On Computing the Maximum Parsimony Score of a Phylogenetic Network

Mareike Fischer, Leo van Iersel, Steven Kelk, Celine Scornavacca

SIAM J. DISCRETE MATH.
Vol. 29, No. 1, pp. 559–585

ON COMPUTING THE MAXIMUM PARSIMONY SCORE OF A PHYLOGENETIC NETWORK*

MAREIKE FISCHER†, LEO VAN IERSEL‡, STEVEN KELK§, AND CELINE SCORNAVACCA¶

Abstract. Phylogenetic networks are used to display the relationship among different species whose evolution is not treelike, which is the case, for instance, in the presence of hybridization events or horizontal gene transfers. Tree inference methods such as maximum parsimony need to be modified in order to be applicable to networks. In this paper, we discuss two different definitions of maximum parsimony on networks, "hardwired" and "softwired," and examine the complexity of computing them given a network topology and a character. By exploiting a link with the problem MULTITERMINAL CUT, we show that computing the hardwired parsimony score for 2-state characters is polynomial-time solvable, while for characters with more states this problem becomes NP-hard but is still approximable and fixed parameter tractable in the parsimony score. On the other hand we show that, for the softwired definition, obtaining even weak approximation guarantees is already difficult for binary characters and restricted network topologies, and fixed-parameter tractable algorithms in the parsimony score are unlikely. On the positive side we show that computing the softwired parsimony score is fixed-parameter tractable in the level of the network, a natural parameter describing how tangled reticulate activity is in the network. Finally, we show that both the hardwired and the softwired parsimony scores can be computed efficiently using integer linear programming. The software has been made freely available.

Key words. phylogenetic trees, phylogenetic networks, parsimony, complexity, approximability, fixed-parameter tractability, software

AMS subject classifications. 68W25, 05C20, 90C27, 92B10

DOI. 10.1137/140959948

1. Introduction.

In phylogenetics, graphs are used to describe the relationships among different species. Traditionally, these graphs are trees, and biologists aim at reconstructing the so-called tree of life, i.e., the tree of all living species [24]. However, trees cannot display reticulation events such as hybridizations or horizontal gene transfers, which are known to play an important role in the evolution of certain species [2, 4, 23, 25]. In such cases considering phylogenetic networks rather than trees is potentially more adequate, where in its broadest sense a phylogenetic network can simply be thought of as a graph (directed or undirected) with its leaves labelled by species [11, 26, 27]. Phylogenetic networks can also be useful in the absence of reticulate events, since such networks can represent uncertainty in the true (tree-shaped) phylogeny. Hence, tree reconstruction methods, i.e., the methods used to

*Received by the editors March 7, 2014; accepted for publication (in revised form) January 14, 2015; published electronically March 24, 2015. This work appeared as contribution 2015-033 of the Institut des Sciences de l'Evolution de Montpellier.

http://www.siam.org/journals/sidma/29-1/95994.html

†Department of Mathematics and Computer Science, Ernst-Moritz-Arndt University Greifswald, 17487 Greifswald, Germany (email@mareikefischer.de).

‡Delft University of Technology, 2600 AA Delft, The Netherlands (l.j.j.v.iersel@gmail.com). This author's research was funded by a Veni grant of the Netherlands Organisation for Scientific Research (NWO).

§Department of Knowledge Engineering (DKE), Maastricht University, 6200 MD Maastricht, The Netherlands (steven.kelk@maastrichtuniversity.nl).

¶Institut des Sciences de l'Evolution (ISEM, UMR 5554 CNRS), Université Montpellier II, 34095 Montpellier Cedex 5, France (celine.scornavacca@univ-montp2.fr). This author's work was partially supported by ANCESTROME project ANR-10-IABI-0-01.

infer the best tree from, e.g., DNA or protein data, need to be adapted to networks.

One of the most famous tree reconstruction methods is maximum parsimony [6]. While this method has been shown to have drawbacks such as statistical inconsistency in the so-called Felsenstein zone [7], it is still widely used mainly due to its simplicity: maximum parsimony does not depend on a phylogenetic model and works in a purely combinatorial way. Moreover, for a given tree the optimal parsimony score can be found in polynomial time using the well-known Fitch algorithm [8] or the Sankoff algorithm [30]. This problem of finding the optimal parsimony score for a given tree is often referred to as the "small parsimony" problem. The "big parsimony" problem, on the other hand, aims at finding a most parsimonious tree amongst all possible trees—and this problem has proven to be NP-hard. Note that the latter problem is a close relative of the classical Steiner Tree problem [1, 10].

Recent studies have introduced extensions of the tree-based parsimony concept to phylogenetic networks [17, 19, 28], and a biological case study was presented in [16]. Basically, maximum parsimony on networks can be viewed in two ways: If one thinks of evolution as a tree-like process (but maybe with different trees for different parts of the genome, all of which are represented by a single network), one can define the parsimony score of a character on a network as the score of the best tree inside the network. The other way of looking at maximum parsimony on networks is just the same as the Fitch algorithm's view on trees: One can try to find the assignment of states to internal nodes of the network such that the total number of edges that connect nodes in different states is minimized. While the first concept may be regarded as more biologically motivated, the second one is in a mathematical sense the natural extension of the parsimony concept to networks. Both concepts of parsimony on networks are considered in this paper, and we formally introduce them in section 2 as softwired and hardwired parsimony, respectively.

Given an alignment (e.g., DNA) and a criterion like maximum parsimony, several questions come to mind: How hard is it to calculate the parsimony score (in both the hardwired and softwired senses) for a given network ("small parsimony" problem)? How hard is it to find a best network ("big parsimony" problem)? In this paper we consider only the "small parsimony" problem. Moreover, we focus on computing the parsimony score of a single character, since the parsimony score of an alignment can be computed by summing up the parsimony scores of the individual characters.

Kannan and Wheeler [19] introduced the hardwired parsimony score for networks and conjectured that it would be NP-hard to compute. We show in section 3 that this problem is indeed NP-hard and APX-hard (Corollary 3.3) whenever characters employing more than two states are used, but we also show that it is polynomial-time solvable for binary characters (Corollary 3.2). We also analyze the behavior of the algorithm in [19], which we call the ExtendedFitch algorithm, showing that it does not compute the hardwired parsimony score optimally and that it does not approximate the softwired parsimony score well.

In section 4, we consider the complexity of computing the softwired parsimony score. Previously, this problem was shown to be NP-hard and APX-hard but only for nonbinary networks with outdegree at most 20 [17]. We show in Theorem 4.2 that the softwired parsimony problem is NP-hard even for binary networks (and binary characters), and we additionally show that NP-hardness cannot be overcome by considering only so-called binary tree-child time-consistent networks (Theorem 4.3). Moreover, we show a much stronger inapproximability than the APX-hardness shown in [17]. We show that, for any constant $\epsilon > 0$, an approximation factor of $|X|^{1-\epsilon}$ is not

possible in polynomial time, unless P = NP, where $|X|$ denotes the number of species under investigation. Moreover, this holds even for tree-child time-consistent networks (Theorem 4.7). Our inapproximability result shows that a trivial approximation factor of $|X|$ is in a certain sense the best that is possible in polynomial time. For binary networks we show a slightly weaker inapproximability threshold: $|X|^{\frac{1}{3}-\epsilon}$ (Theorem 4.8). We note that the hardness results in [28] are not directly comparable to our results, since they adopt the *recombination network* model of phylogenetic networks (see, e.g., [11, 14] and also the discussion in [17]).

While we show that the hardwired parsimony score is fixed-parameter tractable with the parsimony score as a parameter (Corollary 3.4), we show that the softwired parsimony score is not, unless P = NP. Indeed, we show (in Corollary 4.4) that it is even NP-hard to determine whether the softwired parsimony score is equal to one (see [9, 29] for an introduction to fixed-parameter tractability). On the positive side, we show in section 5 (in Theorem 5.7) that the softwired parsimony score is fixed-parameter tractable in the *level* of the network, a parameter describing the maximum amount of reticulate activity in a biconnected component of the network (see, e.g., [21, 22] for an overview). Moreover, in section 6 we present an integer linear program to calculate both the softwired and the hardwired parsimony scores of a given character (or, more generally, multiple sequence alignment) on a phylogenetic network and give a preliminary analysis of its performance. This is the first practical exact method for computation of parsimony on medium to large networks, supplementing the heuristics given by [17, 19]. An implementation of this program is freely available [13].

Finally, in section 7, we summarize our results and state some open problems for future research.

2. Preliminaries. Let X be a finite set. An *unrooted phylogenetic network* on X is a connected, undirected graph that has no degree-2 nodes and that has its degree-1 nodes (the leaves) bijectively labelled by the elements of X. A *rooted phylogenetic network* on X is a directed acyclic graph that has a single indegree-0 node (the root), has no indegree-1 outdegree-1 nodes, and has its outdegree-0 nodes (the leaves) bijectively labelled by the elements of X. We identify each leaf with its label. The indegree of a node v of a rooted phylogenetic network is denoted by $\delta^-(v)$, and v is said to be a *reticulation node* (or a *reticulation*) if $\delta^-(v) \geq 2$. An edge (u, v) is called a *reticulation edge* if v is a reticulation node, and it is called a *tree edge* otherwise. A proper subset $C \subset X$ is referred to as a *cluster* of X. A *cherry* is a triple of nodes of which two are leaves and the third is the common parent of these leaves.

When we refer to a *phylogenetic network*, it can be either rooted or unrooted. We use $V(N)$ and $E(N)$ to denote, respectively, the node and edge set of a phylogenetic network N. To simplify notation, we use the notation (u, v) for a directed as well as for an undirected edge between u and v. A phylogenetic network is *binary* if each node has total degree at most 3 and (in the case of a rooted network) the root has outdegree 2 and all reticulations have outdegree 1.

The *reticulation number* of a phylogenetic network N can be defined as $|E(N)| - |V(N)| + 1$. Hence, the reticulation number of a rooted binary network is simply the number of reticulation nodes. A *rooted phylogenetic tree* is a rooted phylogenetic network with no reticulation nodes, i.e., with reticulation number 0. A *biconnected component* of a phylogenetic network is a maximal biconnected subgraph (i.e., a biconnected subgraph that is not contained in a larger biconnected subgraph). A phylogenetic network is said to be a *level-k* network if each biconnected component has reticulation number at most k and at least one biconnected component has reticula-

tion number exactly k.

A rooted phylogenetic network N is said to be *tree-child* if each nonleaf node has a child that is not a reticulation, and N is said to be *time-consistent* if there exists a "time-stamp" function $t : V(N) \to \mathbb{N}$ such that for each edge (u, v) it holds that $t(u) = t(v)$ if v is a reticulation and $t(u) < t(v)$ otherwise [3].

If F is a finite set and $p \in \mathbb{N}$, then a p-state *character* on F is a function from F to $\{1, \ldots, p\}$. A p-state character is *binary* if $p = 2$. Let α be a p-state character on X and N be a phylogenetic network on X. Then, a p-state character τ on $V(N)$ is an *extension* of α to $V(N)$ if $\tau(x) = \alpha(x)$ for all $x \in X$. Given a p-state character τ on $V(N)$ and an edge $e = (u, v)$ of N, the *change* $c_\tau(e)$ on edge e w.r.t. τ is defined as

$$c_\tau(e) = \begin{cases} 0 \text{ if } \tau(u) = \tau(v), \\ 1 \text{ if } \tau(u) \neq \tau(v). \end{cases}$$

The *hardwired parsimony score* of a phylogenetic network N and a p-state character α on X can be defined as

$$PS_{\mathrm{hw}}(N, \alpha) = \min_{\tau} \sum_{e \in E(N)} c_\tau(e),$$

where the minimum is taken over all extensions τ of α to $V(N)$.

