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

Jaroslaw Byrka[1,2], Katharina T. Huber[3], Steven Kelk[1],

[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

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 [12]. 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.

* 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.

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 [3]. 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* [7][14][16][19]. Such structures are more general than trees and allow us to capture the phenomenon of reticulate evolution i.e. non tree-like evolution. 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 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 [10][12], although the problem becomes polynomial-time solveable if the input triplets are *dense* i.e. if there is at least one triplet in the input for each subset of three species [10][13].

Several authors have considered algorithmic strategies of use when the algorithms from [1] and [13] fail to find a tree or network. Gąsieniec et al [9] 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, [4][11][20] 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, [12] 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 [9] 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 and leads immediately to a simple extension of the 1/3 result from [9]. 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 [12], and which is thus worst-case optimal for level-1 networks. We also demonstrate how the derandomization can be optimized if we have more information about the structure of $\mathcal{N}$. Specifically, we show an alternative worst-case optimal level-1 algorithm with an optimized running time of $O(|T|n^2)$ where $|T|$ is the number of triplets in the input (Theorem 2). 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 [10]) 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.

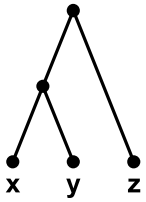

Figure 1. One of the three possible triplets on the set of species $\{x, y, z\}$. Note that, as with all figures in this article, all arcs are assumed to be directed downwards, away from the root.

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 $u \to x$, $u \to y$, $v \to u$ and $v \to z$*[4].

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 $\mathcal{N}$.

Lemma 1. *Given a phylogenetic network $\mathcal{N}$ with $n_+ = |V(\mathcal{N})|$ vertices, and a triplet t, it is possible to determine in time $O(n_+^4)$ whether t is consistent with $\mathcal{N}$.*

Proof. See Appendix. □

3 Labelling a network topology

Suppose we are given a topology N with n leaves, and a set T of m triplets where $L^N = \{l_1, l_2, \ldots, l_n\}$ and $X = X(T) = \{x_1, x_2, \ldots, x_n\}$. 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 $\{x_1, x_2, \ldots, x_n\}$. 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.

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 $T_1(n) = \{t_1, t_2, \ldots, t_{3\binom{n}{3}}\}$ 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 $t_1, t_2, \ldots, t_{3\binom{n}{3}}$ in (N, γ_0). Since a $\#N$ fraction of $t_1, t_2, \ldots, t_{3\binom{n}{3}}$ is consistent with (N, γ_0), a $\#N$ fraction of these positions

[4] 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.

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 $\frac{2 \cdot (n-3)!}{n!} = \frac{1}{3 \cdot \binom{n}{3}}$.

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). □

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 using the method of conditional expectations. Specifically, we will iteratively use a procedure that computes the expected fraction of consistent triplets assuming that labels of certain leaves are already fixed, and that the labelling is chosen uniformly at random for the remaining leaves. In order to compute this expectation, we will calculate, for each triplet $t \in T$, the probability that such a random labelling γ will make t consistent with (N, γ). To calculate this probability we will consider all the possible positions of triplet t in the topology N and check if this position is still *available* for t, given the already chosen labels. Among these available positions we will count those that make a triplet consistent. Observe, that each of the positions available for t has the same probability of being chosen for t, hence the probability of t being consistent with (N, γ) is just the fraction of consistent positions among those available for t. To speed up the algorithm, part of the computation will be done in a preprocessing phase. In this preprocessing we will compute, for every position p in N, whether p is a position that makes a triplet consistent. For each position we use Lemma 1 to test if this position is a consistent one. The overall algorithm is thus as follows:

1. Preprocessing: compute which positions in N make a triplet consistent.
2. $\Gamma \leftarrow$ set of all possible labellings.
3. 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 labelled by x
     - compute $E(f(N, \gamma_x, T))$, where γ_x is chosen randomly from Γ_x, by the above described procedure
   - $\Gamma \leftarrow \Gamma_x$ s.t. $x = argmax_x E(f(N, \gamma_x, T))$
4. return $\gamma_0 \leftarrow$ the only element of Γ.

The resulting γ_0 fixes labels for all the leaves, hence it is no longer a random labelling, but just a function. It remains only to analyse the quality of γ_0 and the running time of the derandomization.

