 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 indegree 1 and outdegree 0 (*leaves*), where the leaves are distinctly labelled. In general directed acyclic graphs a reticulation vertex is a vertex with indegree 2. 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.

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

A tree can thus be considered a level-0 network. A network that is level-k but not level-$(k-1)$ is called a *strict level-k network*.

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. Furthermore, 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 $L_t \subseteq L'$. We use L as shorthand for $L(T)$.

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

By extension, a set of triplets T is said to be *consistent* with N (interchangeably: N is consistent with T) if every triplet in T is consistent with N. To clarify triplet consistency we observe that the network in Figure 2 is consistent with (amongst others) $ab|c$, $bc|a$ and $dg|k$ but not consistent with (for example) $ah|f$ or $hk|i$.

We will now define SN-sets, introduced in [18], which will play a crucial role in the rest of the paper. For a triplet set T, let σ_T be the operation on subsets X of $L(T)$ defined by $\sigma_T(X) = X \cup \{c \in L(T) | \exists x, y \in X : xc|y \in T\}$. The set $SN_T(X)$ is defined as the closure of X w.r.t. the operation σ_T. Define an *SN-set* of T as a set of the form $SN_T(X)$ for some $X \subseteq L(T)$, i.e. SN-sets are the subsets of $L(T)$ that are closed under the operation σ_T. An SN-set X is *maximal* with respect to a triplet set T if $X \neq L(T)$ and $L(T)$ is the only SN-set that is a strict superset of X. In [18] it is shown that the maximal SN-sets of T partition $L(T)$ if T is dense.

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. A cut-arc is *highest* if there does not exist a cut-arc $a' = (u', v')$ such that u is reachable from v'. We say that a vertex is *below* (u, v) if it is reachable from v. A little thought should make it clear that for each cut-arc a in a network N consistent with a dense triplet set T, the set S of leaves below a is an SN-set of T and that the sets of leaves below highest cut-arcs partition L_N. The following lemma reveals a crucial characteristic that we exploit in our algorithm.

Lemma 1. *Let N be a network consistent with dense triplet set T. Each maximal SN-set S in T can be expressed as the union of the leaves below one or more highest cut-arcs in N.*

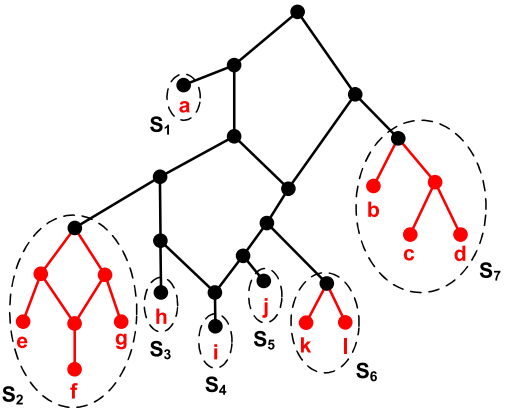

Fig. 3. Constructing a level-2 network by recursively constructing "simple" networks, given the partition $L = S_1 \cup \ldots \cup S_7$

3 Constructing Level-2 Networks from Dense Triplet Sets

This section describes our main result, a polynomial time algorithm that constructs a level-2 network from a dense triplet set T if such a network exists. The algorithm is recursive. The main idea is visualised in Figure 3. Suppose that we know the correct partition $L = S_1 \cup \ldots \cup S_7$ of the leaves. Then an algorithm can replace each set S_i by a single (meta-)leaf and start with constructing this "simple" level-2 network (in black). In Subsection 3.1 we formally introduce simple level-2 networks and show how they can be constructed. The complete level-2 network can then be obtained by replacing each meta-leaf S_i by a recursively created level-2 network. In spirit, this procedure resembles that for level-1 networks [18]. However, besides the fact that the simple level-2 networks are more complex, it also turns out that finding the right partition (and especially the proof of correctness) is far more involved than in the level-1 version of the problem. There does for example not always exist a level-2 network where the sets

of leaves below highest cut-arcs correspond to the maximal SN-sets, as is the case for level-1 networks. How the recursion works in detail and how the correct partition can be found is explained in Subsection 3.2. Due to space constraints we have to defer all proofs to a full version of the paper.