Now, consider a phylogenetic network N on X and a phylogenetic tree T on X, where either both N and T are rooted or both are not. We say that T is *displayed* by N if T can be obtained from a subgraph of N by suppressing nonroot nodes with total degree 2. For a rooted phylogenetic network N, a *switching* of N is obtained by, for each reticulation node, deleting all but one of its incoming edges [21]. We denote the set of switchings of N by N. It can easily be seen that T is displayed by N if and only if T can be obtained from a switching of N by deleting indegree-0 outdegree-1 nodes, deleting unlabelled outdegree-0 nodes, and suppressing indegree-1 outdegree-1 nodes. Let $\mathcal{T}(N)$ denote the set of all phylogenetic trees on X that are displayed by N. The *softwired parsimony score* of a phylogenetic network N and a p-state character α on X can be defined as

$$PS_{\mathrm{sw}}(N, \alpha) = \min_{T \in \mathcal{T}(N)} \min_{\tau} \sum_{e \in E(T)} c_\tau(e),$$

where the second minimum is taken over all extensions τ of α to $V(T)$.

Note that both the hardwired and the softwired parsimony scores can be used for rooted as well as for unrooted networks, although the softwired parsimony score might seem more relevant for rooted networks and the hardwired parsimony score for unrooted ones.

It can easily be seen that if N is a tree, $PS_{\mathrm{hw}}(N, \alpha) = PS_{\mathrm{sw}}(N, \alpha)$. However, we now show that if N is a network, the difference between the two can be arbitrarily

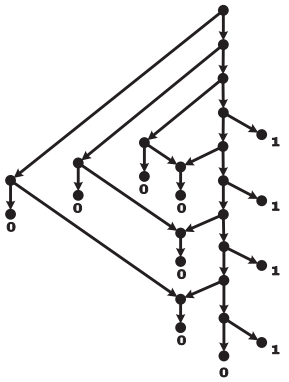

FIG. 1. *Example of a rooted phylogenetic network N and binary character α for which the difference between $PS_{hw}(N,\alpha)$ and $PS_{sw}(N,\alpha)$ is arbitrarily large ($r-1$ if the network is extended to have r reticulations and $n = 3r+2$ leaves) and for which* EXTENDEDFITCH *does not compute an $o(n)$-approximation of $PS_{sw}(N,\alpha)$.*

large.

Figure 1 presents an example of a rooted binary phylogenetic network N and a binary character α, where $PS_{\mathrm{sw}}(N,\alpha) = 2$ regardless of the number of reticulation nodes in N. This is due to the fact that all right-hand-side parental edges of the reticulation nodes could be switched off, such that only one change from 0 to 1 would be required in the resulting tree on the edge just above all taxa labelled "1" and another change from 1 to 0 in the (0,1)-cherry. However, $PS_{\mathrm{hw}}(N,\alpha)$ can be made arbitrarily large by extending the construction in the expected fashion, as in this network we have $PS_{\mathrm{hw}}(N,\alpha) = r+1$, where r denotes the number of reticulation nodes in N. So the difference $PS_{\mathrm{hw}}(N,\alpha) - PS_{\mathrm{sw}}(N,\alpha)$ equals $r-1$, where r can be made arbitrarily large, which shows that $PS_{\mathrm{hw}}(N,\alpha)$ is not an $o(n)$-approximation of $PS_{\mathrm{sw}}(N,\alpha)$, where n is the number of taxa. Note that the construction shown in Figure 1 is binary, tree-child, and time-consistent.

2.1. Extended Fitch algorithm. The hardwired parsimony score for rooted networks was introduced in [19], where a heuristic was proposed by extending the well-known Fitch algorithm for trees. We will call this algorithm EXTENDEDFITCH. Its description can be found in Algorithms 3 and 4 of [19]. We show in this section that EXTENDEDFITCH does not provide a good approximation for the softwired parsimony score and does not compute the hardwired parsimony score optimally.

The example from Figure 1 can be used again, in order to show that EXTENDEDFITCH does not compute an $o(n)$-approximation of the softwired parsimony score. Indeed, EXTENDEDFITCH gives all internal nodes character state 0, leading to a total score of $r+1 = (n+1)/3$ (which is in this case indeed equal to the hardwired parsimony score), while we already showed that $PS_{\mathrm{sw}}(N,\alpha) = 2$. Hence, the approximation ratio of EXTENDEDFITCH is at least $(r+1)/2$ as a function of r and at least $(n+1)/6$ as a function of n.

Note that Theorem 4.7 is, furthermore, complexity-theoretic evidence that EXTENDEDFITCH, a polynomial-time algorithm, cannot approximate the softwired parsimony score of a network well (unless P = NP).

FIG. 2. *Example of a rooted phylogenetic network N and binary character α for which* EXTENDEDFITCH *does not provide the optimal parsimony score $PS_{hw}(N, \alpha)$. The small numbers refer to the two possible internal labellings as suggested by* EXTENDEDFITCH. *Both require two changes on the marked edges. However, the optimal parsimony score is* 1: *If all internal nodes are labelled* 1*, then only one change is needed on the edge from the reticulation node to the leaf labelled* 0.

Next we show that EXTENDEDFITCH does not compute $PS_{\text{hw}}(N, \alpha)$ optimally, even if α is a binary character. Consider Figure 2. This figure displays a rooted phylogenetic network N with three leaves, one of which is directly connected to the only reticulation node. The network is again binary, tree-child, and time-consistent. EXTENDEDFITCH fixes the reticulation node to be in state 0, whereas all other internal nodes can be either 0 or 1. The two optimal solutions found by EXTENDEDFITCH are illustrated by Figure 2; they uniformly set all internal nodes other than the reticulation node either to state 0 or to state 1. Thus, the resulting score is 2, as either both pending edges leading to the leaves labelled 1 need a change or both edges leading to the reticulation node. The most parsimonious solution, however, would be to set all internal nodes—including the reticulation node—to state 1. This way, only one change on the edge from the reticulation node to its pending leaf would be required. Therefore, EXTENDEDFITCH cannot be used to calculate the hardwired parsimony score of a character on a network exactly.

2.2. A comment on model differences. Our core definition of phylogenetic network is slightly less restricted than the definitions given by [17, 19]. However, the strong hardness and inapproximability results we give in this paper still hold under heavy topological and biological restrictions (degree restrictions, tree-child, time-consistent) that are often subsumed into the core definitions given in other papers. Moreover, an obvious advantage of our definition is that all the positive results in the paper apply to the largest possible class of phylogenetic networks.

3. Computing the hardwired parsimony score of a phylogenetic network. Given an undirected graph G and a set Γ of nodes of G called *terminals*, a *multiterminal cut* of (G, Γ) is a subset E' of the edges of G such that each terminal is in a different connected component of the graph obtained from G by removing the edges of E'. A *minimum multiterminal cut* is a multiterminal cut of minimum size. The following theorem shows that computing the hardwired parsimony score of a phylogenetic network is at most as hard as MULTITERMINAL CUT, the problem of finding a minimum multiterminal cut. Note that this was previously observed, without proof, in [31].

Merging a subset X' of the leaves of an unrooted phylogenetic network N produces a graph G that has a node for each node of N that is not in X' plus an additional node v_U. For each edge of N that is not incident to a leaf in X', G has the corresponding edge. In addition, for each edge $\{u, w\}$ of N with $u \in U$, $w \notin U$, G has an edge $\{v_U, w\}$.

THEOREM 3.1. *Let N be an unrooted phylogenetic network on X and α be a p-state character on X. Let G be the graph obtained from N by merging all leaves x with $\alpha(x) = i$ into a single node γ_i for $i = 1, \ldots, p$. Then, the size of a minimum multiterminal cut of $(G, \{\gamma_1, \ldots, \gamma_p\})$ is equal to $PS_{hw}(N, \alpha)$.*

Proof. First consider an extension τ of α to $V(N)$ for which $PS_{\mathrm{hw}}(N, \alpha) = \sum_{e \in E(N)} c_\tau(e)$ (i.e., an optimal extension). Let E' be the set of edges e with $c_\tau(e) = 1$. Since $\tau(\gamma_i) = i$ for $i = 1, \ldots, p$, any path from γ_i to γ_j with $i \neq j$ contains at least one edge of E'. Hence, E' is a multiterminal cut. Moreover, $PS_{\mathrm{hw}}(N, \alpha) = \sum_{e \in E(N)} c_\tau(e) = |E'|$. Hence, $PS_{\mathrm{hw}}(N, \alpha)$ is greater than or equal to the size of a minimum multiterminal cut.

Now consider a minimum multiterminal cut E' of G and let G' be the result of removing the edges in E' from G. We define an extension τ of α to $V(N)$ as follows. First, we set $\tau(x) = \alpha(x)$ for all $x \in X$. Then, for each node v that is in the same connected component of G' as γ_i, set $\tau(v) = i$. Finally, for each remaining node, set $\tau(v) = p$. Then, each edge $e \notin E'$ has $c_\tau(e) = 0$. Consequently, each edge e with $c_\tau(e) = 1$ is in E'. Hence, $PS_{\mathrm{hw}}(N, \alpha) \leq \sum_{e \in E(N)} c_\tau(e) \leq |E'|$. It follows that $PS_{\mathrm{hw}}(N, \alpha)$ is less than or equal to the size of a minimum multiterminal cut, which concludes the proof. □

Although Theorem 3.1 is restricted to unrooted networks, it can easily be extended to rooted networks, since the hardwired parsimony score is independent of the directions of the edges.

COROLLARY 3.2. *Computing the hardwired parsimony score of a phylogenetic network and a binary character is polynomial-time solvable.*

Proof. This follows directly from Theorem 3.1, because, in the case of two terminals, MULTITERMINAL CUT becomes the classical minimum $s-t$-cut problem, which is polynomial-time solvable. □

COROLLARY 3.3. *Computing the hardwired parsimony score of a phylogenetic network and a p-state character, for $p \geq 3$, is NP-hard and APX-hard.*

Proof. We reduce from MULTITERMINAL CUT, which is NP-hard and APX-hard for three or more terminals [5].

Let G be an undirected graph and $\Gamma = \{\gamma_1, \ldots, \gamma_k\}$ be a set of terminals. Note that feasible solutions to MULTITERMINAL CUT must contain all edges between adjacent terminals. For this reason we begin by removing such edges from G. Next we repeatedly delete all degree-1 nodes that are not terminals, until no such nodes are left, because the edges adjacent to such nodes cannot contribute to a multiterminal cut.

We construct a finite set X and a k-state character α on X as follows. For each pair of a terminal γ_i and a node v_i^j adjacent to γ_i in G, put an element x_i^j in X and set $\alpha(x_i^j) = i$. Now we construct a new graph N from G by deleting each γ_i and adding a leaf labelled x_i^j with an edge (v_i^j, x_i^j) for each $x_i^j \in X$. Note that merging, for each i, all leaves x of G with $\alpha(x) = i$ into a single node γ_i would give graph G back. Now, N might contain degree-2 nodes, which are not permitted in our definition of phylogenetic network, but as we explain in Appendix A, there is a simple transformation that removes such nodes without altering the hardwired parsimony score or the cut properties of the graph. We apply this transformation to N if necessary. Suppose then that the resulting graph N is connected and hence an unrooted phylogenetic network on X. Then it follows from Theorem 3.1 that the size of a minimum multiterminal cut of (G, Γ) is equal to $PS_{\mathrm{hw}}(N, \alpha)$.

Now suppose that N is not connected. Observe that the proof of Theorem 3.1

still holds if N is not connected. Moreover, computing the hardwired parsimony score of a connected unrooted phylogenetic network is at least as hard as computing the hardwired parsimony score of a not-necessarily connected phylogenetic network, because we can sum the parsimony scores of the connected components. This reduction is clearly approximation-preserving. Finally we note that computing the hardwired parsimony score of a rooted phylogenetic network is just as hard as computing this score of an unrooted phylogenetic network, because the hardwired parsimony score does not depend on the orientation of the edges. □

COROLLARY 3.4. *Computing the hardwired parsimony score of a phylogenetic network and a p-state character is fixed-parameter tractable (FPT) in the parsimony score. Moreover, there exists a polynomial-time* 1.3438*-approximation for all p and a* $\frac{12}{11}$*-approximation for* $p = 3$.

