

Worst-case optimal approximation algorithms for maximizing triplet consistency within phylogenetic networks*

Jaroslav Byrka^{1,2}, Pawel Gawrychowski⁴, Katharina T. Huber³, Steven Kelk¹,

¹ Centrum voor Wiskunde en Informatica,
Kruislaan 413, NL-1098 SJ Amsterdam, Netherlands
{J.Byrka,S.M.Kelk}@cwi.nl

² Eindhoven University of Technology,
P.O. Box 513, 5600 MB Eindhoven, The Netherlands

³ School of Computing Sciences, University of East Anglia,
Norwich, NR4 7TJ, United Kingdom
katharina.huber@cmp.uea.ac.uk

⁴ Institute of Computer Science, University of Wrocław,
ul. Joliot-Curie 15, 50-383 Wrocław, Poland

Abstract This article concerns the following question arising in computational evolutionary biology. For a given subclass of phylogenetic networks, what is the maximum value of $0 \leq p \leq 1$ such that for every input set T of rooted triplets, there exists some network $\mathcal{N}(T)$ from the subclass such that at least $p|T|$ of the triplets are consistent with $\mathcal{N}(T)$? Here we prove that the set containing all triplets (the full triplet set) in some sense defines p , and moreover that any network \mathcal{N} achieving fraction p' for the full triplet set can be converted in polynomial time into an isomorphic network $\mathcal{N}'(T)$ achieving fraction $\geq p'$ for an arbitrary triplet set T . We demonstrate the power of this result for the field of phylogenetics by giving worst-case optimal algorithms for level-1 phylogenetic networks (a much-studied extension of phylogenetic trees), improving considerably upon the $5/12$ fraction obtained recently by Jansson, Nguyen and Sung in [14]. For level-2 phylogenetic networks we show that $p \geq 0.61$. We note that all the results in this article also apply to weighted triplet sets.

1 Introduction

One of the most commonly encountered problems in computational evolutionary biology is to plausibly infer the evolutionary history of a set of species, often abstractly modelled as a tree, using obtained biological data. Existing algorithms for directly constructing such a tree do not scale well (in terms of running time) and this has given rise to *supertree* methods: first infer trees for small subsets of the species and then puzzle them together into a bigger tree such that in some well-defined sense the information in the subset trees is preserved [5]. In the fundamental case where the subsets in question each contain exactly three species - subsets of two or fewer species cannot convey information - we speak of *rooted triplet* methods.

In recent years improved understanding of the complex mechanisms driving evolution has stimulated interest in reconstructing evolutionary *networks* [9][16][18][23]. Such structures are more general than trees and allow us to capture the phenomenon of reticulate evolution i.e. non tree-like evolution caused by processes such as hybrid speciation and horizontal gene transfer. A natural abstraction of reticulate evolution, used already in several papers, is to permit *recombination vertices*, vertices with indegree greater than one. Informally a *level- k phylogenetic network* is an evolutionary network in which each biconnected component contains at most k such recombination vertices. Phylogenetic trees form the base: they are level-0 networks. The higher the level of a network, the more

* This research has been partly funded by the Dutch BSIK/BRICKS project, and by the EU Marie Curie Research Training Network ADONET, Contract No MRTN-CT-2003-504438.

intricate the pattern of reticulate evolution that it can accommodate. Note that phylogenetic networks can also be useful for visualising two or more competing hypotheses about tree-like evolution.

Various authors have already studied the problem of constructing phylogenetic trees (and more generally networks) which are consistent with an input set of rooted triplets. Aho et al [1] showed a simple polynomial-time algorithm which, given a set of rooted triplets, finds a phylogenetic tree consistent with all the triplets, or shows that no such tree exists. For the equivalent problem in level-1 and level-2 networks the problem becomes NP-hard [12][14], although the problem becomes polynomial-time solvable if the input triplets are *dense* i.e. if there is at least one triplet in the input for each subset of three species [12][15].

Several authors have considered algorithmic strategies of use when the algorithms from [1] and [15] fail to find a tree or network. Gąsieniec et al [10] gave a polynomial-time algorithm which always finds a tree consistent with at least $1/3$ of the (weighted) input triplets, and furthermore showed that $1/3$ is best possible when all possible triplets on n species (the *full triplet set*) are given as input. On the negative side, [6][13][24] showed that it is NP-hard to find a tree consistent with a maximum number of input triplets. In the context of level-1 networks, [14] gave a polynomial-time algorithm which produces a level-1 network consistent with at least $5/12 \approx 0.4166$ of the input triplets. They also described an upper-bound, which is a function of the number of distinct species n in the input, on the percentage of input triplets that can be consistent with a level-1 network. As in [10] this upper bound is tight in the sense that it is the best possible for the full triplet set on n species. They computed a value of n for which their upper bound equals approximately 0.4883, showing that in general a fraction better than this is not possible. The apparent convergence of this bound from above to 0.4880... begs the question, however, whether a fraction better than $5/12$ is possible for level-1 networks, and whether the full triplet set is in general always the worst-case scenario for such fractions.

In this paper we answer these questions in the affirmative, and in fact we give a much stronger result. In particular, we develop a probabilistic argument that (as far as such fractions are concerned) the full triplet set is indeed always the worst possible case, irrespective of the type of network being studied (Proposition 1, Corollary 1). Furthermore, by using a generic, derandomized polynomial-time (re)labelling procedure we can convert a network \mathcal{N} which achieves a fraction p' for the full triplet set into an isomorphic network $\mathcal{N}'(T)$ that achieves a fraction $\geq p'$ for a given input triplet set T (Theorem 1). In this way we can easily use the full triplet set to generate, for any network structure, a lower bound on the fraction that can be achieved for arbitrary triplet sets within such a network structure. The derandomization we give is fully general (with a highly optimized running time) and leads immediately to a simple extension of the $1/3$ result from [10]. For level-1 networks we use the derandomization to give a polynomial-time algorithm which achieves a fraction *exactly equal* to the level-1 upper-bound identified in [14], and which is thus worst-case optimal for level-1 networks. We formally prove that this achieves a fraction of at least 0.48 for all n . Finally, we demonstrate the flexibility of our technique by proving that for level-2 networks (see [12]) we can, for any triplet set T , find in polynomial time a level-2 network consistent with at least a fraction 0.61 of the triplets in T (Theorem 3).

We emphasize that in this article we are optimizing (and thus defining worst-case optimality) with respect to $|T|$, the number of triplets in the input, not $Opt(T)$, the size of the optimal solution for that specific T . The latter formulation we call the MAX variant of the problem. The fact that $Opt(T)$ is always bounded above by $|T|$ implies that an algorithm that obtains a fraction

p' of the input T is trivially also a p' -approximation for the corresponding MAX problem. Better approximation factors for the MAX problem might, however, be possible. We discuss this further in Section 7.