3.1 Simple Level-2 Networks

We now introduce the class of level-2 networks that we name *simple* level-2 networks. Informally these are the basic building blocks of level-2 networks in the sense that each biconnected component of a level-2 network is in essence a simple level-2 network. We first introduce a *simple level-k generator*:

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

In simple level-k generators, vertices with indegree 2 and outdegree 0 as well as all arcs are labelled and called *sides*. A simple case-analysis shows that there is only one simple level-1 generator and that there are four simple level-2 generators, depicted in Figure 4.

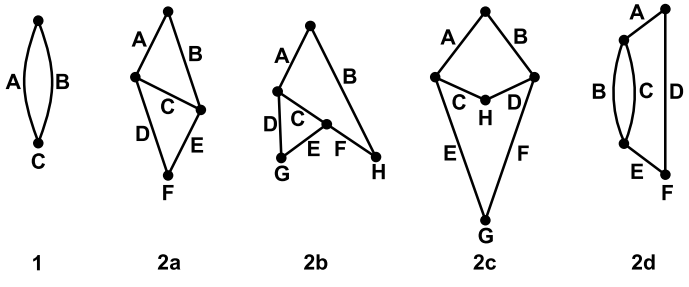

Fig. 4. The only simple level-1 generator and all four simple level-2 generators

Definition 4. *A simple level-k network N is a network obtained by applying the following transformation to some simple level-k generator such that the resulting graph is a valid network:*

1. *replace each arc X by a path and for each internal vertex v of the path add a new leaf x and an arc (v, x); and*
2. *for each vertex Y of indegree 2 and outdegree 0 add a new leaf y and an arc (Y, y).*

Note that in the above definition a path used to replace an edge possibly contains no internal vertices. An exception to this, however, is that whenever there are multiple arcs, we replace at least one of them by a path of at least three vertices. There is a nice and simple characterisation of simple level-k networks.

Lemma 2. *A strict level-k network is a simple level-k network if and only if it contains no nontrivial cut-arcs.*

All simple level-1 networks on dense triplet sets can be found by an algorithm by Jansson, Nguyen and Sung [16]. We designed Algorithm SL2 to find all simple level-2 networks consistent with a dense set of triplets T. Before describing the algorithm, we will first give an analysis of the structure of simple level-2 networks, whereon the algorithm is based. We use the notion *reticulation leaf* for a leaf whose parent is a reticulation vertex and say that a network is a *caterpillar* if the deletion of all leaves gives a directed path. We call a subset of the leaves L' a *caterpillarset* if there exists a network consistent with the input triplets that contains a caterpillar with leaves L' as subgraph. We prove that there are only $O(n)$ caterpillarsets and designed a routine to construct all of them from the triplets in $O(n^5)$ time. Now consider any simple level-2 network. If we remove a reticulation leaf and its parent, there is one reticulation vertex y left and below it is a caterpillar. If we now remove this last reticulation vertex and the caterpillar below it, we obtain a tree, which is unique [17] and can be constructed using the algorithm of Aho et al. [1]. The algorithm SL2 will first identify this tree and then reconstruct the simple level-2 network from it as follows.

Algorithm 1. SL2 (sketch)

```
 1: for each leaf x ∈ L do
 2:    delete all triplets containing x
 3:    for each caterpillarset Q do
 4:       delete all triplets containing a leaf in Q
 5:       build the unique tree consistent with the remaining triplets
 6:       for every two arcs a1 and a2 in this tree do
 7:          subdivide a1 and a2 by new vertices and put the caterpillar consistent with
             T|Q below below both new vertices by introducing a reticulation vertex y
 8:          for every two arcs a3 and a4 in the resulting level-1 network do
 9:             subdivide a3 and a4 by new vertices and make x a new reticulation leaf
                below both new vertices
10:             if the obtained network N is a simple level-2 network consistent with T
                then
11:                output N.
```

Now suppose that triplet set T is consistent with some simple level-2 network N. At some iteration the algorithm will choose the right leaf and caterpillarset to remove and the right arcs to subdivide and the algorithm will construct the network N. Furthermore, for each constructed network the algorithm checks whether it is a simple level-2 network consistent with T. We conclude that the algorithm finds exactly all simple level-2 networks consistent with T.