Proof. The corollary follows from the corresponding results on minimum multiterminal cut [20, 32] by Theorem 3.1. □

4. Complexity of computing the softwired parsimony score of a rooted phylogenetic network. In section 4.1, we consider the complexity of computing the softwired parsimony score exactly. Subsequently, section 4.2 determines the complexity of approximating this score.

4.1. Complexity of computing the softwired parsimony score exactly. In the following, we show that computing the softwired parsimony score of a binary character on a binary rooted phylogenetic network is NP-hard. We reduce from CLUSTER CONTAINMENT, which is known to be NP-hard for general networks [15, 18]. However, in order to prove our result for binary networks, we first need to show that CLUSTER CONTAINMENT is NP-hard for binary phylogenetic networks, too; this intermediate result has to the best of our knowledge not appeared earlier in the literature. We do this via EDGE CLUSTER CONTAINMENT and BINARY EDGE CLUSTER CONTAINMENT, as described in the following. Thus, we state the following questions and analyze their complexity.

(BINARY) EDGE CLUSTER CONTAINMENT
Instance: A set X of taxa, a rooted (binary) phylogenetic network N (with edge set E and node set V) on X, a cluster $C \subset X$, and an edge $e = (u, v) \in E$.
Question: Is there a switching S of N containing the node v such that the taxa descending from v in S are precisely the taxa in C?

If the answer is yes to the question above, we say that e *represents* C. We denote by $\mathcal{C}(N)$ the set of clusters represented by edges in E.

(BINARY) CLUSTER CONTAINMENT
Instance: A set X of taxa, a rooted (binary) phylogenetic network N (with edge set E and node set V) on X, and a cluster $C \subset X$.
Question: Is there a switching S of N which contains an edge $e = (u, v) \in E$ such that the taxa descending from v in S are precisely the taxa in C?

The following observation follows directly by a simple (Turing) reduction from CLUSTER CONTAINMENT.

OBSERVATION 1. EDGE CLUSTER CONTAINMENT *is NP-hard.*

Next we use the previous observation to prove the following small result.

OBSERVATION 2. BINARY EDGE CLUSTER CONTAINMENT *is NP-hard.*

Proof. We reduce from EDGE CLUSTER CONTAINMENT. Assume there is an algorithm $\mathcal{A}$ to decide BINARY EDGE CLUSTER CONTAINMENT in polynomial time. Let N be a phylogenetic network on X containing an edge e. We want to know whether e represents a particular cluster C. Let N^B be an arbitrary binary refinement of N.

Note that N^B contains all edges of N, and possibly some more (unless N is already binary), in the sense that edges in N^B could be contracted to once again obtain N. Hence, e is contained in N^B, too. An example of a binary refinement of a nonbinary network is depicted in Figure 3. So we can use $\mathcal{A}$ to decide whether e represents C in N^B. Note that e represents C in N^B if and only if e represents C in N, because it is easy to see that refining a network does not change the clusters pending on a particular edge. Therefore, this method would provide a polynomial-time algorithm to solve Edge Cluster Containment. □

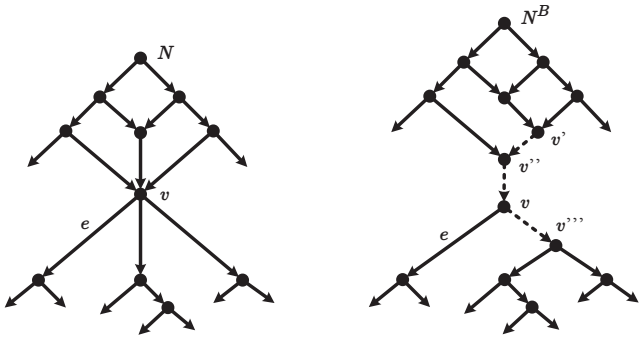

Fig. 3. *Illustration of a rooted phylogenetic network N with a node v of total degree 6 and a possible binary refinement N^B of N, where three copies of v, namely v', v'', and v''', as well as three new edges (dashed lines) are inserted. Note that each edge of N has a unique image in N^B.*

Now we are in a position to prove that Binary Cluster Containment is NP-hard, which is the essential ingredient to our proof of Theorem 4.2.

Lemma 4.1. Binary Cluster Containment *is NP-hard.*

Proof. We reduce from Binary Edge Cluster Containment. Let N^B be a rooted binary phylogenetic network on X and C be a cluster of X. Assume there is an algorithm $\mathcal{A}$ to answer Binary Cluster Containment in polynomial time. Let $e = (v, u)$ be an edge in N^B. We add two new nodes v_1, v_2 to N^B as follows: Subdivide e into three edges $e_1 := (v, v_1)$, $e_2 := (v_1, v_2)$, and $e_3 := (v_2, u)$. Now introduce two new edges $e_4 := (v_1, h_2)$ and $e_5 := (v_2, h_1)$, where h_1 and h_2 are two new taxa. We call the resulting modified network $\tilde{N}^B$. An example of this transformation is depicted in Figure 4.

Note that by construction, $\tilde{N}^B$ is binary. We now use algorithm $\mathcal{A}$ to decide in polynomial time whether $\tilde{N}^B$ contains the cluster $C \cup \{h_1\}$. Note that this is the case if and only if e_2 in $\tilde{N}^B$ represents C, which, by construction, is the case if and only if e represents C in N^B. Therefore, this method would provide a polynomial-time algorithm to solve Binary Edge Cluster Containment. □

The following theorem was shown by [17] for nonbinary networks. The advantage of the proof given below is that it shows that the problem is even NP-hard for binary networks, demonstrates a direct and insightful relationship between cluster containment and parsimony, and leads directly to the conclusion that the problem is not even FPT (unless P = NP).

Theorem 4.2. *Computing the softwired parsimony score of a binary character on a binary rooted phylogenetic network is NP-hard.*

Proof. We reduce from Binary Cluster Containment. Let N be a rooted binary phylogenetic network on taxon set X and $C \subset X$ be a cluster. Then, by

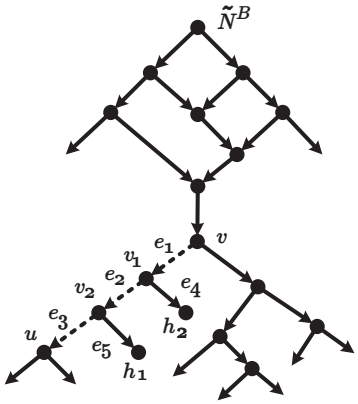

FIG. 4. *Illustration of the modifications applied to N^B as depicted by Figure* 3*, resulting in the modified binary network $\tilde{N}^B$.*

definition of $\mathcal{C}(N)$, C is in $\mathcal{C}(N)$ if and only if there is a tree T on X displayed by N with an edge $e = (u, v)$ such that the taxa descending from v in T are precisely the elements of C. This is the case if and only if v is the root of a subtree of T with leaf set C. Now assume that there is an algorithm $\mathcal{A}$ to compute the softwired parsimony score of a binary character on a rooted binary phylogenetic network in polynomial time. Then, we can solve BINARY CLUSTER CONTAINMENT by the following algorithm $\tilde{\mathcal{A}}$:

1. Introduce a modified version $\hat{N}$ of N as follows: Add an additional taxon z to N and a new node $\hat{\rho}$ as well as the edges $(\hat{\rho}, z)$ and $(\hat{\rho}, \rho)$, where ρ is the root of N. Thus, the taxon set $\hat{X}$ of $\hat{N}$ is $X \cup \{z\}$ and the root of $\hat{N}$ is $\hat{\rho}$.
2. Construct a binary character α on $\hat{X}$ as follows:

$$\alpha(x) := \begin{cases} 1 & \text{if } x \in C, \\ 0 & \text{if } x \in \hat{X} \setminus C. \end{cases}$$

   Note that $\alpha(z) = 0$ as $z \notin X$ and thus $z \notin C$.
3. Calculate the parsimony score $PS_{\text{sw}}(\hat{N}, \alpha)$ using algorithm $\mathcal{A}$.

Note that $PS_{\text{sw}}(\hat{N}, \alpha) = 1$ if and only if N displays a tree T which has a subtree with label set C. This is due to the fact that, as $\alpha(z) = 0$, the softwired parsimony score of $\hat{N}$ can only be 1 if ρ and $\hat{\rho}$ receive state 0. Otherwise, there would be a change required on one of the edges $(\hat{\rho}, z)$ or $(\hat{\rho}, \rho)$ and additionally at least one more change in the part of $\hat{N}$ corresponding to N, as X employs both states 1 and 0 for taxa in or not in C, respectively, because $C \subsetneq X$. Moreover, if ρ is in state 0, the softwired parsimony score of $\hat{N}$ is 1 precisely if $\hat{N}$ displays a tree T which requires only one change, and that change has to be a change from 0 to 1. This is the case if and only if N displays a tree T with a subtree with leaf labels C. This case is illustrated by Figure 5. Note that if $\mathcal{A}$ is polynomial, so is $\tilde{\mathcal{A}}$. Therefore, computing the softwired parsimony score of a binary character on a binary rooted phylogenetic network is NP-hard. □

We can extend the NP-hardness result to a more restricted class of rooted phylogenetic networks.

THEOREM 4.3. *Computing the softwired parsimony score of a binary tree-child time-consistent rooted phylogenetic network and a binary character is NP-hard.*

Proof. We can make any network tree-child and time-consistent by doing the following for each reconciliation edge (u, r): adding a node u', replacing (u, r) with two edges (u, u') and (u', r), and adding a new cherry with an edge from u' to the root of the cherry. If we give the two leaves of each cherry character states 0 and 1, then the softwired parsimony score is increased exactly by the number of added cherries. ∎

COROLLARY 4.4. *It is NP-hard to decide whether the softwired parsimony score of a binary rooted time-consistent phylogenetic network and a binary character is equal to one. In particular, there is no FPT algorithm with the parsimony score as a parameter unless $P = NP$.*

Proof. It has been proven in [15] that CLUSTER CONTAINMENT is NP-hard even for time-consistent networks by reducing from CLUSTER CONTAINMENT on general networks. Since the proof in [15] transforms the input network in a way that preserves binarity, the corollary follows directly from the combination of the result by [15] and Theorem 4.2. ∎

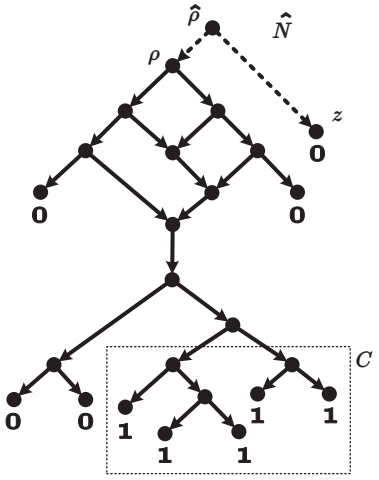

FIG. 5. *Illustration of the extension of a rooted binary phylogenetic network N (solid lines) to the rooted binary phylogenetic network $\hat{N}$ as described in the proof of Theorem 4.2. The additional taxon z is assigned state 0 along with all taxa in $X \setminus C$, whereas all taxa in C are assigned state 1. Then, the softwired parsimony score of $\hat{N}$ is 1 if and only if N displays a tree T with a pending subtree with leaf set C.*

4.2. Complexity of approximating the softwired parsimony score. In order to proceed to the question of approximability, we require a new definition and a lemma. Recall that $\mathcal{S}(N)$ is the set of all switchings of a network. Given a rooted phylogenetic network N, we define $PS_{\mathcal{S}}(N, \alpha)$ as

$$\min_{S \in \mathcal{S}(N)} \min_{\tau} \sum_{e \in E(S)} c_\tau(e),$$

where the second minimum is taken over all extensions τ of α to $V(S)$.

The following lemma states that optimal solutions can equivalently be modelled as selecting the lowest-score switching, ranging over all extensions τ of a character α to the nodes of the *network*. This is the characterization of optimality used in sec-

tion 6 and enables us to circumvent some of the suppression and deletion technicalities associated with the concept "display." Since the lemma is intuitively clear, we defer its proof to Appendix B.