The results in this article are given in terms of unweighted triplet sets. A natural extension, especially in phylogenetics, is to attach a weight to each triplet $t \in T$ i.e. a value $w(t) \in \mathbb{Q}_{\geq 0}$ denoting the relative importance of (or confidence in) t . In this weighted version of the problem fractions are defined relative to the total weight of T (defined as the sum of the weights of all triplets in T), not to $|T|$. It is easy to verify that all the results in this article also hold for the weighted version of the problem.

2 Definitions

A *phylogenetic network* (*network* for short) \mathcal{N} on species set X is defined as a pair (N, γ) where N is the *network topology* (*topology* for short) and γ is a *labelling* of the topology. The topology is 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 (*recombination vertices*) or indegree 1 and outdegree 0 (*leaves*). A labelling is a bijective mapping from the leaf set of N (denoted L^N) to X . Let $n = |X| = |L^N|$.

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 $k \in \mathbb{N}$ recombination vertices.

We define *phylogenetic trees* to be the class of level-0 networks.

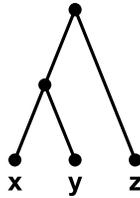


Figure 1. One of the three possible triplets on the set of species $\{x, y, z\}$. Note that, as with all figures in this article, all arcs are assumed to be directed downwards, away from the root.

The unique *rooted triplet* (*triplet* for short) on a species set $\{x, y, z\} \subseteq X$ 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 $yx|z$). For any set T of triplets define $X(T)$ as the union of the species sets of all triplets in T .

Definition 2. A triplet $xy|z$ is consistent with a network \mathcal{N} (interchangeably: \mathcal{N} is consistent with $xy|z$) if \mathcal{N} contains a subdivision of $xy|z$, i.e. if \mathcal{N} contains vertices $u \neq v$ and pairwise internally vertex-disjoint paths $u \rightarrow x$, $u \rightarrow y$, $v \rightarrow u$ and $v \rightarrow z$ ⁵.

⁵ Where it is clear from the context, as in this case, we may refer to a leaf by the species that it is mapped to.

By extension, a set of triplets T is consistent with a network \mathcal{N} (interchangeably: \mathcal{N} is consistent with T) iff, for all $t \in T$, t is consistent with \mathcal{N} . Checking consistency can be done in polynomial time, see Lemmas 2 and 3.

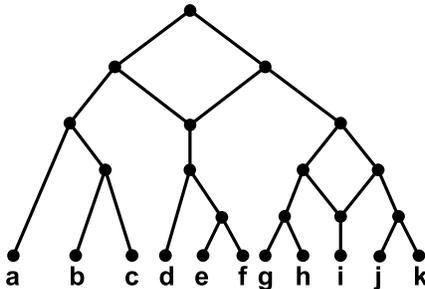


Figure 2. An example of a level-1 network.

3 Labelling a network topology

Suppose we are given a topology N with n leaves, and a set T of triplets where $L^N = \{l_1, l_2, \dots, l_n\}$ and $X = X(T) = \{x_1, x_2, \dots, x_n\}$. The specific goal of this section is to create a labelling γ such that the number of triplets from T consistent with (N, γ) , is maximized.

Let $f(N, \gamma, T)$ denote the fraction of T that is consistent with (N, γ) .

Consider the special set $T_1(n)$, the *full triplet set*, of all the possible $3\binom{n}{3}$ triplets with leaves labelled from $\{x_1, x_2, \dots, x_n\}$. Observe that for this triplet set the number of triplets consistent with a phylogenetic network (N, γ) does not depend on the labelling γ . We may thus define $\#N = f(N, \gamma, T_1(n)) = f(N, T_1(n))$ by considering any arbitrary, fixed labelling γ . (Note that $\#N = 1/3$ if N is a tree topology.)

We will argue that the triplet set $T_1(n)$ is the worst-case input for maximizing $f(N, \gamma, T)$ for any fixed topology N on n leaves. In particular we prove the following:

Proposition 1. *For any topology N with n leaves and any set of triplets T , if the labelling γ is chosen uniformly at random, then the quantity $f(N, \gamma, T)$ is a random variable with expected value $E(f(N, \gamma, T)) = \#N$.*

Proof. Consider first the full triplet set $T_1(n) = \{t_1, t_2, \dots, t_{3\binom{n}{3}}\}$ and an arbitrary fixed labelling γ_0 . By labelling N we fix the *position* of each of the triplets in N . Formally, a position of a triplet $t = xy|z$ (with respect to γ_0) is a triplet $p = \gamma_0^{-1}(t) = \gamma_0^{-1}(x)\gamma_0^{-1}(y)|\gamma_0^{-1}(z)$ on the leaves of N . We may list possible positions for a triplet in N as those corresponding to $t_1, t_2, \dots, t_{3\binom{n}{3}}$ in (N, γ_0) . Since a $\#N$ fraction of $t_1, t_2, \dots, t_{3\binom{n}{3}}$ is consistent with (N, γ_0) , a $\#N$ fraction of these positions makes the triplet consistent. Now consider a single triplet $t \in T$ and a labelling γ that is chosen randomly from the set Γ of $n!$ possible bijections from L^N to X . Observe, that for each $t_i \in T_1(n)$, exactly $2 \cdot (n-3)!$ labellings $\gamma \in \Gamma$ make triplet t have the same position in (N, γ) as t_i has in (N, γ_0) (the factor of 2 comes from the fact that we think of $xy|z$ and $yx|z$ as being the same triplet). Any single labelling occurs with probability $\frac{1}{n!}$, hence triplet t takes any single position with probability $\frac{2 \cdot (n-3)!}{n!} = \frac{1}{3\binom{n}{3}}$.

Since for an arbitrary $t \in T$ each of the $3 \cdot \binom{n}{3}$ positions have the same probability and $\#N$ of them make t consistent, the probability of t being consistent with (N, γ) is $\#N$. The expectation is thus that a fraction $\#N$ of the triplets in T are consistent with (N, γ) . \square

From the expected value of a random variable we may conclude the existence of a realization that attains at least this value.

Corollary 1. *For any topology N and any set of triplets T there exists a labelling γ_0 such that $f(N, \gamma_0, T) \geq \#N$.*

We may deterministically find such a γ_0 by derandomizing the argument in a greedy fashion, using the method of conditional expectation (see e.g. [19]). In particular, we will show the following.

Theorem 1. *For any topology N and any triplet set T , a labelling γ_0 such that $f(N, \gamma_0, T) \geq \#N$ can be found in time $O(m^3 + n|T|)$, where m and n are the numbers of vertices and leaves of N .*

We use standard arguments from conditional expectation to prove that the solution is of the desired quality. It is somewhat more sophisticated to optimize the running time of this derandomization procedure. We therefore dedicate the following subsection to the proof of Theorem 1.

3.1 An optimized derandomization procedure

We start with a sketch of the derandomization procedure.