For several reasons the actual algorithm is more complicated than the sketch above. Firstly, some special cases occur when there are no leaves on certain sides of the simple level-2 network. These cases can be dealt with by introducing dummy vertices. Secondly, we can improve the running time of the algorithm

slightly if we loop through only a part of all combinations of arcs. Finally, we present an $O(n^3)$ routine that checks whether a given network N is a simple level-2 network consistent with a given triplet set T. This leads to an overall running time of $O(n^8)$.

Theorem 1. *Algorithm SL2 finds all simple level-2 networks consistent with a dense triplet set and can be implemented to run in time* $O(n^8)$. □

3.2 From Simple to General Level-2 Networks

This section explains how to build general level-2 networks by recursively building simple level-1 and -2 networks. The following theorem will be crucial.

Theorem 2. *Let T be a dense triplet set consistent with some level-2 network N. Then there exists a level-2 network N' consistent with T such that at most one maximal SN-set of T equals the union of the sets of leaves below two highest cut-arcs and each other maximal SN-set is equal to the set of leaves below just one highest cut-arc.*

For a collection $\mathcal{S} = \{S_1, \ldots, S_q\}$ of SN-sets let $T\nabla\mathcal{S}$ 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. The above theorem implies that, after possibly splitting one SN-set, we can replace each SN-set by a single leaf and the problem then essentially reduces to constructing a simple level-1 or -2 network for the induced triplet set. Given that there is at most one maximal SN-set that needs to be split into two subsets, we can simply try splitting each maximal SN-set of T in turn, as well as considering the case where no maximal SN-set of T is split. There are only $O(n)$ maximal SN-sets. The following lemma tells us how to split the chosen maximal SN-set into two subsets.

Lemma 3. *Let T be a dense set of triplets and N' a network with the properties described in Theorem 2. Suppose T contains a maximal SN-set X which occurs as the union of the sets S_1 and S_2 of leaves below two highest cut-arcs. Then $T|X$ contains precisely two maximal SN-sets and these are S_1 and S_2.*

The general outline of our algorithm LEVEL2, which constructs level-2 networks from dense triplet sets, is as follows. First we compute the maximal SN-sets. If there are precisely two maximal SN-sets then we recursively create two level-2 networks for the two maximal SN-sets and connect their roots to a new root. Otherwise, we try splitting each maximal SN-set in turn and we try the case where no maximal SN-set is split. If $\mathcal{S}$ is the obtained set of SN-sets then we compute the induced set of triplets $T\nabla\mathcal{S}$ and try to construct a simple level-1 or -2 network N consistent with $T\nabla\mathcal{S}$ using algorithm SL2. We recursively create level-2 networks for each SN-set in $\mathcal{S}$ and replace each leaf of N by the corresponding, recursively created, level-2 network.

It can furthermore be proven that if it is necessary to split an SN-set X then the simple level-2 network must be of type $2c$ and X must be below the

two reticulation leaves. Exploiting this fact we can prove that LEVEL2 can be implemented to run in time $O(n^8)$, which is equal to $O(|T|^{\frac{8}{3}})$. A simplified version of the algorithm is displayed below.