LEMMA 4.5. *Consider a rooted phylogenetic network N on X and a p-state character α on X. Then*

$$PS_{\mathcal{S}}(N, \alpha) = PS_{sw}(N, \alpha).$$

The next straightforward corollary will be useful when describing approximation-preserving reductions.

COROLLARY 4.6. *Given a network N on X and a character α on X, a tree $T \in \mathcal{T}(N)$, a switching $S \in \mathcal{S}(N)$ corresponding to T, and an extension τ of α to $V(T)$, we can construct in polynomial time an extension τ' of α to $V(S)$ such that $\sum_{e \in E(S)} c_{\tau'}(e) = \sum_{e \in E(T)} c_{\tau}(e)$.*

We now show that it is very hard to approximate the softwired parsimony score on rooted networks. We give two inapproximability results. The first, the stronger of the two, applies to nonbinary networks and holds even when the network is both tree-child and time-consistent. It shows that in a complexity-theoretic sense trivial approximation algorithms are the best one can hope for in this case. The second result, which is only slightly weaker, applies to binary networks. Both results are much stronger than the APX-hardness result presented by [17]. At the present time we do not have an inapproximability result for networks that are simultaneously binary and tree-child (and time-consistent): in this sense Theorem 4.3 is currently the strongest hardness result we have for such networks.

Before proceeding we formally define the output of an algorithm that approximates $PS_{\mathrm{sw}}(N, \alpha)$ as a tree $T \in \mathcal{T}(N)$ and a certificate that $T \in \mathcal{T}(N)$, i.e., a switching $S \in \mathcal{S}(N)$ corresponding to T. The certificate is useful, because it is NP-hard to determine whether a tree is displayed by a network [18]. The parsimony score (i.e., value of the objective function) associated with the output T is then

$$PS(T, \alpha) = \min_{\tau} \sum_{e \in E(T)} c_{\tau}(e),$$

where the minimum is taken over all extensions τ of α to $V(T)$. Note that $PS(N, \alpha) = PS_{sw}(N, \alpha) = PS_{hw}(N, \alpha)$ holds when N is a phylogenetic tree. Note also that $PS(T, \alpha)$ and a corresponding extension τ can easily be found in polynomial time by applying Fitch's algorithm to T. If necessary, Corollary 4.6 can then be applied to transform this in polynomial time into an extension τ' of α to $V(N)$ such that the switching S has parsimony score at most $PS(T, \alpha)$. We note that Theorems 4.7 and 4.8 below even hold for approximation algorithms that output an approximate parsimony score instead of a tree and a switching.

Consider the following simple observation.

OBSERVATION 3. *The softwired parsimony score of a rooted phylogenetic network N on X and a p-state character α on X can be (trivially) approximated in polynomial time with approximation factor $|X|$ for any $p \geq 2$.*

Proof. Let $s \in \{1, \ldots, p\}$ be a state to which at least a fraction $1/p$ of X is mapped by α. Let T be an arbitrary tree in $\mathcal{T}(N)$. We extend α to $V(T)$ by labelling all internal nodes of T with s. Clearly, $PS_{sw}(N, \alpha) = 0$ if and only if α maps all elements in X to the same character state, in which case the extension of α to $V(T)$ also yields a parsimony score of 0. Otherwise, $PS_{sw}(N, \alpha) \geq 1$ and the extension

described yields a parsimony score of at most $(1 - 1/p)|X| < |X|$, from which the result follows. □

The following theorem shows that, in an asymptotic sense, Observation 3 is actually the best result possible, even when the topology of the network is quite heavily restricted.

THEOREM 4.7. *For every constant $\epsilon > 0$ there is no polynomial-time approximation algorithm that can approximate $PS_{sw}(N, \alpha)$ to a factor $|X|^{1-\epsilon}$, where N is a tree-child, time-consistent network and α is a binary character on X, unless $P = NP$.*

Proof. We reduce from the NP-hard decision problem 3-SAT. This is the problem of determining whether a boolean formula in CNF form, where each clause contains at most three literals, is satisfiable. Let $B = (V, C)$ be an instance of 3-SAT, where V is the set of variables and C is the set of clauses. Let $|V| = n$. Observe that $|C| = m$ is at most $O(n^3)$, because in a decision problem it makes no sense to include repeated clauses.

For each constant $\epsilon > 0$, we will show how to construct a parsimony instance (N, α) such that the existence of a polynomial-time $|X|^{1-\epsilon}$-approximation would allow us to determine in polynomial time whether B is a YES or a NO instance, from which the theorem will follow. The construction can be thought of as an "inapproximability" variant of the hardness construction used in [18].

Throughout the proof we will make heavy use of the equivalence described in Lemma 4.5. Specifically, we will characterize optimal solutions to the softwired parsimony problem as the score yielded by the lowest-score switching, ranging over all extensions of α to $V(N)$.

We begin by proving the result for networks that are time-consistent but not tree-child. Later we will show how to extend the result to networks that are time-consistent and tree-child.

The centerpiece of the construction is the following *variable gadget*. Let z be a variable in V. We introduce two nodes, which we refer to as z and $\neg z$ and collectively call *connector* nodes. We introduce two sets of taxa, $X_{z,0}$ and $X_{z,1}$, each containing $f(n, \epsilon)$ taxa, where $f(n, \epsilon)$ is a function that we will specify later. For each taxon $x \in X_{z,i}$ we set $\alpha(x) = i$. By introducing $2 \cdot f(n, \epsilon)$ reticulation nodes we connect each taxon in $X_{z,0}$ and $X_{z,1}$ to both z and $\neg z$ (see Figure 6). Observe that if both z and $\neg z$ are labelled with the same character state, the parsimony score of this gadget (and thus of the network as a whole) will be at least $f(n, \epsilon)$. On the other hand, if z and $\neg z$ are labelled with different character states, the gadget contributes (locally) zero to the parsimony score. The idea is thus that we label $(z, \neg z)$ with $(1, 0)$ if we wish to set variable z to be TRUE and $(0, 1)$ if we wish z to be FALSE, i.e., $\neg z$ is TRUE. By choosing $f(n, \epsilon)$ to be very large we will ensure that z and $\neg z$ are never labelled with the same character state in "good" solutions.

We construct one variable gadget for each $z \in V$. Next we add the root ρ and two nodes s_0 and s_1. We connect ρ to s_0 and to s_1. Next we connect s_0 (respectively, s_1) to every connector node (ranging over all variable gadgets). Hence, every connector node has indegree 2. The idea is that (without loss of generality) s_0 (respectively, s_1) can be assumed to be labelled 0 (respectively, 1). Therefore, if a connector node is labelled with state 0 (respectively, 1), it will choose s_0 (respectively, s_1) to be its parent, and these edges will not contribute any mutations to the parsimony score. There are two points to note here. First, there will be exactly one mutation incurred on the two edges (ρ, s_0) and (ρ, s_1), and this has an important role in the ensuing inapproximability argument; we shall return to this later. Second, it could happen

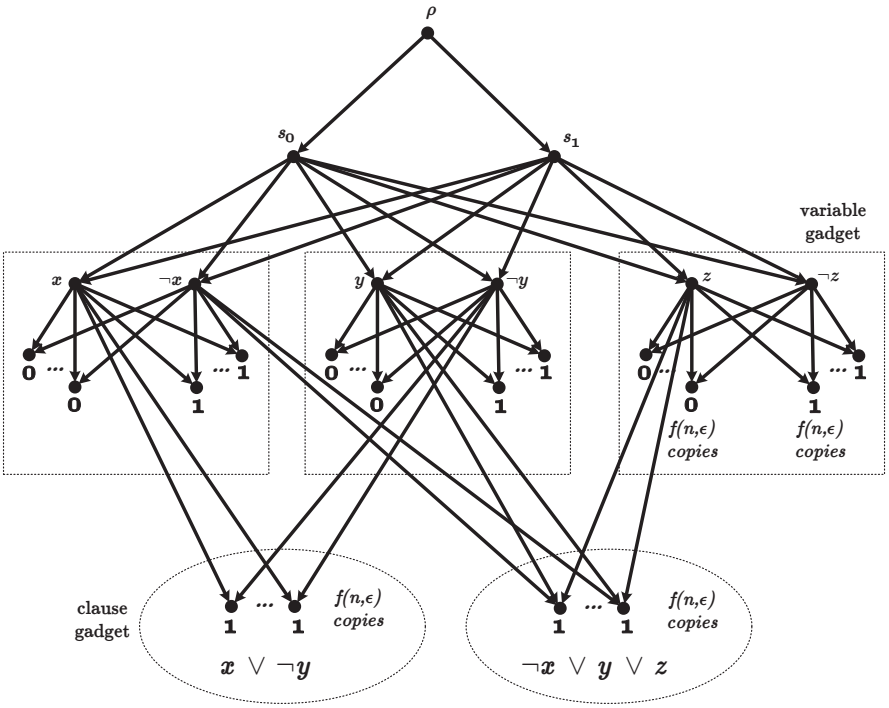

FIG. 6. *An encoding of the* 3*-SAT instance* $(x \vee \neg y) \wedge (\neg x \vee y \vee z)$ *as described in Theorem* 4.7. *Note that the network is time-consistent—a possible time-stamp allocates* 1 *to the root,* 2 *to all reticulation nodes plus* s_0 *and* s_1*, and* 3 *to the nodes in clause gadgets for clauses consisting of only one literal (as these are tree nodes)—but not tree-child: a slight modification is required to make it tree-child.*

that the labelling of s_0 and s_1 is $(1, 0)$ rather than $(0, 1)$, but in that case the analysis is entirely symmetrical.

It remains only to describe the *clause gadgets*. These are very simple. For each clause $c \in C$ we introduce a size $f(n, \epsilon)$ set of taxa that we call X_c. For each taxon $x \in X_c$ we set $\alpha(x) = 1$. By introducing $f(n, \epsilon)$ nodes—these will be reticulations, unless the clause contains only one literal—we connect each taxon in X_c to the connector nodes in the variable gadgets corresponding to the literals in the clause. For example, if c is the clause $(\neg x \vee y \vee z)$, each node in X_c has $\neg x$, y, and z as its parents. Similarly to the variable gadgets, observe that if none of the literals corresponding to c is labelled 1 (i.e., set to TRUE), the clause gadget corresponding to c will raise the parsimony score by at least $f(n, \epsilon)$, but if that at least one literal is TRUE, the (local) parsimony cost will be zero.

Observe first that $PS_{sw}(N, \alpha) \geq 1$, because both character states appear in the range of α. More fundamentally, $PS_{sw}(N, \alpha) = 1$ if B is satisfiable—in which case the single mutation occurs on one of the edges (ρ, s_0) and (ρ, s_1)—and $PS_{sw}(N, \alpha) \geq f(n, \epsilon)$ if B is unsatisfiable. This dichotomy holds, because, to have $PS_{sw}(N, \alpha) < f(n, \epsilon)$, it is necessary that the connector nodes in the variable gadgets always have a labelling of the form $(0, 1)$ or $(1, 0)$ and that for every clause c the nodes in X_c all have at least one TRUE parent, i.e., B is satisfiable.

The high-level idea is to choose $f(n,\epsilon)$ to be so large that even a weak approximation factor will be sufficient to determine without error whether B is satisfiable or unsatisfiable. The reduction is as follows. Let $T \in \mathcal{T}(N)$ be the tree produced by the approximation algorithm for $PS_{sw}(N,\alpha)$, let $S \in \mathcal{S}(N)$ be a corresponding switching, and let $PS(T,\alpha)$ be the corresponding parsimony score. If $PS(T,\alpha) \geq f(n,\epsilon)$, we declare that the SAT instance B is unsatisfiable. Otherwise, we declare that B is satisfiable. Note that this reduction is also valid when the approximation algorithm outputs a parsimony score instead of a tree and a switching. If $PS(T,\alpha) < f(n,\epsilon)$, we can also find a corresponding satisfying truth assignment for B. For this we do need the tree T and switching S in order to find an extension τ' of α to $V(N)$ such that the parsimony score of S under τ' is also strictly less than $f(n,\epsilon)$. For each variable z in the SAT instance, τ' has to label z and $\neg z$ with different character states, and for each clause $c \in C$ in the SAT instance, at least one of its literals has to be labelled with character state 1. The satisfying assignment is thus the following: for each variable z, z is TRUE if z is labelled 1 and FALSE if $\neg z$ is labelled 1.