1. $\Gamma \leftarrow$ set of all possible labellings.
2. while there is a leaf l whose label is not yet fixed, do:
 - for every species x that is not yet used
 - let Γ_x be the set of labellings from Γ where l is labeled by x
 - compute $E(f(N, \gamma_x, T))$, where γ_x is chosen randomly from Γ_x
 - $\Gamma \leftarrow \Gamma_x$ s.t. $x = \operatorname{argmax}_x E(f(N, \gamma_x, T))$
3. return $\gamma_0 \leftarrow$ the only element of Γ .

The following lemma shows that the solution produced is of the desired quality.

Lemma 1. $f(N, \gamma_0, T) \geq \#N$.

Proof. By Proposition 1 the initial random labelling γ has the property $E(f(N, \gamma, T)) = \#N$. It remains to show that this expectation is not decreasing when labels of leaves get fixed during the algorithm. Consider a single update $\Gamma \leftarrow \Gamma_x$ of the range of the random labelling. By the choice of the leaf l to get a fixed label we choose a partition of Γ into blocks Γ_x . The expectation $E(f(N, \gamma, T))$ is an average of $f(N, \gamma, T)$ over Γ , and at least one of the blocks Γ_x of the partition has this average at least as big as the total average. Hence, by the choice of Γ_x with the highest expectation of $f(N, \gamma_x, T)$, we get $E(f(N, \gamma_x, T)) \geq E(f(N, \gamma, T))$. \square

We will now propose an efficient implementation of the derandomization procedure. Since the procedure takes a topology N as an input, we need to express the running time of this procedure in terms of the size of N . We will use $m = |V(N)|$ for the number of vertices of N , and n for the number of leaves.

We will need a generalized definition of consistency. In Definition 2 it is assumed that x, y and z are leaves of the network \mathcal{N} . In the following we will use a predicate $\text{consistent}(x, y, z)$ defined on vertices of \mathcal{N} which coincides with Definition 2 when x, y and z are leaves. The predicate $\text{consistent}(x, y, z)$ is defined to be true if \mathcal{N} contains vertices $u \neq v$ and pairwise internally vertex-disjoint paths $u \rightarrow x$, $u \rightarrow y$, $v \rightarrow u$ and $v \rightarrow z$; paths are allowed to be of length 0 (e.g. $u = x$), but vertices x, y and z need to be all different. We begin with a consistency checking algorithm.

Lemma 2. *Given a network \mathcal{N} with m vertices, we can preprocess it in time $O(m^3)$ so that given any three vertices x, y, z we can check whether $xy|z$ is consistent with \mathcal{N} in time $O(1)$.*

Proof. Assume that vertices of \mathcal{N} are numbered according to a topological order so that whenever (u, v) is an edge, $u < v$. Observe that $\text{consistent}(x, y, z)$ can be expressed in terms of $\text{consistent}(x', y', z')$ where $\max(x', y', z') < \max(x, y, z)$ as follows:

1. $x, y < z$. Then $\text{consistent}(x, y, z)$ holds iff for some $z' \neq x, y$ (z', z) is an edge and $\text{consistent}(x, y, z')$ holds.
2. $x, z < y$. Then $\text{consistent}(x, y, z)$ holds iff (x, y) is an edge and $\text{join}(x, z)$ holds or for some $y' \neq x, z$ (y', y) is an edge and $\text{consistent}(x, y', z)$ holds.
3. $y, z < x$. Then $\text{consistent}(x, y, z)$ holds iff (y, x) is an edge and $\text{join}(y, z)$ holds or for some $x' \neq y, z$ (x', x) is an edge and $\text{consistent}(x', y, z)$ holds.

where $\text{join}(x, z)$ is a predicate stating that \mathcal{N} contains $t \neq x$ and internally vertex-disjoint paths $t \rightarrow x$ and $t \rightarrow z$. It can be verified in linear time, for example by checking if there is a path $r \rightarrow z$ (where r is the root) after removing x from the network.

So, determining all consistent triplets can be done by calculating all $\text{consistent}(x, y, z)$. The number of operations we have to perform for each x, y, z is not greater than sum of indegrees of those vertices which makes the overall complexity $O(m^3 + m^2|E(\mathcal{N})|) = O(m^3)$. \square

Although we do not use the result in this article, we note that the above lemma can be strengthened to obtain the following, which is proved in the appendix.

Lemma 3. *Given a level- k network \mathcal{N} , we can preprocess it in time $O(m + mk^2)$ so that given any three vertices x, y, z we can check whether $xy|z$ is consistent with \mathcal{N} in time $O(1)$.*

Armed with Lemma 2 (or Lemma 3) we are now ready to proceed with the derandomization. Assume that we assign labels to leaves in order of their position in the topological ordering of Lemma 2; for simplicity let us refer to the leaves as $1, 2, \dots, n$. The most time-consuming part of the algorithm is calculating the probability that a given triplet $t = ab|c$ is consistent with a random labelling having already specified the labels of leaves $1, 2, \dots, k-1$. Let γ be this partial labelling. We need to consider six cases:

1. Labels a, b, c have not been assigned yet. The probability is then simply the number of consistent triplets $xy|z$ with $k \leq x, y, z$ divided by $3^{\binom{n-k+1}{3}}$.
2. $\gamma(x) = a$ but both b and c have not been assigned yet. The probability is the number of $k \leq y, z$ such that $xy|z$ is a consistent triplet divided by $2^{\binom{n-k+1}{2}}$.
3. $\gamma(z) = c$ but both a and b have not been assigned yet. The probability is the number of $k \leq x < y$ such that $xy|z$ is a consistent triplet divided by $\binom{n-k+1}{2}$.
4. $\gamma(x) = a, \gamma(y) = b$ but c has not been assigned yet. The probability is the number of $k \leq z$ such that $xy|z$ is a consistent triplet divided by $n - k + 1$.
5. $\gamma(x) = a, \gamma(z) = c$ but b has not been assigned yet. The probability is the number of $k \leq y$ such that $xy|z$ is a consistent triplet divided by $n - k + 1$.
6. $\gamma(x) = a, \gamma(y) = b, \gamma(z) = c$. The probability is 0 or 1, depending on whether $xy|z$ is consistent or not.