Theorem 1. *For any topology N and any triplet set T on n leaves, a labelling γ_0 such that $f(N, \gamma_0, T) \geq \#N$ can be found in time $O(n_+^4 \cdot n^3)$, where $n^+ = |V(N)|$.*

Proof. We will argue that, for the γ_0 produced by the above algorithm, $f(N, \gamma_0, T) \geq \#N$. 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 $E(f(N,\gamma_x,T)) \geq E(f(N,\gamma,T))$.

We estimate the running time of this derandomization as follows. By Lemma 1, the preprocessing phase takes $O(n_+^4 \cdot n^3)$ time. In the main calculation part we need to fix every leaf, and we try every available species on it, which gives $O(n^2)$ tries. Each time we need to calculate the expected fraction of consistent triplets. The easy way to do so is to try every single triplet on every single position, which takes $O(|T| \cdot n^3)$ time. We may save time by grouping those triplets that do not have any leaf fixed yet: they have the same probability of being consistent with (N,γ). This group of triplets may be served in $O(n^3)$ time, and for each of the other triplets it suffices to enumerate the possible fixings of at most 2 leaves, which takes in total $O(n^2 \cdot |T|)$ time. Concluding, the derandomization time is dominated by the preprocessing phase and takes at most $O(n_+^4 \cdot n^3)$ time. □

3.1 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. [9] 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] [5] [6] [17] [15] 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 [12]. 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 [12] it was shown how to construct a special level-1 topology $C(n)$, which we call a *galled caterpillar*[5], 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 $n > 2$,

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

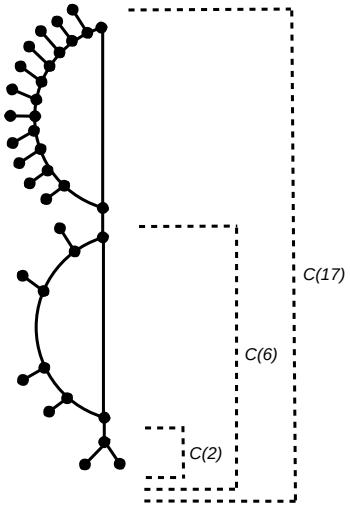

Figure 2. This is galled caterpillar $C(17)$. It contains two galls and ends with a tail of two leaves. $C(17)$ contains 11 leaves in the top gall because Equation 1 is maximised for $a = 11$.

In Figure 2 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 kth 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 2. *Let T be a set of input triplets labelled by n species. Then in time $O(n^7)$ 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. □

[5] In [12] 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 2 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 asymptope 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 2 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 [12].

Lemma 3. *$S(n)/3\binom{n}{3} > 0.48$ 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 [21]. 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 $n \geq 116$. □

4.2 Optimising the derandomization when the underlying structure is already known

The derandomization presented in Theorem 1 essentially works by exhaustive search. If we know more about the structure of the topology that we are labelling, the derandomization can be made much faster. We continue with level-1 networks to demonstrate this.

Lemma 4. *The running time named in Lemma 2 can be improved to $O(|T|n^2)$ by customizing the derandomization procedure.*

Proof. We begin by constructing in time $O(n^2)$ the topology $C(n)$ (see Lemma 2). Let r be the number of recombination vertices in $C(n)$. We partition the leaves of $C(n)$ into $(r+1)$ *blocks*. Let d_0 be the number of leaves in block 0 i.e. between the root and the first recombination vertex. Let d_r be the number of leaves in block r i.e. beneath the last recombination vertex (this value will by definition be either 1 or 2). For $0 < i < r$ let d_i be the number of leaves in block i i.e. beneath the ith recombination vertex (numbering downwards from the root) but above the $(i+1)$th recombination vertex. Let d_i^+ refer to $\sum_{j>i} d_i$. We impose an ordering $<_c$ on the leaves, defined as follows. For two

leaves l_1, l_2 in the same block, l_1 is earlier in the ordering (i.e. $l_1 <_c l_2$) iff l_1 is closer to the root of $C(n)$ then l_2 (in terms of shortest directed path.) For two leaves in different blocks, for example blocks b and b', the leaf in block $\min(b, b')$ is earlier in the ordering. Note that if block r contains 2 leaves then these leaves are actually indistinguishable under $<_c$ (see Figure 2); to ensure that $<_c$ is a total ordering we in this case impose an arbitrary ordering on those two leaves. This does not cause problems with the rest of the analysis.