We now show that the reduction is correct assuming that we choose $f(n,\epsilon)$ such that $|X|^{1-\epsilon} < f(n,\epsilon)$. First assume that B is satisfiable. Recall that in this case $PS_{sw}(N,\alpha) = 1$. Hence, an approximation algorithm with approximation ratio at most $|X|^{1-\epsilon}$ will return a solution with parsimony score at most $|X|^{1-\epsilon} < f(n,\epsilon)$. Therefore, the reduction above correctly decides that B is satisfiable. Now assume that B is not satisfiable. In that case, $PS_{sw}(N,\alpha) \geq f(n,\epsilon)$. Hence an approximation algorithm has to produce a solution with parsimony score at least $f(n,\epsilon)$ and the reduction above correctly decides that B is not satisfiable.

It remains to show that we can choose $f(n,\epsilon)$ such that $|X|^{1-\epsilon} < f(n,\epsilon)$. Observe that $|X| = 2n \cdot f(n,\epsilon) + m \cdot f(n,\epsilon)$. Given the relationship between n and m, a (crude) upper bound on $|X|$ is $n^5 \cdot f(n,\epsilon)$ for sufficiently large n. Hence it is sufficient to ensure $f(n,\epsilon)^{1-\epsilon} \cdot n^{5(1-\epsilon)} < f(n,\epsilon)$. Suppose $f(n,\epsilon) = n^{g(\epsilon)}$, where $g(\epsilon)$ is a function that depends only on ϵ. Then we need $g(\epsilon)(1-\epsilon) + 5(1-\epsilon) < g(\epsilon)$, which implies that taking $g(\epsilon) = \lceil 6\epsilon^{-1}(1-\epsilon) \rceil$ is sufficient.

The network we constructed above is time-consistent (see Figure 6) but not tree-child: potentially only the root ρ has at least one child that is not a reticulation. We can transform the network as follows. For each node v with indegree greater than 1 and outdegree 0 we simply add an outgoing edge to a new node v', where v' receives time-stamp 3, and $\alpha(v')$ takes over the character state $\alpha(v)$. Next we introduce $2n+2$ new taxa. For s_i, $i \in \{0,1\}$, we introduce a new node s_i' (with time-stamp 3), add an edge (s_i, s_i'), and set $\alpha(s_i') = i$. For each variable z in the SAT instance B, we introduce two new taxa z' and $\neg z'$ (both of which receive time-stamp 3), add edges (z, z') and $(\neg z, \neg z')$, and set $\alpha(z') = \alpha(\neg z') = 0$. The network is now both tree-child and time-consistent. Now, observe that the two taxa introduced underneath s_0 and s_1 do not change the optimum parsimony score, because without loss of generality we can assume that s_0 is labelled 0 and s_1 is labelled 1. However, for each variable z, some extra mutations might be incurred on the edges (z, z') and $(\neg z, \neg z')$. As long as $f(n,\epsilon)$ is chosen to be large enough, these (at most) $2n$ extra mutations do not significantly alter the reduction: in optimal solutions each $(z, \neg z)$ pair will still be labelled with different character states, reflecting a satisfying truth assignment for B, whenever B is satisfiable. In fact, if B is satisfiable, then exactly n extra mutations will be incurred on the edges (z, z') and $(\neg z, \neg z')$ (in an optimal solution), since at least one of z and $\neg z$ will be labelled 0. So, if B is satisfiable, $PS_{sw}(N,\alpha) = n+1$, and if B is unsatisfiable, $PS_{sw}(N,\alpha) \geq f(n,\epsilon)$. As long as we choose $|X|^{1-\epsilon}(n+1) < f(n,\epsilon)$, any

$|X|^{1-\epsilon}$-approximation will be forced to return a tree T such that $PS(T,\alpha) < f(n,\epsilon)$ whenever B is satisfiable, and this can be transformed in the same way as before in polynomial time into a satisfying assignment for B.

Hence we need to choose $f(n,\epsilon)$ such that $|X|^{1-\epsilon}(n+1) < f(n,\epsilon)$, where this time $|X| = 2n \cdot f(n,\epsilon) + m \cdot f(n,\epsilon) + (2n+2)$. As before, $|X| \leq n^5 \cdot f(n,\epsilon)$ holds for sufficiently large n, as does $n+1 < n^2$. So establishing $f(n,\epsilon) > n^{(7-5\epsilon)} f(n,\epsilon)^{1-\epsilon}$ would be sufficient. Letting $f(n,\epsilon) = n^{g(\epsilon)}$ and taking logarithms, it is sufficient to choose $g(\epsilon)$ such that $g(\epsilon) > 7 - 5\epsilon + g(\epsilon)(1-\epsilon)$. Taking $g(\epsilon) = \lceil \frac{7-5\epsilon}{\epsilon} \rceil + 1$ is sufficient for this purpose, and we are done. ∎

For binary networks, we get a slightly weaker inapproximability result.

THEOREM 4.8. *For every constant $\epsilon > 0$ there is no polynomial-time approximation algorithm that approximates $PS_{sw}(N,\alpha)$ to a factor $|X|^{\frac{1}{3}-\epsilon}$, where N is a rooted binary phylogenetic network on X and α is a binary character on X, unless $P = NP$.*

Proof. We reduce again from 3-SAT. Let, as before, $B = (C,V)$ be an instance of 3-SAT. Let $V = \{v_1, \ldots, v_n\}$ and $F := f(n,\epsilon)$. We will describe a construction of a rooted binary phylogenetic network N and binary character α. The first part of the construction is essentially a binary version of the network constructed in the proof of Theorem 4.7. The main difference will be the construction of the so-called zero-gadgets. For ease of notation, we will create vertices with indegree 1 and outdegree 1, which could be suppressed, and we create reticulations with indegree greater than 2, which could be refined arbitrarily. Observe that neither suppressing indegree-1 and outdegree-1 vertices nor refining reticulations with indegree greater than 2 alters the softwired parsimony score. However, to simplify the proof we do not suppress or refine these vertices. Furthermore, we assume without loss of generality that each literal is contained in at least one clause.

Our construction is as follows. We create a root ρ with two directed paths $(\rho, a_1, \ldots, a_{2n})$ and $(\rho, b_1, \ldots, b_{2n})$ leaving it. Then, for each variable v_i, we create a reticulation vertex which we will also call v_i and has reticulation edges $(a_{2n-2i+2}, v_i)$, (b_{2i-1}, v_i). Moreover, we create a reticulation vertex $\neg v_i$ with reticulation edges $(a_{2n-2i+1}, \neg v_i), (b_{2i}, \neg v_i)$. The vertices v_i and $\neg v_i$ are called "literal vertices." Then, for each clause c, create F vertices $c_1, \ldots, c_F$, which we will call "clause vertices," and for each such clause vertex c_f, create an edge (c_f, c'_f) to a new leaf c'_f with character state $\alpha(c'_f) = 1$ (the "clause leaves"). So, in total we have $|C|F$ clause leaves.

We now connect the literal vertices to the clause vertices. For each variable x and its negation $\neg x$, we do the following. Suppose that x is in U clauses, corresponding to $F \cdot U$ clause vertices, $c_x^1, \ldots, c_x^{F \cdot U}$. Create a directed path $(x, d_x^1, \ldots, d_x^{F \cdot U})$ and edges (d_x^k, c_x^k) for $k = 1, \ldots, F \cdot U$. Similarly, if $\neg x$ is in Υ clauses with $F \cdot \Upsilon$ corresponding clause vertices $(\gamma_x^1, \ldots, \gamma_x^{F \cdot \Upsilon})$, we create a directed path $(\neg x, \delta_x^1, \ldots, \delta_x^{F \cdot \Upsilon})$ and edges $(\delta_x^\kappa, \gamma_x^\kappa)$ for $\kappa = 1, \ldots, F \cdot \Upsilon$. For each pair k and κ with $k \in \{1, \ldots, F \cdot U\}$ and $\kappa \in \{1, \ldots, F \cdot \Upsilon\}$, we create the following "zero-gadget." Let u be the parent of c_x^k that is reachable from x (hence, in the first iteration, $u = d_x^k$). Replace the edge (u, c_x^k) by a directed path $(u, u_1, \ldots, u_F, c_x^k)$. Similarly, let μ be the parent of γ_x^κ that is reachable from $\neg x$ (initially, $\mu = \delta_x^\kappa$), and replace the edge (μ, γ_x^κ) by a directed path $\mu, \mu_1, \ldots, \mu_F, \gamma_x^\kappa$. Then, for $f = 1, \ldots, F$, create a new reticulation z_f, with reticulation edges (u_f, z_f) and (μ_f, z_f), and an edge (z_f, z'_f) to a new leaf z'_f with character state $\alpha(z'_f) = 0$.

See Figure 7 for an example of the construction for $F = 1$. For larger F, the construction is similar but with more copies of each clause vertex and more copies of each zero-gadget.

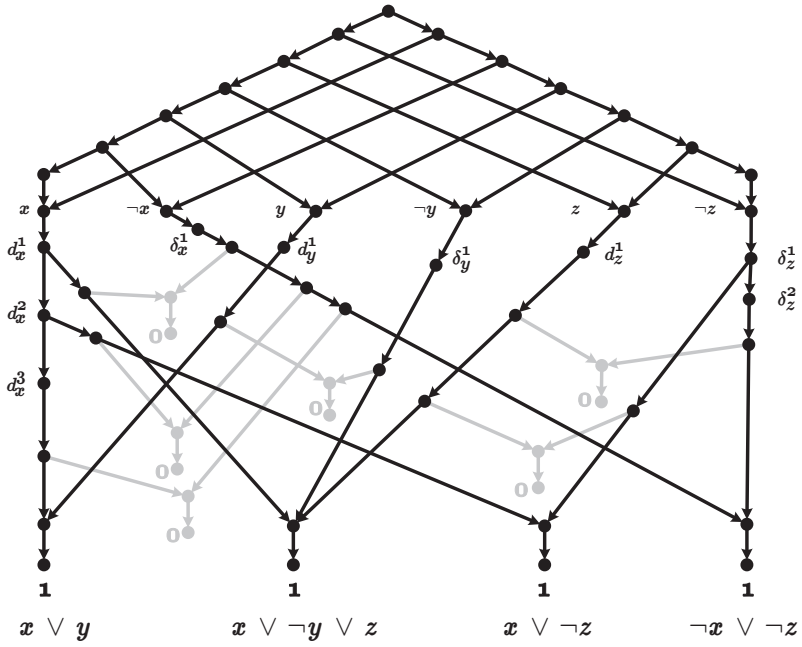

FIG. 7. *An encoding of the* 3-*SAT instance* $(x \vee y) \wedge (x \vee \neg y \vee z) \wedge (x \vee \neg z) \wedge (\neg x \vee \neg z)$ *as described in Theorem* 4.8*, with* $F = 1$*. The zero-gadgets are indicated in grey.*

The constructed network N has $|C|F$ clause leaves (all having character state 1) and at most $|V||C|^2F^3$ leaves for the zero-gadgets (all having character state 0). Hence, the total number of leaves is at most $|C|F + |V||C|^2F^3$.

We will show that if (C, V) is satisfiable, $PS(N_{\mathrm{sw}}, \alpha) = 1$, and if (C, V) is not satisfiable, $PS(N_{\mathrm{sw}}, \alpha) \geq F$. We will use the following definitions. For two vertices u and v, we say that v is a *tree-descendant* of u if v is reachable from u by a directed path that does not contain any reticulations apart from possibly u. In particular, each vertex is a tree-descendant of itself. Furthermore, given an extension τ of α to $V(N)$, we say that there is a *change at* vertex v if the character state of v is different from the character states of all its parents. The *number of changes* of network N and extension τ is the number of vertices at which there is a change. It follows from Lemma 4.5 that the softwired parsimony score of a network N and character α is equal to the minimum number of changes over all possible extensions τ of α to $V(N)$.