In the first five cases, we can do rather better than simply counting all the $xy|z$ each time from scratch. Define:

$$\begin{aligned} \text{count3}(k) &:= \#\langle x,y,z \rangle \text{ such that } k \leq x,y,z \text{ and } x < y \text{ and } \text{consistent}(x,y,z) \\ \text{count2x}(k,x) &:= \#\langle y,z \rangle \text{ such that } k \leq y,z \text{ and } \text{consistent}(x,y,z) \\ \text{count2z}(k,z) &:= \#\langle x,y \rangle \text{ such that } k \leq x < y \text{ and } \text{consistent}(x,y,z) \\ \text{count1xy}(k,x,y) &:= \#\langle z \rangle \text{ such that } k \leq z \text{ and } \text{consistent}(x,y,z) \\ \text{count1xz}(k,x,z) &:= \#\langle y \rangle \text{ such that } k \leq y \text{ and } \text{consistent}(x,y,z) \end{aligned}$$

It is easy to see that we can compute all the above values in time $O(m^3)$ as follows:

$$\begin{aligned} \text{count3}(k) &= \text{count3}(k+1) + \text{count2x}(k+1,k) + \text{count2z}(k+1,k) \\ \text{count2x}(k,x) &= \text{count2x}(k+1,x) + \text{count1xz}(k+1,x,k) + \text{count1xy}(k+1,x,k) \\ \text{count2z}(k,z) &= \text{count2z}(k+1,z) + \text{count1xz}(k+1,k,z) \\ \text{count1xy}(k,x,y) &= \text{count1xy}(k+1,x,y) + \text{consistent}(x,y,k) \\ \text{count1xz}(k,x,z) &= \text{count1xz}(k+1,x,z) + \text{consistent}(x,k,z) \end{aligned}$$

Having preprocessed the above values, we can calculate each probability in time $O(1)$, giving us a total running time of $O(m^3 + n^2|T|)$. It turns out, however, that we can do slightly better. As it currently stands, for each leaf and each unused label we calculate the expected number of consistent triplets after assigning this label to this leaf separately. To further improve the complexity of the running time we can try to do all such calculations at once (for a fixed leaf): for each triplet and for each unused label we calculate the probability that this triplet is consistent after we use this label. (In other words, we switch from a leaf-label-triplet loop nesting to leaf-triplet-label). Then it turns out that those probabilities are the same for almost all unused labels. More formally:

Lemma 4. *Let Γ be the set of labellings that assign $\gamma(i)$ to leaf i for each $i = 1, 2, \dots, k-1$ and Γ_x be the subset of Γ that assign x to leaf k . We can find x for which $E(f(N, \gamma_x, T))$ is maximum in time $O(|T|)$ assuming the above preprocessing.*

Proof. Let $e_x = E(f(N, \gamma_x, T))$. Each such e_x is a sum of probabilities corresponding to the elements of T . We will start with all e_x equal 0 and consider triplets one by one. For each triplet t we will increase the appropriate values of e_x by the corresponding probabilities. Again we should consider six cases depending on how $t = ab|c$ looks like. Let $A = \frac{1}{3\binom{n-k+1}{3}}$, $B = \frac{1}{\binom{n-k+1}{2}}$ and $C = \frac{1}{n-k+1}$:

1. Labels a, b, c have not been assigned yet. We should increase e_a and e_b by $\text{count2x}(k+1,k)\frac{B}{2}$, e_c by $\text{count2z}(k+1,k)B$ and the remaining e_l by $\text{count3}(k+1)A$.
2. $\gamma(x) = a$ but both b and c have not been assigned yet. We should increase e_b by $\text{count1xy}(k+1,a,k)C$, e_c by $\text{count1xz}(k+1,x,k)C$ and the remaining e_l by $\text{count2x}(k+1,x)\frac{B}{2}$.
3. $\gamma(z) = c$ but both a and b have not been assigned yet. We should increase e_a and e_b by $\text{count1xz}(k+1,k,z)C$ and the remaining e_l by $\text{count2z}(k,z)B$.
4. $\gamma(x) = a, \gamma(y) = b$ but c has not been assigned yet. We should increase e_c by $\text{consistent}(x,y,k)$ and the remaining e_l by $\text{count1xy}(k+1,x,y)C$.
5. $\gamma(x) = a, \gamma(z) = c$ but b has not been assigned yet. We should increase e_b by $\text{consistent}(x,k,z)$ and the remaining e_l by $\text{count1xz}(k+1,x,z)C$.
6. $\gamma(x) = a, \gamma(y) = b, \gamma(z) = c$. We should increase all e_l by $\text{consistent}(x,y,z)$.

The naïve implementation of the above procedure would require time $O(|T|n)$. We can improve it by observing that we are interested only in the relative increment of the different e_x , not in their actual values. So, instead of increasing all e_l with $l \notin S$ by some δ (for some $S \subseteq X$), we can decrease all e_l with $l \in S$ by this δ . Then processing each triplet takes time $O(1)$ as we only have to change at most 3 values of e_x .

□

We may now compose these lemmas into the following.

Proof (of Theorem 1). Consider the procedure sketched at the beginning of the subsection and implemented according to the above lemmas. By Lemma 1 it produces good labellings. By Lemma 2 and n applications of Lemma 4, it can be implemented to run in $O(m^3 + n|T|)$ time. □

3.2 Consequences of Theorem 1

The above theorem gives a new perspective on the problem of approximately constructing phylogenetic networks. From the algorithm of Gąsieniec et al. [10] we can always construct a phylogenetic tree that is consistent with at least 1/3 of the the input triplets. In fact, the trees constructed by this algorithm are very specific - they are always caterpillars. (A caterpillar is a phylogenetic tree such that, after removal of leaves, only a directed path remains.) Theorem 1 implies that not only caterpillars, but all possible tree topologies have the property, that given any set of triplets we may find in polynomial time a proper assignment of species into leaves with the guarantee that the resulting phylogenetic tree is consistent with at least a third of the input triplets.

The generality of Theorem 1 makes it meaningful not only for trees, but also for any other subclass of phylogenetic networks (e.g. for level- k networks). Let us assume that we have focussed our attention on a certain subclass of networks. Consider the task of designing an algorithm that for a given triplet set constructs a network from the subclass consistent with at least a certain fraction of the given triplets. A worst-case approach as described in this section will never give us a guarantee better than the maximum value of $\#N$ ranging over all topologies N in the subclass. Therefore, if we intend to obtain networks consistent with a big fraction of triplets and if our criteria is to maximize this fraction in the worst case, then our task reduces to finding topologies within the subclass that are good for the full triplet set. Theorem 1 potentially has a further use as a mechanism for comparing the quality of phylogenetic networks generated by other methods, because it provides lower bounds for the fraction of T that a given topology and/or subclass of topologies can be consistent with. (Although a fundamental problem in phylogenetics [2] [7] [8] [20] [17] the science of network comparison is still very much in its infancy.)

For level-0 networks (i.e. phylogenetic trees) the problem of finding optimal topologies for the full triplet set is simple: any tree is consistent with exactly 1/3 of the full triplet set. For level-1 phylogenetic networks a topology that is optimal for the full triplet set was constructed in [14]. We may use this network and Theorem 1 to obtain an algorithm that works for any triplet set and creates a network that is consistent with the biggest possible fraction of triplets in the worst case (see Section 4 for more details). For level-2 networks we do not yet know the optimal structure of a topology for the full triplet set, but we will show in Section 5 that we can construct a network that has a guarantee of being consistent with at least a fraction 0.61 of the input triplets.