Algorithm 2. LEVEL2 (sketch)

```
compute the set SN of maximal SN-sets of T
if |SN| = 2 then
    N consists of a root connected to two leaves: the elements of SN
else
    if T∇SN is consistent with a simple level-1 network then
        let N be such a network
    else if T∇SN is consistent with a simple level-2 network then
        let N be such a network
    else
        for X ∈ SN do
            compute the set of maximal SN-sets SN' of T|X
            if |SN'| = 2 then
                S := SN \ {X} ∪ SN'
                if T∇S is consistent with a simple level-2 network of type 2c where the
                elements of SN' are the two reticulation leaves then
                    let N be such a network
replace each leaf X of N by a recursively created level-2 network for T|X.
```

Theorem 3. *Algorithm LEVEL2 constructs, in $O(|T|^{\frac{8}{3}})$ time, a level-2 network consistent with a dense set of triplets T if and only if such a network exists.*

4 Experimental Results

The algorithm from the previous section has been implemented in Java and applied to experimental data. The implementation was made publicly available [21]. The data of this application consists of sequences from different isolates of the yeast *Cryptococcus gattii*. This yeast is potentially dangerous and an ongoing outbreak on the Westcoast of Canada [20], which started in 1999, has caused many infections and even some fatalities. We have constructed a phylogenetic network for these isolates as a tool to find the origin of this *C. gattii* outbreak. We have blinded the names of the isolates here since the biological part of the research has not yet reached a conclusion.

Given the *C. gattii* sequences we have constructed a set of triplets as follows. Firstly, all identical sequences are combined into a single *sequence type (ST)*. One of the sequence types (that is only distantly related to the others) is used as an outgroup and we have applied the Maximum Likelihood method of PHYML [7] to each subset of four sequence types that includes the outgroup. Each output tree of PHYML gives us one input triplet for our algorithm LEVEL2. Running our algorithm for all ST's tells us that there exists no level-2 network consistent with all triplets. Therefore, we have applied our algorithm to a set containing as many

ST's as possible (where certain important ST's get priority over others) without destroying level-2-realisability. This set has been found by searching through all subsets. Given this subset and all triplets the execution of the algorithm LEVEL2 took 0.8 seconds on a Pentium IV, 3 GHz PC with 1GB memory. The resulting level-2 network is displayed in Figure 5. This network is consistent with all 1330 triplets that were generated over this set of taxa. The figure displays both a reticulate pattern and a dichotomous and tree-like structure. Our method is able to differentiate and visualise these.

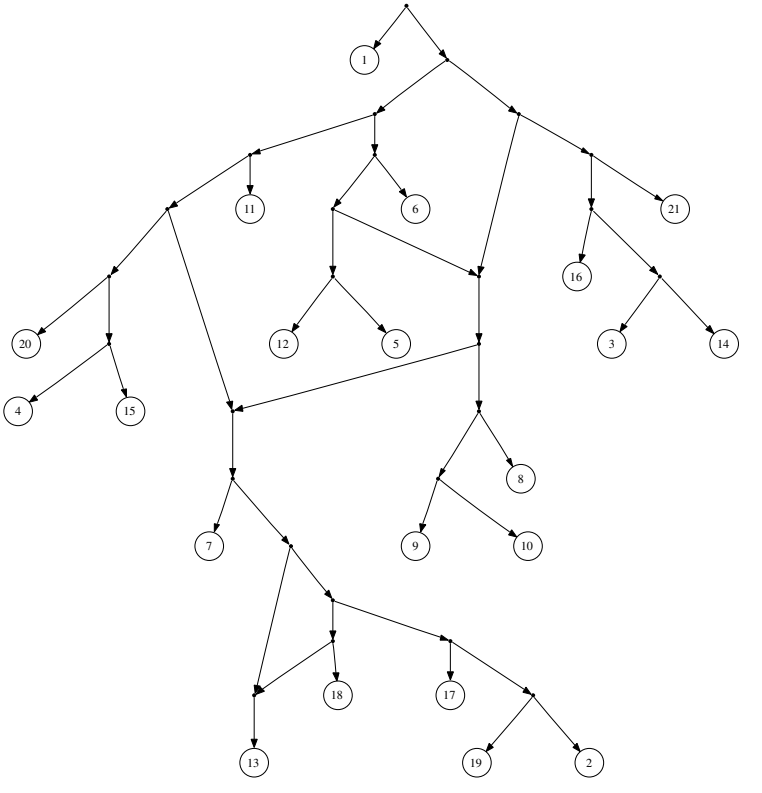

Fig. 5. The network constructed by the algorithm LEVEL2 for the triplets based on the yeast data set

A great advantage of our algorithm is that it is extremely fast. However, for data for which not all triplets can be found accurately or for which there are many reticulations that do not fit into a level-2 network, it might only be possible to find a phylogenetic network for a subset of the taxa. On the positive side, even in such cases, our level-2 networks could possibly give a better representation of reality than a tree or a level-1 network.