First suppose that (C, V) is satisfiable. Then, given a satisfying truth assignment, we can assign character states as follows. All vertices on the path $(\rho, a_1, \ldots, a_{2n})$ and all clause vertices c_f receive state 1. All vertices on the path $(b_1, \ldots, b_{2n})$ and all reticulations z_f of zero-gadgets receive state 0. For each variable x that is set to true by the truth assignment, we give state 1 to all tree-descendants of literal vertex x and state 0 to all tree-descendants of literal vertex $\neg x$. Similarly, for each variable x that is set to false by the truth assignment, we give state 0 to all tree-descendants of literal vertex x and state 1 to all tree-descendants of literal vertex $\neg x$. This concludes the assignment of character states. Now observe the following. Consider a clause c and a corresponding clause vertex c_f, which has state 1. Since c is satisfied by the truth

assignment, at least one parent of c_f also has state 1. Hence, there are no changes at the clause vertices. Moreover, for each reticulation z_f of a zero-gadget (which has state 0), there is at least one parent that also has state 0, because for each variable x either all tree-descendants of x or all tree-descendants of $\neg x$ have state 0. Using these observations, it can easily be checked that the only change is at b_1. Hence, if (C, V) is satisfiable, $PS(N_{\text{sw}}, \alpha) = 1$.

Next, we show that if (C, V) is not satisfiable, $PS(N_{\text{sw}}, \alpha) \geq F$. We do this by assuming that $PS(N_{\text{sw}}, \alpha) < F$ and showing that this implies that (C, V) is satisfiable. Let τ be an extension of α to $V(N)$ with less than F changes. For a positive literal x and clause vertex c_x^k for a clause containing x, let $P(x, c_x^k)$ denote the directed path from d_x^k to c_x^k (with d_x^k as defined in the construction of N). Similarly, for a negative literal $\neg x$ and clause vertex γ_x^κ for a clause containing $\neg x$, let $P(\neg x, \gamma_x^\kappa)$ denote the directed path from δ_x^κ to γ_x^κ (with δ_x^κ as defined in the construction of N). Moreover, for any literal ℓ (of the form x or $\neg x$) and clause vertex c_f for a clause containing ℓ, let $P'(\ell, c_f)$ denote path $P(\ell, c_f)$ excluding its first vertex. We compute a truth assignment as follows. A variable x is set to true if and only if for some clause vertex c_x^k for a clause containing x it holds that all vertices on the path $P(x, c_x^k)$ have state 1. We now prove that the obtained truth assignment is a satisfying truth assignment. Assume that a certain clause c is not satisfied. Consider a clause vertex c_f corresponding to clause c. Observe that, for two different clause vertices c_{f_1}, c_{f_2} corresponding to clause c and for any two literals ℓ_1, ℓ_2 contained in clause c, the paths $P'(\ell_1, c_{f_1}), P'(\ell_2, c_{f_2})$ are vertex-disjoint. Hence, since τ has less than F changes, there exists at least one clause vertex c_f corresponding to clause c for which there are no changes at c'_f or at any vertex on a directed path $P'(\ell, c_f)$ for any literal ℓ contained in clause c. Since c'_f has state 1, it follows that c_f has state 1, and hence that at least one parent of c_f has state 1, and hence that there exists at least one literal ℓ contained in clause c such that all vertices on the path $P(\ell, c_f)$ have state 1. If ℓ is of the form x (a positive literal), then this immediately implies that ℓ is set to true, contradicting the assumption that clause c is not satisfied. Now consider the case that ℓ is of the form $\neg x$ (a negative literal). Let κ be such that $\gamma_x^\kappa = c_f$. We have shown that all vertices on the path $P(\ell, c_f)$ from δ_x^κ to $\gamma_x^\kappa = c_f$ have state 1. For every $k \in \{1, \ldots, F \cdot U\}$, there is a zero-gadget for x, k, and κ. Each such zero-gadget contains a directed path $(u_1, \ldots, u_F)$ on $P(x, c_x^k)$ and a directed path $\mu_1, \ldots, \mu_F$ on $P(\neg x, \gamma_x^\kappa)$. Since all vertices on the path $P(\neg x, \gamma_x \kappa)$ have state 1 and there are less than F changes, at least one vertex of the path $(u_1, \ldots, u_F)$ has state 0. Hence, at least one vertex on $P(x, c_x^k)$ has state 0 for all $k \in \{1, \ldots, F \cdot U\}$. It follows that x is set to false and hence that literal $\ell = \neg x$ is set to true, contradicting the assumption that clause c is not satisfied. Therefore, we have shown that if (C, V) is not satisfiable, $PS(N_{\text{sw}}, \alpha) \geq F$.

It remains to describe how to choose $F = f(n, \epsilon)$ such that $|X|^{\frac{1}{3}-\epsilon} < f(n, \epsilon)$. Recall that $|X| \leq |C| f(n, \epsilon) + |V||C|^2 f(n, \epsilon)^3$. Then, with $n = |V|$ and recalling that $|C| = O(n^3)$, we can bound this by $|X| \leq n^8 f(n, \epsilon)^3$ for sufficiently large n. Hence, it is enough to show that $n^{8(\frac{1}{3}-\epsilon)} f(n, \epsilon)^{3(\frac{1}{3}-\epsilon)} < f(n, \epsilon)$. Taking $f(n, \epsilon) = n^{g(\epsilon)}$, we need $8(\frac{1}{3} - \epsilon) + 3(\frac{1}{3} - \epsilon) g(\epsilon) < g(\epsilon)$. Hence, it is sufficient to take $g(\epsilon) = \lceil \frac{8}{9\epsilon} \rceil$. ∎

In particular, Theorem 4.8 shows that there can be no $O(\log(|X|))$-approximation for computing the softwired parsimony score of a binary rooted phylogenetic network, unless P = NP. We remark that the network constructed in the proof of Theorem 4.8 cannot easily be made tree-child. Hence the inapproximability of binary tree-child

networks is still open. It does seem that the constructed network can be made time-consistent, but we omit a proof.

Although we have shown above that there is no algorithm for computing the softwired parsimony score that is FPT in the parsimony score (unless P = NP), there obviously exists such an algorithm that is FPT in the reticulation number of the network: a network with reticulation number r has 2^r switchings, and for each switching Fitch's algorithm can be used. Moreover, in the next section we show that there even exists an algorithm that is FPT in the level of the network, a parameter potentially much smaller than the reticulation number.

5. An FPT algorithm in the level of the network for computing the softwired parsimony score of a network. In the first part of this section we describe a polynomial-time dynamic programming (DP) algorithm that works on rooted trees and computes a slight generalization of the softwired parsimony score. We then show how this can be used as a subroutine in computing the softwired parsimony score of networks such that the running time is FPT in the level of the network.

5.1. A DP algorithm for (not necessarily phylogenetic) rooted trees with weights . Let $\mathcal{P} = \{1, \ldots, p\}$ be the set of character states. Let T be a rooted tree and $L(T)$ be the set of leaves of T. T is not necessarily a phylogenetic tree, because only a subset $L \subseteq L(T)$ needs to be labelled, and T is allowed to have nodes with indegree and outdegree both equal to 1. (Later on, we will see that this allows us to model switchings.) For a node $v \in V(T)$, let T_v be the subtree of T rooted at v.

We are given a p-state character $\alpha : L \to \mathcal{P}$. Additionally, we are given a function $w : (V(T) \times \mathcal{P}) \to \mathbb{N}$, where $\mathbb{N} = \{0, 1, \ldots\}$.

Consider the following definition, where c_τ is the *change* function described in the preliminaries and the minimum ranges over all extensions τ of α to $V(T)$:

$$PS_{sw}(T, \alpha, w) = \min_{\tau} \left(\left(\sum_{e \in E(T)} c_\tau(e) \right) + \left(\sum_{v \in V(T)} w(v, \tau(v)) \right) \right). \tag{5.1}$$

We can think of this as being the parsimony score with an optional added "weighting" that to varying degrees "penalizes" nodes when they are allocated a certain character state. This weighting $w(v, s)$ will be used in the next section to model the contribution to the optimum parsimony score of the subnetworks of N rooted at v when v is forced to be labelled with character state s.

To compute $PS_{sw}(T, \alpha, w)$ we introduce the value $PS_{sw}(T, \alpha, w, s)$, with $s \in \mathcal{P}$, where we add the restriction that the root of T must be labelled with character state s. Clearly,

$$PS_{sw}(T, \alpha, w) = \min_{s \in \mathcal{P}} PS_{sw}(T, \alpha, w, s). \tag{5.2}$$

We denote by $PS_{sw}(T, \alpha, w, \cdot)$ the vector $(PS_{sw}(T, \alpha, w, 1), \ldots, PS_{sw}(T, \alpha, w, p))$. This vector is computed as described in Algorithm 1, where $\delta(s, s') = 0$ if $s = s'$ and 1 otherwise, and $C(v)$ is the set of children of a nonleaf node v. Note that the

optimal τ can be constructed by backtracking, if necessary.

Algorithm 1: Compute $PS_{sw}(T, \alpha, w, \cdot)$

1 **for** *each node* v *of* $V(T)$ *considered in postorder* **do**
2 **if** v *is a leaf* **then**
3 **if** $v \in L$ **then**
4 $PS_{sw}(T_v, \alpha, w, s) = w(v, s)$ if $s = \alpha(v)$ and ∞ otherwise;
5 **else**
6 $PS_{sw}(T_v, \alpha, w, s) = w(v, s)$ for each $s \in \mathcal{P}$;
7 **else**
8 $PS_{sw}(T_v, \alpha, w, s) = w(v, s) + \sum_{v' \in C(v)} \left(\min_{s' \in \mathcal{P}} \left(PS_{sw}(T_{v'}, \alpha, w, s') + \delta(s, s') \right) \right)$
 $\forall s \in \mathcal{P}$;
9 **return** $PS_{sw}(T, \alpha, w, \cdot)$; `//note that` $T = T_{root(T)}$

The running time of Algorithm 1 is $O(p^2|V(T)|)$.

LEMMA 5.1. *Algorithm* 1 *correctly computes* $PS_{sw}(T_v, \alpha, w, \cdot)$ *for every* $v \in V(T)$. *In particular, it correctly computes* $PS_{sw}(T, \alpha, w, \cdot)$.

Sketch of proof. We sketch only the proof of the correctness for line 8. This follows from the fact that if the state of a node v is fixed as s, then the only local decisions that have to be made to optimize $PS_{sw}(T_v, \alpha, w, s)$ are to choose the character state s' for each child v' of v. A change is incurred whenever $s' \neq s$. Once s' has been chosen, we are free to (and therefore should) use optimal subsolutions corresponding to the case when the root of subtree $T_{v'}$ has state s', i.e., $PS_{sw}(T_{v'}, \alpha, w, s')$. We omit details. ∎

The following lemma shows that Algorithm 1 can be used to compute the parsimony score of a phylogenetic tree.

LEMMA 5.2. *Consider a rooted phylogenetic tree* T *on* X *and a* p*-state character* α *on* X. *Then, if* $w(v, s) = 0$ *for all* $v \in V(T)$ *and* $s \in \mathcal{P}$, *then* $PS_{sw}(T, \alpha, w) = PS_{sw}(T, \alpha)$.

Proof. This follows by combining Lemma 5.1 with (5.2) and (5.1). In particular, in (5.1) the right-hand side of the expression degenerates to the familiar parsimony definition, because w is 0 everywhere. ∎

5.2. Extending the DP algorithm to networks. Let N be a level-k network on X. We say that a biconnected component is *trivial* if it consists of a single edge (a cut-edge). Thanks to the following results, we can envisage N as comprising nontrivial biconnected components, each with reticulation number at most k, arranged in a tree-like backbone.

LEMMA 5.3. *Let* N *be a rooted phylogenetic network on* X *and* B *be a biconnected component of* N. *Then* B *contains exactly one node* r_B *without ancestors in* B.