4 Application to level-1 phylogenetic networks

4.1 A worst-case optimal polynomial-time algorithm for level-1 networks

In [14] it was shown how to construct a special level-1 topology $C(n)$, which we call a *galled caterpillar*⁶, such that $\#C(n) \geq \#N$ for all level-1 topologies N on n leaves. The existence of $C(n)$, which has a highly regular structure, was proven by showing that any other topology N can be transformed into $C(n)$ by local rearrangements that never decrease the number of triplets the associated network is consistent with. It was shown that $\#C(n) = S(n)/3\binom{n}{3}$, where $S(0) = S(1) = S(2) = 0$ and, for $n > 2$,

$$S(n) = \max_{1 \leq a \leq n} \left\{ \binom{a}{3} + 2 \binom{a}{2} (n-a) + a \binom{n-a}{2} + S(n-a) \right\}. \quad (1)$$

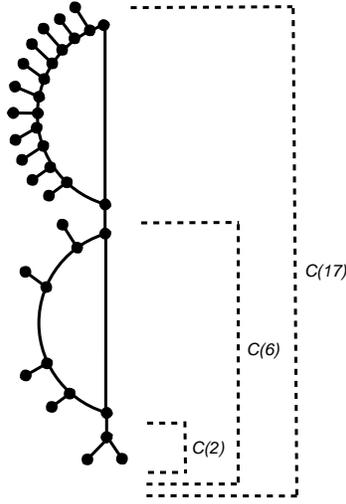


Figure 3. This is galled caterpillar $C(17)$. It contains two galls and ends with a tail of two leaves. $C(17)$ contains 11 leaves in the top gall because Equation 1 is maximised for $a = 11$.

In Figure 3 an example of a galled caterpillar is shown. All galled caterpillars on $n \geq 3$ leaves consist of one or more *galls* chained together in linear fashion and terminating in a tail of one or two leaves. Observe that the recursive structure of $C(n)$ mirrors directly the recursive definition of $S(n)$ in the sense that the value of a chosen at recursion level k is equal to the number of leaves found in the k th gall, counting downwards from the root. In the definition of $C(n)$ it is not specified how the a leaves at a given recursion level are distributed within the gall, but it is easy to verify that placing them all on one side of the gall (as shown in the figure) is sufficient.

Lemma 5. *Let T be a set of input triplets labelled by n species. Then in time $O(n^3 + n|T|)$ it is possible to construct a level-1 network \mathcal{N} , isomorphic to the galled caterpillar $C(n)$, consistent with at least a fraction $S(n)/3\binom{n}{3}$ of T .*

Proof. First we construct the level-1 topology $C(n)$. Using dynamic programming to compute all values of $S(n')$ for $0 \leq n' \leq n$ we can do this in time $O(n^2)$. Note that $C(n)$ contains in total $O(n)$ vertices. It remains only to choose an appropriate labelling of the leaves of $C(n)$, and this is achieved by substituting $C(n)$ for N in Theorem 1; this dominates the running time. \square

⁶ In [14] this is called a *caterpillar network*.

Note that, because $C(n)$ achieves the best possible fraction for the input $T_1(n)$, the fraction achieved by Lemma 5 is worst-case optimal for all n . Empirical experiments suggest that the function $S(n)/3\binom{n}{3}$ is strictly decreasing and approaches a horizontal asymptote of 0.4880... from above; for values of $n = 10^1, 10^2, 10^3, 10^4$ the respective ratios are 0.511..., 0.490..., 0.4882..., 0.4880.... It is difficult to formally prove convergence to 0.4880... so we prove a slightly weaker lower bound of 0.48 on this function. From this it follows that in all cases the algorithm described in Lemma 5 is guaranteed to produce a network consistent with at least a fraction 0.48 of T , improving considerably on the $5/12 \approx 0.4166$ fraction achieved in [14].

Lemma 6. $S(n)/3\binom{n}{3} > 0.48$ for all $n \geq 0$.

Proof. This can easily be computationally verified for $n < 116$, we have done this with a computer program written in Java [25]. To prove it for $n \geq 116$, assume by induction that the claim is true for all $n' < n$. Instead of choosing the value of a that maximises $S(n)$ we claim that setting a equal to $z = \lfloor 2n/3 \rfloor$ is sufficient for our purposes. We thus need to prove the following inequality:

$$\frac{\binom{z}{3} + 2\binom{z}{2}(n-z) + z\binom{n-z}{2} + S(n-z)}{3\binom{n}{3}} > 48/100.$$

Combined with the fact that, by induction, $S(n-z)/3\binom{n-z}{3} > 48/100$, it is sufficient to prove that:

$$\frac{\binom{z}{3} + 2\binom{z}{2}(n-z) + z\binom{n-z}{2} + 144/100\binom{n-z}{3}}{3\binom{n}{3}} > 48/100$$

Using Mathematica we rearrange the previous inequality to:

$$\frac{\lfloor 2n/3 \rfloor \left(22 + 9n + 33n^2 - 6(7 + 18n)\lfloor 2n/3 \rfloor + 86\lfloor 2n/3 \rfloor^2 \right)}{n(2 - 3n + n^2)} < 0$$

Taking $(2n/3) - 1$ as a lower bound on $\lfloor 2n/3 \rfloor$, and $2n/3$ as an upper bound, it can be easily verified that the above inequality is satisfied for $n \geq 116$. \square

To conclude this section we combine Lemmas 5 and 6 into the following Theorem.

Theorem 2. *Let T be a set of input triplets labelled by n species. In time $O(n^3 + n|T|)$ it is possible to construct a level-1 network \mathcal{N} consistent with at least a fraction $S(n)/3\binom{n}{3} > 0.48$ of T , and this is worst-case optimal.*

5 A lower bound for level-2 networks

Theorem 3. *Let T be a set of input triplets labelled by n species. It is possible to find in polynomial time a level-2 network $\mathcal{N}(T)$ such that $\mathcal{N}(T)$ is consistent with at least a fraction 0.61 of T .*

Proof. We prove this by induction. For $n < 16813$ we use a computational proof. For $n \geq 16813$ we use Mathematica to show that, assuming the induction base, a fraction 0.61 can be achieved by repeatedly chaining together a very basic type of level-2 network into some kind of “level-2 caterpillar”; the details are deferred to the appendix. \square

6 The complexity of optimisation

Given a topology N and a set of triplets T , the techniques described in this article guarantee to find a labelling γ such that $f(N, \gamma, T) \geq \#N$. It is natural to explore the complexity of finding, in polynomial time, (approximations to) an optimal labelling of N for a particular triplet set T . The observation below rules out (under standard complexity-theoretic assumptions) the existence of a *Polynomial-Time Approximation Scheme* (PTAS) for this problem. Secondly we observe that a PTAS for the problem MAX-LEVEL-0 (which carries the name MCTT in [24]) can also be ruled out. We discuss the consequences of this in the next section.

Problem: MAX-LEVEL-0-LABELLING

Input: A level-0 topology N (i.e. a topology of a phylogenetic tree) and a set T of rooted triplets.

Output: The maximum value of s such that there exists a labelling γ of N making (N, γ) consistent with at least s triplets from T .

Problem: MAX-LEVEL-0

Input: A set T of rooted triplets.

Output: The maximum value of s such that there exists a level-0 network \mathcal{N} (i.e. a phylogenetic tree) consistent with at least s triplets from T .