We will consider the leaves of $C(n)$ in the order specified by $<_c$. Assume that the labelling of the first $(k-1)$ leaves has been determined. To calculate the labelling of the kth leaf we choose a label which maximises the expected number of triplets satisfied, assuming the labelling of the remaining $n-k$ leaves is chosen uniformly at random. For each possible labelling of the kth leaf, and for each triplet t in the input, we thus need to calculate the probability $P(t)$ that t is consistent with the network if the remaining $n-k$ leaves are labelled uniformly at random. The derandomization described in Section 3 essentially computes this probability by exhaustively trying all ways of mapping the unassigned species from t into the unlabelled leaves. We can do this much faster by observing that in this case there are essentially only 6 different classes of triplets. Let I be the set of species used to label the first k leaves. Let $O = X \setminus I$ be the not yet assigned species. All triplets are thus of the form $OO|I, II|O, IO|I, OI|O, OO|O$ or $II|I$. We assume that the kth leaf is in block i. Let l refer to the number of unlabelled leaves in block i. Let γ be the partially complete labelling (at any iteration only defined on the range I). We now consider each class of triplets in turn.

Case $OO|I$: $P(t)$ will always be 1, irrespective of how the remaining $n-k$ leaves are labelled.

Case $II|O$: Let $t = xy|z$. If x and y are not mapped to the same block, $P(t)$ is zero. Otherwise, t is consistent with the network iff z is not mapped to the l unassigned leaves remaining in that block. So in this case $P(t) = 1 - l/(n-k)$.

Case $IO|I$: Let $t = xy|z$ where x and z are the I species. $P(t)$ is 0 if $\gamma^{-1}(x) <_c \gamma^{-1}(z)$, and 1 otherwise.

Case $OI|O$: Let $t = xy|z$ where y is the I species. Consider a labelling γ_1 obtained by extending γ uniformly at random, as described. If any of the following three conditions are true the triplet t is not consistent with the network: (1) If x and y are not mapped to the same block; (2) if $\gamma_1^{-1}(z) <_c \gamma_1^{-1}(x)$; (3) if z and x are mapped to the same block with $\gamma_1^{-1}(x) <_c \gamma_1^{-1}(z)$. Thus, $P(t) = l d_i^+ / (n-k)(n-k-1)$.

Case $OO|O$: Let $t = xy|z$. Again, assume that labelling γ_1 has been obtained by extending γ uniformly at random. Let $s = (n-k-3)!$. There are in total $2s\binom{n-k}{3}$ choices for γ_1 of the form $\gamma_1^{-1}(z) <_c \gamma_1^{-1}(y) <_c \gamma_1^{-1}(x)$ or $\gamma_1^{-1}(z) <_c \gamma_1^{-1}(x) <_c \gamma_1^{-1}(y)$ which make t consistent with the network. The only other types of labelling γ_1 which make t consistent with the network are of the form $\gamma_1^{-1}(y) <_c \gamma_1^{-1}(x) <_c \gamma_1^{-1}(z)$ or $\gamma_1^{-1}(x) <_c \gamma_1^{-1}(y) <_c \gamma_1^{-1}(z)$. In all these cases x and y have to be mapped to the same block, and z has to be mapped to a lower block. The case when x and y are both mapped to block i contributes $2s\binom{l}{2} d_i^+$. The case when x and y are mapped together to some block below block i contributes $2s \sum_{j>i} \binom{d_j}{2} d_j^+$. Given that there are in total $(n-k)! = (n-k)(n-k-1)(n-k-2)s$ choices for γ_1, we have that:

$$P(t) = \frac{2\binom{n-k}{3} + 2\binom{l}{2} d_i^+ + 2\sum_{j>i} \binom{d_j}{2} d_j^+}{(n-k)(n-k-1)(n-k-2)}$$

Case $II|I$: Let $t = xy|z$. $P(t)$ will in this case be 1 or 0 because species x, y, z have already been assigned to leaves. The remainder of the analysis is essentially the same as in case $OO|O$: $P(t) = 1$ iff (1) $\gamma^{-1}(z) <_c \gamma^{-1}(y) <_c \gamma^{-1}(x)$ or (2) $\gamma^{-1}(z) <_c \gamma^{-1}(x) <_c \gamma^{-1}(y)$ or (3) $\gamma^{-1}(y) <_c \gamma^{-1}(x) <_c \gamma^{-1}(z)$ or $\gamma^{-1}(x) <_c \gamma^{-1}(y) <_c \gamma^{-1}(z)$ with x and y mapped to the same block and z to a lower block.