Proof. Suppose there exist two roots in B, r_1 and r_2. In a rooted network N there always exists a directed path from the root of N to each node in N. Hence there exists a node v in N such that there is a simple directed path from v to r_1 and a simple directed path from v to r_2. (Note that we do not exclude the possibility that $v \in \{r_1, r_2\}$.) By merging these two paths we see that there is an undirected simple path P between r_1 and r_2 such that for at least one of r_1 and r_2 the edge of P incident to it is oriented towards it. We want to argue that all nodes and edges of P are also in B, which will contradict the assumption that r_1 and r_2 are both roots of B. In fact, it holds that if any two nodes u and v in B have a simple undirected path P'

between them, all nodes and edges of P' are also in B. If this were not true, then P' would contain some node not in B, and this in turn would mean that, in the journey from u to v, P' would have to pass through some cut node twice, contradicting its simplicity. Hence all nodes of P' are in B, and by maximality all of the edges of P' are too. □

LEMMA 5.4. *Let N be a rooted phylogenetic network on X. Then, if r is a reticulation, all incoming edges of r are in the same biconnected component of N.*

Proof. The lemma can be proven by applying an argument similar to that used in the proof of Lemma 5.3. We therefore omit the proof. □

We define the switchings of a biconnected component B of N analogous to the definition of the switchings of a network, i.e., a *switching* of a biconnected component B is a rooted tree S_B that can be obtained from B by deleting all but one of the incoming edges of each reticulation. We say that we *apply* switching S_B to N when deleting in N all edges of B not in S_B. The next result is a consequence of Lemma 5.4.

LEMMA 5.5. *Let N be a rooted phylogenetic network on X and S be a switching of N. Then, S can be obtained from N by, for each biconnected component B of N, first choosing a switching S_B and then applying it to N.*

COROLLARY 5.6. *Let N be a rooted phylogenetic network on X, S be a switching of N, and B be a biconnected component of N. Let S_B be the switching of B induced by S and S'_B be a different switching of B. Let S' be the graph obtained from N by applying to N all switchings of all biconnected components induced by S except S_B, and then finally applying the switching S'_B. Then S' is a switching of N.*

We are now ready to describe our algorithm for computing the softwired parsimony score of a phylogenetic network N and p-state character α. Note that each cut-edge (u, v) is seen as a biconnected component with root u and only one switching.

Algorithm 2: Compute $PS_{sw}(N, \alpha)$

```
1 for each node v of N and state s in P do w(v, s) ← 0;
2 for each node r of N that is a root of at least one biconnected component in
  postorder do
3     for each biconnected component B_r rooted at r do
4         for each switching S_r of B_r do
5             compute PS_sw(S_r, α, w, ·) using Algorithm 1;
6         for each s in P do
7             PS_sw(B_r, α, w, s) = min_{S_r ∈ B_r} PS_sw(S_r, α, w, s);
8             w(r, s) ← w(r, s) + PS_sw(B_r, α, w, s);
9 return min_{s∈P} w(root(N), s);
```

THEOREM 5.7. *Computing the softwired parsimony score of a rooted phylogenetic network N and a p-state character, for any $p \in \mathbb{N}$, is FPT if the parameter is the level of the network.*

Proof. In Lemma 5.3, we proved that each biconnected component B of N contains only one root r_B. We denote by $BT(N)$ the graph obtained as follows: we create a node v_r in $BT(N)$ for each node r of N that is the root of at least one (trivial or nontrivial) biconnected component of N, and we create an edge $(v_r, v_{r'})$ in $BT(N)$ if r and r' are contained in the same biconnected component, r is the root of this biconnected component, and $r' \neq r$. It is easy to see that $BT(N)$ is connected.

Moreover, it cannot contain any reticulation, because of Lemma 5.4. Thus $BT(N)$ is a tree. In the following we shall prove that $PS_{sw}(N,\alpha) = \min_{s\in\mathcal{P}} w(root(N),s)$ with w the weight function computed by Algorithm 2.

Denote by $PS_{sw}(N,\alpha,s)$ the minimum parsimony score for N and α, with the restriction that the root of N must be labelled with character state s. Let N_r be the subnetwork comprising all biconnected components whose roots can be reached by directed paths from r. We will prove that $PS_{sw}(N_r,\alpha,s) = w(r,s)$ for any node r in $V(N)$ associated with a node v_r in $BT(N)$. We prove this equality by induction on the height of v_r, which is defined as the length of a longest path from v_r to a leaf of $BT(N)$.

We begin by proving that the equality is true when the height of v_r is 0. Suppose that r is the root of J different biconnected components, and let B_j be one of these. Then we have that

$$PS_{sw}(B_j,\alpha,w,s) = \min_{S_j\in\mathcal{B}_j} PS_{sw}(S_j,\alpha,w,s) = \min_{S_j\in\mathcal{B}_j} PS_{sw}(S_j,\alpha,s) = PS_{sw}(B_j,\alpha,s),$$

where the second equality holds because of Lemma 5.2, since $w(v,s)$ is equal to zero for all nodes of B_j and s in $\mathcal{P}$. Then, because of Lemma 5.4, we have that

$$w(r,s) = \sum_{B_j\in\mathcal{B}_r} PS_{sw}(B_j,\alpha,w,s) = \sum_{B_j\in\mathcal{B}_r} PS_{sw}(B_j,\alpha,s) = PS_{sw}(N_r,\alpha,s),$$

where $\mathcal{B}_r$ is the set of biconnected components rooted at r.

Suppose now that $w(r,s) = PS_{sw}(N_r,\alpha,s)$ is true for all nodes v_r of $BT(N)$ with height at most h. We want to prove that this holds also for nodes with height $h+1$. Let v_r be such a node, r be the associated node in N, and B_j be a biconnected component rooted at r. Let $N_r(B_j)$ denote the subnetwork of N_r obtained by deleting all vertices and edges that are not reachable from any node in B_j other than its root r. Then, $PS_{sw}(B_j,\alpha,w,s) = PS_{sw}(N_r(B_j),\alpha,s)$, because for each child $v_{r'}$ of v_r in $BT(N)$ with corresponding vertex r' of N it holds, by the induction hypothesis, that $w(r',s) = PS_{sw}(N_{r'},\alpha,s)$. The intuitive idea is that, once a subnetwork has been processed, its influence in the biconnected component above it is expressed using the w function. The claim now follows since $PS_{sw}(N_r,\alpha,s) = \sum_{B_j\in\mathcal{B}_r} PS_{sw}(N_r(B_j),\alpha,s) = \sum_{B_j\in\mathcal{B}_r} PS_{sw}(B_j,\alpha,w,s) = w(r,s)$.

We still need to prove the running time. Algorithm 1 has a running time of $O(p^2|V(T)|)$. Moreover, for each biconnected component we call Algorithm 1 for at most 2^k trees with at most $|V(N)|$ nodes. Moreover, by Lemma 5.4, we have that the number of biconnected components of N is at most $|E(N)|$. Then we have an overall complexity of $O(2^k p^2|V(N)|\cdot|E(N)|)$. This concludes the proof. ∎

6. Maximum parsimony in practice: Integer linear programming. We propose the following integer linear programming (ILP) formulation for computing the hardwired parsimony score of a phylogenetic network, with node set V and edge set E, as well as a p-state character α. All variables are binary. Variable $x_{v,s}$ indicates whether or not node v has character state s, and variable c_e indicates whether there is a change on edge e or not. For a leaf v, parameter $\alpha(v)$ is the given character state at v. Let $\mathcal{P} = \{1,\dots,p\}$.

$$\begin{aligned} &\min \sum_{e\in E} c_e \\ &\text{subject to } \sum_{s\in\mathcal{P}} x_{v,s} = 1 \qquad\qquad \text{for all } v\in V, \end{aligned}$$

$$
\begin{aligned}
& c_e \geq x_{u,s} - x_{v,s} && \text{for all } e = (u,v) \in E, s \in \mathcal{P}, \\
& c_e \geq x_{v,s} - x_{u,s} && \text{for all } e = (u,v) \in E, s \in \mathcal{P}, \\
& x_{v,\alpha(v)} = 1 && \text{for each leaf } v, \\
& c_e \in \{0,1\} && \text{for all } e \in E, \\
& x_{v,s} \in \{0,1\} && \text{for all } v \in V, s \in \mathcal{P}.
\end{aligned}
$$

To see the correctness of the formulation, first observe that the first constraint ensures that each node is assigned exactly one character state. Now consider an edge $e = (u, v)$ and suppose that u and v are assigned different states s and s'. Then, $x_{u,s} \neq x_{v,s}$ (and $x_{u,s'} \neq x_{v,s'}$), and hence the second and third constraints ensure that $c_e = 1$.

For the softwired parsimony score, we extend the ILP formulation as follows. In addition to the variables above, there is a binary variable y_e indicating whether edge e is switched "on" or "off." A change on edge e is counted only if it is switched on. For each reticulation, exactly one incoming edge is switched on.

$$
\begin{aligned}
\min & \sum_{e \in E} c_e \\
\text{subject to } & \sum_{s \in \mathcal{P}} x_{v,s} = 1 && \text{for all } v \in V, \\
& c_e \geq x_{u,s} - x_{v,s} - (1 - y_e) && \text{for all } e = (u,v) \in E, s \in \mathcal{P}, \\
& c_e \geq x_{v,s} - x_{u,s} - (1 - y_e) && \text{for all } e = (u,v) \in E, s \in \mathcal{P}, \\
& \sum_{v:(v,r) \in E} y_{(v,r)} = 1 && \text{for each reticulation } r, \\
& y_e = 1 && \text{for each tree edge } e, \\
& x_{v,\alpha(v)} = 1 && \text{for each leaf } v, \\
& c_e, y_e \in \{0,1\} && \text{for all } e \in E, \\
& x_{v,s} \in \{0,1\} && \text{for all } v \in V, s \in \mathcal{P}.
\end{aligned}
$$

It follows from Lemma 4.5 that the optimum value of this ILP is equal to the softwired parsimony score of the given network and character.

Note that the parsimony score of an alignment can be computed by solving the above ILP formulation for each column (character) separately or combining them in a single ILP. Gaps in the alignment can be accommodated in the formulation by demanding that $x_{v,\alpha(v)} = 1$ only for leaves v for which $\alpha(v)$ is not a gap.

We have implemented both ILP formulations and made the resulting user-friendly software publicly available [13]. Experimental results with CPLEX 12.5 on a 2GHz laptop are in Table 1. Networks were simulated using Dendroscope [12], and character-states were assigned uniformly at random.

For practical applications, parsimony scores have to be computed quickly, since this computation needs to be repeated many times, for example when searching for a network with the smallest parsimony score. Apparent from Table 1 is that parsimony scores can be computed very quickly using ILP for networks with up to 100–150 taxa and up to 50 reticulations. Moreover, parsimony scores can even be computed quickly for much larger networks in the case of binary and ternary characters. This is of interest, because, in practice, many columns of an alignment might contain only two or three different symbols. Despite the theoretical differences in tractability of hardwired and softwired parsimony scores, their computation times using ILP do not differ much in these experiments.

TABLE 1

Time needed to compute hardwired and softwired parsimony scores in six test runs. For each run, an average is taken over 10 *simulated networks with randomly assigned character states. The results are given for a single character.*

	$\lvert\mathcal{X}\rvert$	Avg. number of retic.	Average computation time (s)					
			Hardwired PS			Softwired PS		
			2-state	3-state	4-state	2-state	3-state	4-state
Run 1	50	17.0	0.0	0.0	0.1	0.1	0.1	0.3
Run 2	100	37.0	0.0	0.0	0.2	0.0	0.1	0.6
Run 3	150	54.1	0.0	0.1	0.6	0.1	0.2	0.8
Run 4	200	72.8	0.0	0.1	1.1	0.1	0.4	1.4
Run 5	250	91.3	0.0	0.1	3.5	0.1	0.4	2.2
Run 6	300	112.6	0.0	0.2	5.2	0.1	0.6	3.7

TABLE 2

Summary of the complexity of computing hardwired and softwired parsimony scores of phylogenetic networks.