Observation 1 *MAX-LEVEL-0 and MAX-LEVEL-0-LABELLING are both APX-complete.*

Proof. By approximation-preserving reduction from MAXIMUM SUBDAG / LINEAR ORDERING. See appendix. \square

7 Conclusions and open questions

With Theorem 1 we have described a method which shows how, in polynomial time, good solutions for the full triplet set can be efficiently converted into equally good, or better, solutions for more general triplet sets. Where best-possible solutions are known for the full triplet set, this leads to worst-case optimal algorithms, as demonstrated by Theorem 2. An obvious next step is to use this method to generate algorithms (where possible worst-case optimal) for wider subclasses of phylogenetic networks. Finding the/an “optimal form” of level-2 networks for the full triplet set remains a fascinating open problem.

From a biological perspective (and from the perspective of understanding the relevance of triplet methods) it is also important to attach *meaning* to the networks that the techniques described in this paper produce. For example, we have shown how, for level-1 networks, we can always find a network isomorphic to a galled caterpillar which is consistent with at least a fraction 0.48 of the input. If we do this, does the location of the species within this galled caterpillar communicate any biological information? Also, what does it say about the relevance of triplet methods, and especially the level- k hierarchy, if we know *a priori* that a large fraction (already for level 2 more than 0.61) of the input can be made consistent with some network from the subclass? And, as discussed in Section 3.2, how far can the techniques described in this paper be used as a quality measure for networks produced by other algorithms?

As mentioned in the introduction, an algorithm guaranteed to find a network consistent with a fraction p' of the input trivially becomes a p' -approximation for the MAX variant of the problem (where we optimise not with respect to $|T|$ but with respect to the size of the optimal solution for T .) In fact, the best-known approximation factor for MAX-LEVEL-0 is $1/3$, a trivial extension

of the fact that $p = 1/3$ for trees [10]. On the other hand, the APX-hardness of this problem implies that an approximation factor arbitrarily close to 1 will not be possible. It remains a highly challenging open problem to determine whether better approximation factors can be obtained for the latter problem via some different approach. (We note that an approximation factor $f > 1/2$ would be a major breakthrough because, by the proof of Observation 1, this would improve upon the long-standing best-known approximation factor of $1/2$ for the problem LINEAR ORDERING [21].) Alternatively, there could be some complexity theoretic reason why approximation factors better than p (where p is optimal in our formulation) are not possible. Under strong complexity-theoretic assumptions the best approximation factor possible for MAX-3-SAT, for example, uses a trivial upper bound of all the clauses in the input [11], analogous perhaps to using $|T|$ as an upper bound.

8 Acknowledgements

We thank Leo van Iersel and Jesper Jansson for very helpful comments.

References

1. A.V. Aho, Y. Sagiv, T.G. Szymanski and J.D. Ullman, 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. M. Baroni, C. Semple, and M. Steel, A framework for representing reticulate evolution, *Annals of Combinatorics* 8(4), 391–408 (2004).
3. M. Bender and M. Farach-Colton, The LCA problem revisited, in *Proceedings of LATIN 2000: Theoretical Informatics*, LNCS 1776, 88–94 (2000).
4. M. Bender and M. Farach-Colton, The level ancestor problem simplified, *Theoretical Computer Science* 321(1), 5–12 (2004).
5. O. Bininda-Emonds, *Phylogenetic supertrees: combining information to reveal the tree of life*, Computational Biology Series 4, Kluwer (2004).
6. D. Bryant, Building trees, hunting for trees, and comparing trees: theory and methods in phylogenetic analysis, Ph.D. thesis, University of Canterbury, Christchurch, New Zealand (1997).
7. G. Cardona, F. Rossello and G. Valiente, Tripartitions do not always discriminate phylogenetic networks, arXiv:0707.2376v1 [q-bio.PE] (2007).
8. I. Cassens, P.Mardulyn and M.C.Milinkovitch, Evaluating intraspecific “network” construction methods using simulated sequence data: Do existing algorithms outperform the global maximum parsimony approach?, *Systematic Biology* 54, 363–372 (2005).
9. W.F. Doolittle, Phylogenetic classification and the universal tree, *Nature* 284, 2124–2128 (1999).
10. L. Gasieniec, J. Jansson, A. Lingas and A. Östlin, On the complexity of constructing evolutionary trees, *Journal of Combinatorial Optimization* 3, 183–197 (1999).
11. J. Håstad, Some optimal approximability results, in *Proceedings of the 28th ACM Symposium on Theory of Computing, 1997*, 1–10 (1997).
12. L. van Iersel, J. Keijsper, S. Kelk and L. Stougie, Constructing level-2 phylogenetic networks from triplets, arXiv:0707.2890v1 [q-bio.PE] (2007).
13. J. Jansson, On the complexity of inferring rooted evolutionary trees, in *Proceedings of the Brazilian Symposium on Graphs, Algorithms, and Combinatorics* (GRACO 2001), Electron. Notes Discrete Math. 7, Elsevier, 121–125 (2001).
14. J. Jansson, N. Nguyen and W. Sung, Algorithms for combining rooted triplets into a galled phylogenetic network, *SIAM Journal on Computing* 35(5), 1098–1121 (2006).
15. J. Jansson and W. Sung, Inferring a level-1 phylogenetic network from a dense set of rooted triplets, *Theoretical Computer Science* 363, 60–68 (2006).
16. V. Kunin, L. Goldovsky, N. Darzentas and C. A. Ouzounis, The net of life: Reconstructing the microbial phylogenetic network, *Genome Research* 15, 954–959 (2005).
17. C. R. Linder and L. H. Rieseberg, Reconstructing patterns of reticulate evolution in plants, *American Journal of Botany* 91, 1700–1708 (2004).
18. W. Martin. Mosaic bacterial chromosomes: a challenge on route to a tree of genomes, *BioEssays* 21, 99–104 (1999).

19. R. Motwani and P. Raghavan, *Randomized algorithms*, Cambridge University Press, New York, NY (1995).
20. L. Nakhleh, J. Sun, T. Warnow, C.R. Linder, B.M.E. Moret and A. Tholse, Towards the development of computational tools for evaluating phylogenetic network reconstruction methods, *Pacific Symposium on Biocomputing* 8, 315–326 (2003).
21. A. Newman and S. Vempala, Fences Are Futile: On Relaxations for the Linear Ordering Problem, in *Proceedings of the 8th International IPCO Conference on Integer Programming and Combinatorial Optimization*, LNCS 2081, 333–347 (2001).
22. C. H. Papadimitriou and M. Yannakakis, Optimization, approximation, and complexity classes, *Journal of Computer and System Sciences* 43, 425–440 (1991).
23. M. C. Rivera and J. A. Lake, The ring of life provides evidence for a genome fusion origin of eukaryotes, *Nature* 43, 152–155 (2004).
24. B. Wu, Constructing the maximum consensus tree from rooted triples, *Journal of Combinatorial Optimization* 8, 29–39 (2004).
25. <http://homepages.cwi.nl/~kelk/tripletverify/>

A Appendix

In this section we give the full proofs for Lemma 3 from Section 3, Theorem 3 from Section 5 and Observation 1 from Section 6.