The running time of the derandomization can be analysed as follows. We begin with a preprocessing phase. Firstly we compute a look-up table for the values $S(n')$ for $0 \leq n' \leq n$. As in Lemma 2 this takes time $O(n^2)$. From this data we construct in time $O(n)$ a look-up table for the values d_i, d_i^+ and the function (used in case $OO|O$) $f(i) = 2\sum_{j>i} \binom{d_j}{2} d_j^+$. Total preprocessing time is thus $O(n^2)$. In the algorithm itself we maintain and update a record, for each species x, of whether it is in I or O and (where relevant) the location of $\gamma^{-1}(x)$ in the $<_c$ ordering, and which block it has been mapped to. This is all computable in time $O(1)$. The algorithm needs n iterations to fix n leaves, and within each iteration $\leq n$ species candidates for the kth leaf have to be tried. For each species we try in the kth leaf we need to compute the conditional expectation by iterating over all the $|T|$ triplets; this takes $O(|T|)$ time because the probabilities for the six triplet cases are all computable in $O(1)$ time (thanks to the pre-computation of the $f(.)$ function.) The total running time is thus $O(|T|n^2)$. □

By combining all three lemmas from this section we obtain the following theorem:

Theorem 2. *Let T be a set of input triplets labelled by n species. In time $O(|T|n^2)$ 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. □

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 [20]) 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. See Appendix. □

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 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 3.1, 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 [9]. 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 thus an interesting open problem to determine whether better approximation factors can be obtained for the latter problem via some different approach. For example, an empirical study in [20] suggests that, for MAX-LEVEL-0, approximation factors in the region of 0.833 might be possible. 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, analogous perhaps to using $|T|$ as an upper bound.

8 Acknowledgements

We thank Leo van Iersel for his assistance with the $O(n_+^4)$ triplet consistency-checking algorithm 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), pp. 405–421 (1981).
2. M. Baroni, C. Semple, and M. Steel, A framework for representing reticulate evolution, *Annals of Combinatorics* 8(4), pp. 391–408 (2004).
3. O. Bininda-Emonds, Phylogenetic supertrees: combining information to reveal the tree of life, Computational Biology Series 4, Kluwer (2004).
4. 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).
5. G. Cardona, F. Rossello and G. Valiente, Tripartitions do not always discriminate phylogenetic networks, arXiv:0707.2376v1 [q-bio.PE] (2007).
6. 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, pp. 363–372 (2005).
7. W.F. Doolittle, Phylogenetic classification and the universal tree, *Nature* 284, pp. 2124–2128 (1999).
8. S. Fortune, J. Hopcroft and J. Wyllie, The directed subgraph homeomorphism problem, *Theoretical Computer Science* 10, pp. 111–121 (1980).
9. L. Gąsieniec, J. Jansson, A. Lingas and A. Östlin, On the complexity of constructing evolutionary trees, *Journal of Combinatorial Optimization* 3, pp. 183–197 (1999).
10. L. van Iersel, J. Keijsper, S. Kelk and L. Stougie, Constructing level-2 phylogenetic networks from triplets, arXiv:0707.2890v1 [q-bio.PE] (2007).
11. 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, pp. 121–125 (2001).
12. J. Jansson, N. Nguyen and W. Sung, Algorithms for combining rooted triplets into a galled phylogenetic network, *SIAM Journal on Computing* 35(5), pp. 1098–1121 (2006).
13. J. Jansson and W. Sung, Inferring a level-1 phylogenetic network from a dense set of rooted triplets, *Theoretical Computer Science* 363, pp. 60–68 (2006).
14. V. Kunin, L. Goldovsky, N. Darzentas and C. A. Ouzounis, The net of life: Reconstructing the microbial phylogenetic network, *Genome Research* 15, pp. 954–959 (2005).
15. C. R. Linder and L. H. Rieseberg, Reconstructing patterns of reticulate evolution in plants, *American Journal of Botany* 91, pp. 1700–1708 (2004).
16. W. Martin. Mosaic bacterial chromosomes: a challenge on route to a tree of genomes, *BioEssays* 21, pp. 99 (1999).
17. 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, pp. 315–326 (2003).
18. C. H. Papadimitriou and M. Yannakakis, Optimization, approximation, and complexity classes, *Journal of Computer and System Sciences* 43, pp. 425–440 (1991).
19. M. C. Rivera and J. A. Lake, The ring of life provides evidence for a genome fusion origin of eukaryotes, *Nature* 43, pp. 152–155 (2004).
20. B. Wu, Constructing the maximum consensus tree from rooted triples, *Journal of Combinatorial Optimization* 8, pp. 29–39 (2004).
21. http://homepages.cwi.nl/∼kelk/tripletverify/