5 Conclusion and Open Questions

In this paper we have shown that in polynomial time we can decide whether a dense triplet set is consistent with a level-2 network, and if so construct such a network. In this way we have brought more complex, interwoven forms of evolution within reach of triplet methods. The described method has shown to be useful in practice. There remain, of course, many open questions and challenges, which we briefly list here.

1. **Applicability.** First practical experiments, one of which we reported in Section 4, show that the implementation we have made is fast and accurate. It remains interesting to test it on other phylogenetic data. We also wonder in how far the critique from certain parts of the community on the validity of many quartet-based methods is also relevant here. This critique in essence rests on the argument that it is in practice far harder to generate high-quality input quartets than is often claimed. The *short quartet method* [5] has been discussed as a way of addressing this critique. This debate needs to be addressed in the context of this paper.
2. **Complexity.** Is the non-dense level-k problem NP-hard for all fixed $k \geq 1$? Is the dense version polynomial-time solvable for all fixed k? In this regard it would be helpful to generalise Theorem 2, which captures the behaviour of SN-sets, and the algorithm that constructs simple level-2 networks. Generalising Theorem 2 will probably be difficult, because it is at this moment not clear whether the technique of "pushing" maximal SN-sets below cut-arcs generalises to level-3 and higher. It is also interesting to see how far the running time of our algorithm can be improved and/or how far this is necessary for further applications. At the moment the running time is $O(|T|^{\frac{8}{3}})$, which seems fast enough in practice. Finally, we would like to know the computational complexity of computing the smallest k for which there exists a level-k network that is consistent with a dense set of triplets.
3. **Bounds.** In [16] the authors determine constructive lower and upper bounds on the value τ for which the following statement is true: for each set of triplets T, not necessarily dense, there exists some level-1 network N which is consistent with at least $\tau|T|$ triplets in T. It is interesting to explore this question for level-2 networks and higher.
4. **Building all networks.** It is not clear whether it is possible to adapt our algorithm to generate *all* level-2 networks consistent with the input triplet set. If so, then such an adaptation could (even in the case that exponentially many networks are produced) be very useful for comparing the plausibility and/or relative similarity of the various solutions.
5. **Properties of constructed networks.** Under what conditions on the triplet set T is there only one network N consistent with T? Under what conditions does T permit some solution N such that the set of all triplets consistent with N, is exactly equal to T? These questions are also valid for level-1 networks.
6. **Different triplet restrictions.** Density is only one of very many possible restrictions on the input triplets. Interesting alternatives are what we have

named *minimal density* and *extreme density*. A minimally dense triplet set has exactly one triplet for every combination of three leaves. In the extreme dense case we assume that the set of input triplets is exactly equal to the set of triplets consistent with some network.

7. **Confidence.** At the moment all input triplets are assumed to be correct. Is there scope for attaching a confidence measure to each input triplet, and optimising on this basis? This is also related to the problem of ensuring that certain triplets are *excluded* from the output network, as explored in [9].
8. **Exponential-time exact algorithms.** It could be interesting, and useful, to develop exponential-time exact algorithms for solving the NP-hard problems for non-dense triplet sets.

Acknowledgements. We thank Katharina Huber for her useful ideas and many interesting discussions.