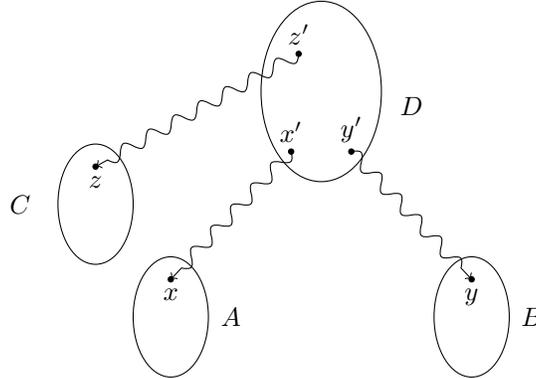


Figure 4. The case $LCA(A, B) = LCA(A, B, C)$ from the proof of Lemma 3.

Lemma 3. *Given a level- k network \mathcal{N} , we can preprocess it in time $O(m + mk^2)$ so that given any three vertices x, y, z we can check whether $xy|z$ is consistent with \mathcal{N} in time $O(1)$.*

Proof. We begin with a definition and two claims. A *chain* is a sequence of vertices $v_0 \rightarrow v_1 \rightarrow \dots \rightarrow v_{k+1}$ such that both in and out degrees of v_1, \dots, v_k are all equal 1. Now, consider an arbitrary biconnected component in \mathcal{N} . We claim that, if each chain within the component is replaced by an edge, the transformed component contains at most $\max(2, 3k)$ vertices. This is trivially true for the biconnected component consisting of only a single edge. To see that it is more generally true, note that the transformed component contains at most k recombination vertices and that all other vertices have outdegree 2 and indegree at most 1. Each such non-recombination vertex creates at least one new path that eventually has to terminate in a recombination vertex, and a recombination vertex can terminate at most two paths. So there are at most $2k$ non-recombination vertices, and the claim follows.

From now on biconnected component (or simply component) refers to a biconnected component containing more than one edge. We claim that all biconnected components within \mathcal{N} are vertex-disjoint. To see why this is, note that each vertex in such a component must have degree at least 2. The value of indegree plus outdegree of all vertices in \mathcal{N} is at most 3, so they cannot be a part of more than one such component.

The above reasoning shows that we could imagine the network as a rooted tree \mathcal{T} in which each vertex corresponds to some bigger component that has relatively simple structure: contracting all chains in it gives us a graph of size $O(k)$.

First we focus on the case when x, y, z lie in one biconnected component. The obvious solution is to simply preprocess such queries for each triple of vertices from one biconnected component. This does not give us the desired complexity yet: while each component is small after contracting chains, it might originally contain as many as $\Omega(m)$ vertices. We may overcome this difficulty by observing that if some chain consist of more than 3 inner vertices, we can replace it by a chain containing exactly 3 of them. Then we preprocess the resulting graph using the $O(|V|^3)$ algorithm from Lemma 2 (observe that $|V| = O(k)$). Given a query concerning consistency of some $xy|z$ we may have to replace some of x, y, z with other vertices lying on the shortened chains, which can be

easily done in time $O(1)$. The whole preprocessing takes time $\sum_i O(k_i^3)$ where all $k_i = O(k)$ and $\sum_i k_i = O(m)$, giving us the desired complexity.

Now we can solve the general case. Let A, B, C be biconnected components such that $x \in A$, $y \in B$ and $z \in C$. We can preprocess \mathcal{T} in linear time so that given its two vertices we can find their lowest common ancestor (LCA) in time $O(1)$ [3]. (This is already sufficient for the case $k = 0$ and contributes the m term in the running time.) If $xy|z$ is consistent there are only two possible situations:

1. $LCA(A, B, C)$ is a proper ancestor of $LCA(A, B)$. This is easy to detect.
2. $LCA(A, B) = LCA(A, B, C)$. Let $D = LCA(A, B)$. We can find x', y', z' - the entrance points within D (see Figure 4) - in time $O(1)$. It can be done by either using level ancestor queries or modifying the LCA algorithm [4]. We can then check whether $x'y'|z'$ is consistent using the above preprocessing.

□

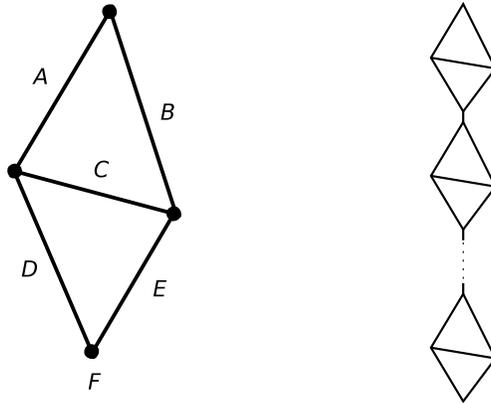


Figure 5. We construct the network $LB_2(n)$ by repeatedly chaining the structure on the left, a *simple level-2 network* (see [12]), together to obtain an overall topology resembling the structure on the right.

Theorem 3. *Let T be a set of input triplets labelled by n species. It is possible to find in polynomial time a level-2 network $\mathcal{N}(T)$ such that $\mathcal{N}(T)$ is consistent with at least a fraction 0.61 of T .*

Proof. We prove the theorem by showing how to construct a topology, which we call $LB_2(n)$, consistent with at least 0.61 of the triplets in $T_1(n)$. Using Theorem 1 $LB_2(n)$ can then be labelled to obtain the network $\mathcal{N}(T)$. We show by induction how $LB_2(n)$ can be constructed. We take $n < 16813$ as the induction base; for these values of n we refer to a simple computational proof written in Java [25]. We now prove the result for $n \geq 16813$. Let us assume by induction that, for any $n' < n$, there exists some topology $LB_2(n')$ such that $\#LB_2(n') \geq 0.61$. If we let $t(n')$ equal the number of triplets in $T_1(n')$ consistent with $LB_2(n')$, we have that $t(n')/3 \binom{n'}{3} \geq 0.61$ and thus that $t(n') \geq 1.83 \binom{n'}{3}$. Consider the structure in Figure 5. For $S \in \{A, B, C, D, E\}$, we define the operation *hanging l leaves from side S* as replacing the edge S with a directed path containing l internal vertices, and then attaching a leaf to each internal vertex. We construct $LB_2(n)$ as follows. We create a copy of the structure from the figure and hang $c = \lfloor 0.385n \rfloor$ leaves from side C , $d = \lfloor 0.07n \rfloor$ from side D and $e = \lfloor 0.26n \rfloor$ from side E . We let $f = \lfloor 0.285n \rfloor$ and add the edge (F, r) , where r is the root of the network $LB_2(f)$. Finally we hang $a = n - (c + d + e + f)$ leaves

from side A ; it might be that $a = 0$. (The only reason we hang leaves from side A is to compensate for the possibility that $c + d + e + f$ does not exactly equal n .) This completes the construction of $LB_2(n)$; note that as in Section 4 the network is constructed by recursively chaining the same basic structure together.