A Appendix

In this section we give the full proofs for Lemma 1 from Section 2, Theorem 3 from Section 5 and Observation 1 from Section 6.

Lemma 1. *Given a phylogenetic network $\mathcal{N}$ with $n_+ = |V(\mathcal{N})|$ vertices, and a triplet t, it is possible to determine in time $O(n_+^4)$ whether t is consistent with $\mathcal{N}$.*

Proof. We let $t = xy|z$ and for simplicity identify x, y, z with the leaves of $\mathcal{N}$ where the respective species are found. Let r be the root of $\mathcal{N}$. We claim that t is consistent with $\mathcal{N}$ iff there exists a vertex $u \notin \{x, y, z, r\}$ in N such that the following three paths exist in $\mathcal{N}$ and are mutually disjoint in terms of their internal vertices: $r \to z, u \to x, u \to y$.

($\Rightarrow$) Clearly, if t is consistent with $\mathcal{N}$ then by Definition 2 the path $v \to z$ exists. If $v = r$ we are done, otherwise the path $r \to z$ can be created by combining $v \to z$ with any path $r \to v$.

($\Leftarrow$) Suppose that the three paths exist. Consider any path P from r to u. Given that $\mathcal{N}$ is a directed acyclic graph such a path cannot intersect with the internal vertices of the paths $u \to x$ and $u \to y$. If P does not intersect with the internal vertices of the path $r \to z$ we are done. Otherwise let v be the last vertex from P that intersects with a vertex from $r \to z$, and let $v \to u$ be the subpath of P starting from v. It follows that the paths $u \to x, u \to y, v \to u, v \to z$ exist and are mutually disjoint on their internal vertices.

If we already knew u, we could use the algorithm in [8] (Thm. 3) to determine whether there exist three (internally vertex) disjoint paths $r \to z, u \to x, u \to y$ in $\mathcal{N}$. To be specific, the algorithm from [8] shows how (in this case) the problem can be reduced to searching for a path between two vertices in a state-space graph G' containing $O(n_+^3)$ vertices. The graph G' is acylic, and the fact that every vertex in $\mathcal{N}$ has outdegree ≤ 2 implies that G' has a linear number of edges. So path-finding can also be done in $O(n_+^3)$ time, giving an overall running time of $O(n_+^3)$ if we know u. However, we have to guess u by trying all n_+ possibilities for it, giving a running time of $O(n_+^4)$ for the entire algorithm. □

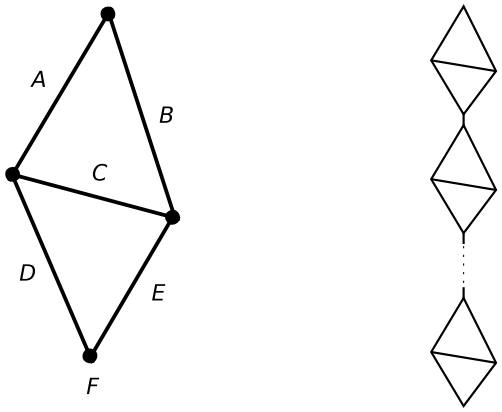

Figure 3. We construct the network $LB_2(n)$ by repeatedly chaining the structure on the left, a *simple level-2 network* (see [10]), together to obtain an overall topology resembling the structure on the right.

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 [21]. 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 3. 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 4, and can be downloaded from [21]. 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. □

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 [20] 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 [18]. They proved that the MAXIMUM SUBDAG problem is APX-complete.

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 (top down) 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.

```
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) ]
```

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

```
Reduce[ (-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, Integers ]
```

$$n \in \text{Integers} \;\&\&\; (n \le -1 \;||\; n \ge 16\,813)$$

Figure 4. The Mathematica script used in the proof of Theorem 3.

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 result of Papadimitiou and Yannakakis [18].

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. □