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. Bryant, D.: Building Trees, Hunting for Trees, and Comparing Trees: Theory and Methods in Phylogenetic Analysis, Ph.D. thesis, University of Canterbury, Christchurch, New Zealand (1997)
3. Bryant, D., Steel, M.: Constructing Optimal Trees from Quartets. Journal of Algorithms 38(1), 237–259 (2001)
4. Chan, H.-L., Jansson, J., Lam, T.W., Yiu, S.-M.: Reconstructing an Ultrametric Galled Phylogenetic Network from a Distance Matrix. Journal of Bioinformatics and Computational Biology 4(4), 807–832 (2006)
5. Erdös, P.L., Steel, M.A., Szekely, L.A., Warnow, T.: A few logs suffice to build (almost) all trees (Part II). Theoretical Computer Science 221(1), 77–118 (1999)
6. Gąsieniec, L., Jansson, J., Lingas, A., Östlin, A.: On the complexity of constructing evolutionary trees. Journal of Combinatorial Optimization 3, 183–197 (1999)
7. Guindon, S., Gascuel, O.: A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood. Systematic Biology 52(5), 696–704 (2003)
8. Gusfield, D., Eddhu, S., Langley, C.: Optimal, efficient reconstruction of phylogenetic networks with constrained recombination. Journal of Bioinformatics and Computational Biology 2, 173–213 (2004)
9. 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)
10. Holder, M., Lewis, P.O.: Phylogeny estimation: Traditional and bayesian approaches. Nature Reviews Genetics 4, 275–284 (2003)
11. Huson, D.H., Bryant, D.: Application of Phylogenetic Networks in Evolutionary Studies. Molecular Biology and Evolution 23(2), 254–267 (2006)
12. Huson, D.H., Klöpper, T.H.: Beyond Galled Trees - Decomposition and Computation of Galled Networks. In: Speed, T., Huang, H. (eds.) RECOMB 2007. LNCS (LNBI), vol. 4453, pp. 211–225. Springer, Heidelberg (2007)

13. Iersel, L.J.J. van, Keijsper, J.C.M., Kelk, S.M., Stougie, L.: Constructing level-2 phylogenetic networks from triplets (preprint, 2007), http://arxiv.org/abs/0707.2890
14. Jansson, J.: On the complexity of inferring rooted evolutionary trees. In: proceedings of GRACO 2001, ENDM 7, pp. 121–125. Elsevier, Amsterdam (2001)
15. Jansson, J., Ng, J.H.-K., Sadakane, K., Sung, W.-K.: Rooted maximum agreement supertrees. Algorithmica 43, 293–307 (2005)
16. 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)
17. Jansson, J., Sung, W.-K.: Inferring a Level-1 Phylogenetic Network from a Dense Set of Rooted Triplets. In: Chwa, K.-Y., Munro, J.I.J. (eds.) COCOON 2004. LNCS, vol. 3106, pp. 462–471. Springer, Heidelberg (2004)
18. 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)
19. Jiang, T., Kearney, P.E., Li, M.: A Polynomial Time Approximation Scheme for Inferring Evolutionary Trees from Quartet Topologies and Its Application. SIAM Journal on Computing 30(6), 1942–1961 (2000)
20. Kidd, S., Hagen, F., Tscharke, R., Huynh, M., Bartlett, K., Fyfe, M., MacDougall, L., Boekhout, T., Kwon-Chung, K.J., Meyer, W.: A rare genotype of Cryptococcus gattii caused the Cryptococcosis outbreak on Vancouver Island (British Columbia, Canada). In: Proceedings of the National Academy of Sciences of the United States of America, vol. 101, pp. 17258–17263 (2004)
21. LEVEL2: A fast method for constructing level-2 phylogenetic networks from dense sets of rooted triplets, http://homepages.cwi.nl/~kelk/level2triplets.html
22. Makarenkov, V., Kevorkov, D., Legendre, P.: Phylogenetic Network Reconstruction Approaches. In: Applied Mycology and Biotechnology. International Elsevier Series 6, Bioinformatics, vol. 6, pp. 61–97 (2006)
23. 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)
24. Semple, C., Steel, M.: Phylogenetics. Oxford University Press, Oxford (2003)
25. Steel, M.: The complexity of reconstructing trees from qualitative characters and subtrees. Journal of Classification 9, 91–116 (1992)
26. Wu, B.Y.: Constructing the maximum consensus tree from rooted triples. Journal of Combinatorial Optimization 8, 29–39 (2004)