We can use Mathematica to show that $LB_2(n)$ is consistent with at least 0.61 of the triplets in $T_1(n)$. In particular, by explicitly counting the triplets consistent with $LB_2(n)$ we derive an inequality expressed in terms of $n, c, d, e, f, t(f)$, which Mathematica then simplifies to a cubic inequality in n that holds for all $n \geq 16813$. (To simplify the inequality we take $x - 1$ as a lower bound on $\lfloor x \rfloor$ and assume that no leaves are hung from side A). The Mathematica script is reproduced in Figure 6, and can be downloaded from [25]. Finally, we comment that the networks computationally constructed for $n < 16813$ are, essentially, built in the same way as the networks described above. The only difference is that, to absorb inaccuracies arising from the floor function, we try several possibilities for how many leaves should be hung from each side; for side C , for example, we try also $(c - 1)$ and $(c + 1)$ leaves. \square

```

c[n] = ((385 / 1000) * n) - 1
d[n] = ((70 / 1000) * n) - 1
e[n] = ((260 / 1000) * n) - 1
f[n] = ((285 / 1000) * n) - 1
t[n] = (3 * (610 / 1000) * Binomial[f[n], 3])

Simplify[
  (Binomial[c[n], 3] + Binomial[d[n], 3] + Binomial[e[n], 3] + t[n] + (Binomial[c[n], 2] * d[n]) +
    (Binomial[c[n], 2] * e[n]) + (Binomial[c[n], 2] * f[n]) + (c[n] * d[n] * e[n]) +
    (c[n] * d[n] * f[n]) + (Binomial[c[n], 2] * e[n]) + (c[n] * e[n] * d[n]) +
    (c[n] * e[n] * f[n]) + (Binomial[c[n], 2] * f[n]) + (c[n] * f[n] * d[n]) +
    (c[n] * f[n] * e[n]) + (Binomial[d[n], 2] * c[n]) + (Binomial[d[n], 2] * e[n]) +
    (Binomial[d[n], 2] * f[n]) + (d[n] * f[n] * c[n]) + (Binomial[d[n], 2] * f[n]) +
    (d[n] * f[n] * e[n]) + (Binomial[e[n], 2] * c[n]) + (Binomial[e[n], 2] * d[n]) +
    (Binomial[e[n], 2] * f[n]) + (e[n] * f[n] * c[n]) + (e[n] * f[n] * d[n]) +
    (Binomial[e[n], 2] * f[n]) + (Binomial[f[n], 2] * c[n]) + (Binomial[f[n], 2] * d[n]) +
    (Binomial[f[n], 2] * e[n])) / (3 * Binomial[n, 3]) > (610 / 1000) ]

-147 984 000 000 + 94 041 640 000 n - 16 470 260 400 n2 + 979 319 n3
----- > 0
n (2 - 3 n + n2)

Reduce[  $\frac{-147 984 000 000 + 94 041 640 000 n - 16 470 260 400 n^2 + 979 319 n^3}{n (2 - 3 n + n^2)} > 0, n, \text{Integers}$  ]

n ∈ Integers && (n ≤ -1 || n ≥ 16 813)

```

Figure 6. The Mathematica script used in the proof of Theorem 3.

Observation 1. *MAX-LEVEL-0 and MAX-LEVEL-0-LABELLING are both APX-complete.*

Proof. By Theorem 1 we may label any tree topology to make it consistent with $1/3$ of the given triplets. Therefore, both problems are in the class APX. To prove APX-hardness we use a reduction proposed by Wu [24] and we show that it is actually an *L-reduction* from the MAXIMUM SUBDAG problem. Both L-reductions and the MAXIMUM SUBDAG problem were studied by Papadimitiou and Yannakakis [22]. MAXIMUM SUBDAG, which is equivalent to the problem LINEAR ORDERING, has been proven APX-complete [21][22].

In the MAXIMUM SUBDAG problem we are given a directed graph $G = (V, A)$, and the goal is to find a maximal cardinality subset of arcs $A' \subset A$ such that $G' = (V, A')$ is acyclic.

In the reduction of Wu one constructs an instance of the MAX-LEVEL-0 problem as follows. Given a directed graph $G = (V, A)$, let $x \notin V$, consider the set of triplets T containing a single triplet $t_{uv} = ux|v$ for every arc $(u, v) \in A$, where $X = X(T) = V \cup \{x\}$. To argue that it is an L-reduction it remains to prove the following two claims.

1) If there exists a subset of arcs $A' \subset A$ such that $G' = (V, A')$ is acyclic and $|A'| = k$, then there exists a phylogenetic tree consistent with at least k triplets from T . To prove this claim, we consider a topological sorting of vertices in graph G' . We construct the phylogenetic tree to be a caterpillar with the leaves labeled (bottom-up) by such sorted vertices, the lowest leaf is labelled by x . It remains to observe that for any arc $(u, v) \in A'$ the corresponding triplet t_{uv} is consistent with the obtained phylogenetic tree.

2) Given a phylogenetic tree \mathcal{B} consistent with l triplets from T , we may construct in polynomial time a subset of arcs $A' \subset A$ such that $G' = (V, A')$ is acyclic and $|A'| = l$. In fact we will show that it suffices to take A' consisting of the arcs (u, v) such that the corresponding triplet t_{uv} is consistent with \mathcal{B} . We only need to argue that for such a choice of A' the resulting graph $G' = (V, A')$ is acyclic. Consider the path in the tree \mathcal{B} from the root node to the leaf labeled by the special species x . For any vertex $v \in A$, the species v has an internal node on this path where he branched out of the evolution of x , namely the lowest common ancestor of u and x ($LCA(u, x)$). Observe, that the position of $LCA(u, x)$ induces a partial ordering $>_{\mathcal{B}}$ on A . Recall, that if a triplet $t_{uv} = ux|v$ is consistent with \mathcal{B} , then $LCA(u, x)$ is a proper ancestor of $LCA(v, x)$. Therefore, the consistent triplets from T induce another partial ordering that may be extended to $>_{\mathcal{B}}$. This implies that for A' containing the arcs (u, v) such that a triplet t_{uv} is consistent with \mathcal{B} the graph $G' = (V, A')$ is acyclic.

With the above construction we have shown that the existence of an ϵ -approximation algorithm for the MAX-LEVEL-0 problem implies existence of an ϵ -approximation algorithm for the MAXIMUM SUBDAG problem. In particular, existence of a Polynomial-Time Approximation Scheme (PTAS) for MAX-LEVEL-0 would imply existence of PTAS for MAXIMUM SUBDAG, which is unlikely due to the results in [21] and [22].

In the reduction we might have assumed a particular topology for the tree. Namely, we might have assumed, that the topology needs to be a caterpillar. Therefore, the problem MAX-LEVEL-0-LABELLING is also APX-hard. \square