	Hardwired	Softwired
Complexity	In P for $p = 2$ NP-hard for $p \geq 3$	NP-hard for $p \geq 2$
Approximation	$\frac{12}{11}$-approx. for $p = 3$ 1.3438-approx. for $p \geq 4$	no $\lvert X\rvert^{1-\epsilon}$-approx. for any $\epsilon > 0$ unless P = NP
Parameterized by PS	FPT	NP-hard to decide if PS=1
Parameterized by level	N/A	FPT

7. Conclusions and open problems. We have clarified the distinction between two possible definitions of the parsimony score of a phylogenetic network, which we call the "softwired" and the "hardwired" parsimony scores. We have shown that computing the hardwired parsimony score is, in various ways, more tractable than computing the softwired score; see Table 2. We have also shown that the intractability results still hold under several topological restrictions. A stimulating open question is to determine the (in)approximability and fixed-parameter tractability of computing the softwired parsimony score of a character on a rooted network that is simultaneously binary and tree-child: this might be considerably more tractable than other versions of the problem. From a practical point of view, we have shown that both the hardwired and the softwired parsimony scores can be computed efficiently using ILP. It will be interesting to explore—in the spirit of studies such as those conducted in [16, 17]—the extension of this work to the notoriously intractable "big parsimony" problem.

Appendix A. Transforming degree-2 nodes (from Corollary 3.3). Although the hardwired parsimony score naturally extends to them, degree-2 nodes are not formally part of our phylogenetic network model. Fortunately, degree-2 nodes can simply be suppressed without altering the hardwired parsimony score. Unfortunately this may in turn create multi-edges which are likewise excluded from our definition. To deal with this, a multi-edge with multiplicity $t \geq 2$ between two nodes u and v can be encoded within the degree restrictions of a phylogenetic network by using a specific gadget. Namely, group the edges into $t' = \lfloor t/2 \rfloor$ pairs, and for each pair P_i ($1 \leq i \leq t'$) (i) delete the two edges concerned, (ii) add two new nodes x_i, y_i, and (iii) add the edges $(u, x_i), (u, y_i), (x_i, y_i), (x_i, v), (y_i, v)$. (If t is odd, the remaining edge can simply remain intact.) Again, this does not alter the hardwired parsimony score.

In fact, both transformations also leave the cut properties of the graph unchanged, which is important for the proof of Corollary 3.3.

Appendix B. Proof of Lemma 4.5.

LEMMA 4.5. *Consider a rooted phylogenetic network N on X and a p-state character α on X. Then,*

$$PS_{\mathcal{S}}(N,\alpha) = PS_{\mathrm{sw}}(N,\alpha).$$

Proof. Let S be a switching of N and τ be an extension of α to $V(S)$—or equivalently to $V(N)$—such that $\sum_{e\in E(S)} c_\tau(e) = PS_{\mathcal{S}}(N,\alpha)$. Let T be the tree obtained from S by deleting indegree-0 outdegree-1 nodes, deleting unlabelled outdegree-0 nodes, and suppressing indegree-1 outdegree-1 nodes, and let τ' be the restriction of τ to the nodes of S still present in T. By construction we have that since S is a switching of N on X, and T has been obtained from S as described above, $T \in \mathcal{T}(N)$. Moreover, since τ is an extension of α to $V(S)$, we have that τ' is an extension of α to $V(T)$. Finally, it is easy to see that $PS_{\mathcal{S}}(N,\alpha) = \sum_{e\in E(S)} c_\tau(e) \geq \sum_{e\in E(T)} c_{\tau'}(e) \geq PS_{\mathrm{sw}}(N,\alpha)$, since suppressing nodes (and consequently edges) cannot increase the sum of changes on the remaining edges.

Now, let T be a tree of $\mathcal{T}(N)$ and τ be an extension of α to $V(T)$ such that $\sum_{e\in E(T)} c_\tau(e) = PS_{\mathrm{sw}}(N,\alpha)$. Moreover, let S be a switching corresponding to T, i.e., such that T can be obtained from S by deleting indegree-0 outdegree-1 nodes, deleting unlabelled outdegree-0 nodes, and suppressing indegree-1 outdegree-1 nodes. (We know that such a switching exists, because $T \in \mathcal{T}(N)$.) Now, let $\tau' : V(S) \to \{1,\dots,p,?\}$ such that $\tau'(u) = \tau(u)$ if $u \in V(T)$ (i.e., u is the image in N of a node of T) and $\tau'(u) = \{?\}$ otherwise. A value in $\{1,\dots,p\}$ is associated with all nodes u of S having $\tau'(u) = \{?\}$ in the following way: We start by setting $\tau'(root(S))$ to $\tau(root(T))$, and then we traverse S in preorder, setting $\tau'(u)$ to $\tau'(u_p)$ for all nodes u having $\tau'(u) = \{?\}$, where u_p is the parent node of u.

First note that the root of T corresponds to the node ρ of S that is closest to the root with the following property: ρ has outdegree 2 or higher, and nodes not labelled ? can be reached from at least two children of ρ.

Then we have that all edges of S not reachable from ρ cost 0, since for all these edges (u,v) we have $\tau'(u) = \tau'(v) = \tau'(root(T))$. Now, let $e = (u,v)$ be an edge of T. In S this edge will often correspond to a set of edges, denoted by $E_S(e)$; see Figure 8. Now, note that the value of $\tau'(\cdot)$ is equal to $\tau(u)$ for all descendants of u in S that cannot be reached via v. Then it is easy to see that the cost of all edges in $E_S(e)$ equals $c_{\tau'}(w,v) = c_\tau(u,v) = c_\tau(e)$, where w is the parent node of v in S.

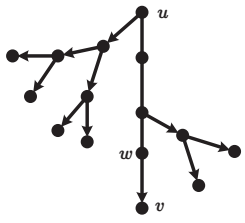

FIG. 8. *An example of an edge (u,v) of T that has been mapped to several edges in the switching underlying T, used in the proof of Lemma 4.5.*

Since this holds for all edges of T, and τ' is clearly an extension of α to $V(S)$,

we have that $PS_{\mathrm{sw}}(N,\alpha) = \sum_{e\in E(T)} c_\tau(e) = \sum_{e\in E(S)} c_{\tau'}(e) \geq PS_{\mathcal{S}}(N,\alpha)$. This concludes the proof. □

Acknowledgment. We thank Mike Steel for useful discussions on the topic of this paper.

REFERENCES

[1] N. Alon, B. Chor, F. Pardi, and A. Rapoport, *Approximate maximum parsimony and ancestral maximum likelihood*, IEEE/ACM Trans. Comput. Biol. Bioinformatics, 7 (2010), pp. 183–187.
[2] M.L. Arnold, *Natural Hybridization and Evolution*, Oxford University Press, New York, 1996.
[3] M. Baroni, C. Semple, and M. Steel, *Hybrids in real time*, Syst. Biol., 55 (2006), pp. 46–56.
[4] J.P. Bogart, *Genetics and Systematics of Hybrid Species*, Science Publishers, Enfield, NH, 2003, pp. 109–134.
[5] E. Dahlhaus, D.S. Johnson, C.H. Papadimitriou, P.D. Seymour, and M. Yannakakis, *The complexity of multiterminal cuts*, SIAM J. Comput., 23 (1994), pp. 864–894.
[6] A.W.F. Edwards and L.L. Cavalli-Sforza, *Reconstruction of evolutionary trees*, in Systematics Association Publication, no. 6, London, 1964, pp. 64–76.
[7] J. Felsenstein, *Cases in which parsimony or compatibility will be positively misleading.*, Syst. Zool., 27 (1978), pp. 401–410.
[8] W. Fitch, *Toward defining the course of evolution: Minimum change for a specific tree topology.*, Syst. Zool., 20 (1971), pp. 406–416.
[9] J. Flum and M. Grohe, *Parameterized Complexity Theory*, Springer, Berlin, 2006.
[10] L.R. Foulds and R.L. Graham, *The Steiner problem in phylogeny is NP-complete.*, Adv. in Appl. Math., 3 (1982), pp. 43–49.
[11] D.H. Huson, R. Rupp, and C. Scornavacca, *Phylogenetic Networks: Concepts, Algorithms and Applications*, Cambridge University Press, Cambridge, UK, 2011.
[12] D.H. Huson and C. Scornavacca, *Dendroscope* 3: *A program for computing and drawing rooted phylogenetic trees and networks*, Syst. Biol., 61 (2012), pp. 1061–1067.
[13] L.J.J. van Iersel, M. Fischer, S.M. Kelk, and C. Scornavacca, *MPNet: Maximum Parsimony on Networks*, http://homepages.cwi.nl/~iersel/MPNet/ (2013).
[14] L.J.J. van Iersel and S.M. Kelk, *When two trees go to war*, J. Theoret. Biol., 269 (2011), pp. 245–255.
[15] L.J.J. van Iersel, C. Semple, and M. Steel, *Locating a tree in a phylogenetic network*, Inform. Process. Lett., 110 (2010), pp. 1037–1043.
[16] G. Jin, L. Nakhleh, S. Snir, and T. Tuller, *Inferring phylogenetic networks by the maximum parsimony criterion: A case study*, Mol. Biol. Evol., 24 (2007), pp. 324–337.
[17] G. Jin, L. Nakhleh, S. Snir, and T. Tuller, *Parsimony score of phylogenetic networks: Hardness results and a linear-time heuristic*, IEEE/ACM Trans. Comput. Biol. Bioinformatics, 6 (2009), pp. 495–505.
[18] I.A. Kanj, L. Nakleh, C. Than, and G. Xia, *Seeing the trees and their branches in the network is hard*, Theoret. Comput. Sci., 401 (2008), pp. 153–164.
[19] L. Kannan and W.C. Wheeler, *Maximum parsimony on phylogenetic networks*, Algorithms Mol. Biol., 7 (2012), 9.
[20] D.R. Karger, P. Klein, C. Stein, M. Thorup, and N.E. Young, *Rounding algorithms for a geometric embedding of minimum multiway cut*, in Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing (STOC '99), 1999, pp. 668–678.
[21] S. Kelk and C. Scornavacca, *Constructing minimal phylogenetic networks from softwired clusters is fixed parameter tractable*, Algorithmica, 68 (2014), pp. 886–915.
[22] S.M. Kelk, C. Scornavacca, and L.J.J. van Iersel, *On the elusiveness of clusters*, IEEE/ACM Trans. Comput. Biol. Bioinformatics, 9 (2012), pp. 517–534.
[23] E.V. Koonin, K.S. Makarova, and L. Aravind, *Horizontal gene transfer in prokaryotes: Quantification and classification*, Annu. Rev. Microbiol., 55 (2001), pp. 709–742.
[24] D.R. Maddison and K.S. Schulz, eds., *The Tree of Life Web Project,* http://www.tolweb.org (2007).
[25] L. McDaniel, E. Young, J. Delaney, F. Ruhnau, K. Ritchie, and J. Paul, *High frequency of horizontal gene transfer in the oceans*, Science, 330 (2010), p. 50.
[26] D.A. Morrison, *Introduction to Phylogenetic Networks*, RJR Productions, Uppsala, Sweden, 2011.
[27] L. Nakhleh, *Evolutionary phylogenetic networks: Models and issues*, in The Problem Solving

Handbook for Computational Biology and Bioinformatics, Springer, Berlin, 2009, pp. 125–158.

[28] C.T. Nguyen, N.B. Nguyen, W.K. Sung, and L. Zhang, *Reconstructing recombination network from sequence data: The small parsimony problem*, IEEE/ACM Trans. Comput. Biol. Bioinformatics, 4 (2007), pp. 394–402.

[29] R. Niedermeier, *Invitation to Fixed Parameter Algorithms*, Oxford Lecture Ser. Math. Appl. 31, Oxford University Press, Oxford, UK, 2006.

[30] D. Sankoff and P. Rousseau, *Locating the vertices of a Steiner tree in an arbitrary metric space*, Math. Programming, 9 (1975), pp. 240–246.

[31] C. Semple and M. Steel, *Phylogenetics*, Oxford University Press, Oxford, UK, 2003.

[32] M. Xiao, *Simple and improved parameterized algorithms for multiterminal cuts*, Theory Comput. Syst., 46 (2010), pp. 723–736.