Computing Galled Networks from Real Data

Daniel Huson, Regula Rupp, Vincent Berry, Philippe Gambette, Christophe Paul

BIOINFORMATICS

Vol. 25 ISMB 2009, pages i85–i93
doi:10.1093/bioinformatics/btp217

Computing galled networks from real data

Daniel H. Huson[1,*], Regula Rupp[1], Vincent Berry[2], Philippe Gambette[2] and Christophe Paul[2]

[1]Center for Bioinformatics ZBIT, Tübingen University, Sand 14, 72076 Tübingen, Germany and [2]Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM, UMR 5506, CNRS), Université Montpellier II, 161, rue Ada, 34392 Montpellier Cedex 5, France

ABSTRACT

Motivation: Developing methods for computing phylogenetic networks from biological data is an important problem posed by molecular evolution and much work is currently being undertaken in this area. Although promising approaches exist, there are no tools available that biologists could easily and routinely use to compute rooted phylogenetic networks on real datasets containing tens or hundreds of taxa. Biologists are interested in clades, i.e. groups of monophyletic taxa, and these are usually represented by clusters in a rooted phylogenetic tree. The problem of computing an optimal rooted phylogenetic network from a set of clusters, is hard, in general. Indeed, even the problem of just determining whether a given network contains a given cluster is hard. Hence, some researchers have focused on topologically restricted classes of networks, such as galled trees and level-k networks, that are more tractable, but have the practical draw-back that a given set of clusters will usually not possess such a representation.

Results: In this article, we argue that galled networks (a generalization of galled trees) provide a good trade-off between level of generality and tractability. Any set of clusters can be represented by some galled network and the question whether a cluster is contained in such a network is easy to solve. Although the computation of an optimal galled network involves successively solving instances of two different NP-complete problems, in practice our algorithm solves this problem exactly on large datasets containing hundreds of taxa and many reticulations in seconds, as illustrated by a dataset containing 279 prokaryotes.

Availability: We provide a fast, robust and easy-to-use implementation of this work in version 2.0 of our tree-handling software Dendroscope, freely available from http://www.dendroscope.org.

Contact: huson@informatik.uni-tuebingen.de

1 INTRODUCTION

Developing appropriate methods for computing phylogenetic networks from biological data has been described as one of the five most important challenges posed to mathematics by biology (Cohen, 2004) and there is a lot of recent work focused on this topic (Doolittle and Bapteste, 2007): One area of research is recombination networks, which have been studied in the context of population genetics for many years, e.g. (Hein, 1993; Lyngsø *et al*., 2005; Song and Hein, 2005), and have recently received more attention in the guise of galled trees and related concepts, e.g. (Gusfield and Bansal, 2005; Gusfield *et al*., 2003; Huson and Kloepper, 2005). Here, the task is to compute a rooted phylogenetic network from a set of binary sequences. A second area is hybridization networks (Bordewich *et al*., 2007; Huson *et al*., 2005; Linder and Rieseberg, 2004), where the goal is to compute a rooted phylogenetic network from a set of incongruent trees. A third area is HGT networks, where the goal usually is to explain the discrepancies between gene trees (or gene content) and a species tree, e.g (Hallett and Lagergren, 2001; Mirkin *et al*., 2003). Another area of research focuses on the computation of rooted networks from triplets (van Iersel *et al*., 2008).

One type of *unrooted* phylogenetic network, called a *split network* (Bandelt and Dress, 1992), is widely used as a tool for displaying incompatible datasets. Its popularity is probably due to the existence of a user-friendly program (Huson and Bryant, 2006) that contains a number of methods, such as Neighbor-net (Bryant and Moulton, 2002) or the Z-closure super network method (Huson *et al*., 2004), for computing such networks.

The current state of the art for *rooted* phylogenetic networks, on the other hand, can be characterized as follows: there are many promising directions to follow and some rudimentary software implementations (Huber *et al*., 2007; Than *et al*., 2008; van Iersel and Kelk, 2008). However, there is no tool currently available that biologists could easily and routinely use on real data. This is an unfortunate state of affairs, because biologists are more interested in rooted networks than in unrooted networks, just as they are more interested in rooted trees than in unrooted ones. (Unrooted networks are very different from rooted networks, so constructions for unrooted networks usually do not carry over to rooted networks.)

The aim of this article is to help fill this gap by providing a theoretically well-founded and practically well-implemented and easy-to-use method for computing rooted phylogenetic networks from a set of clusters, for example, extracted from a set of phylogenetic trees, or directly from sequences. Our approach is based on the concept of a galled network, which is a generalization of the concept of a galled tree. Continuing work started Huson and Kloepper (2007), here we make new contributions to a number of topics, including: complexity of the cluster containment problem, existence of a galled network, desirability of the decomposition property, role of incompatibility statements, complexity and solution of the (RMSC) problem, formulation, complexity and solution of the Minimum Attachment problem (MAP), user-friendly, robust and fast implementation of all algorithms, aimed at real data. We believe that galled networks provide a good trade-off between level of generality and ease of practical computability.

*To whom correspondence should be addressed.

As we will show in this article, one main advantage of galled networks is that the *Cluster Containment* problem is easy to solve for them. This is not true for general rooted phylogenetic networks (Kanj *et al*., 2008). In consequence, one can easily verify algorithmically that a galled network actually contains all the clusters that it claims to represent and a user of the network can easily locate a cluster of interest. Galled networks are particularly applicable if the dataset can basically be represented by a tree, to which some reticulations are attached. In such cases, the computed network explicitly provides a possible evolutionary scenario. In more complicated situations (such as in population studies of sexually reproducing organisms), the network should only be seen as an abstract representation of the given cluster set.

We provide an implementation of our method aimed at phylogenetic trees (as input) in the new version 2.0 of our Dendroscope software (Huson *et al*., 2007). We plan also to integrate the code into our SplitsTree software (Huson and Bryant, 2006), where it will be applicable to character sequences, too.

We demonstrate the performance of our algorithms using two different datasets. The first is a collection of six trees on plant species, the size of the trees ranging from 19 to 65 taxa, on which we compare the performance of our algorithms with results reported in Bordewich *et al*. (2007). The second dataset is a collection of nine trees, each on 279 prokaryotes (Auch *et al*., manuscript in preparation). These examples demonstrate that our methods can be applied to reasonably large datasets in acceptable runtimes (usually only seconds).

2 GALLED NETWORKS COMPARED WITH GALLED TREES, LEVEL K NETWORKS AND MORE GENERAL ROOTED PHYLOGENETIC NETWORKS

Let $\mathcal{X}$ be a set of taxa. A *rooted phylogenetic network* N on $\mathcal{X}$ consists of a connected, directed, acyclic graph with a single node of in-degree 0, called the *root*, together with a bijective labeling λ of $\mathcal{X}$ onto the set of all *leaves* of N, i.e. all nodes of out-degree 0. All nodes of in-degree ≤ 1 are called *tree nodes* and all others are called *reticulate nodes* or simply *reticulations*. Any edge leading to a reticulate node is called a *reticulate edge*, and all others are called *tree edges*.

A *cluster* C of $\mathcal{X}$ is any proper subset of $\mathcal{X}$, explicitly excluding both the empty set $\emptyset$ and the full set $\mathcal{X}$. Let $\mathcal{C}$ be a set of clusters on $\mathcal{X}$. We say that a rooted phylogenetic network N represents $\mathcal{C}$ in the *hardwired sense*, if there exists a one-to-one mapping between the set of clusters $\mathcal{C}$ and the set of tree edges in N such that for each cluster C the set of leaves reachable below the corresponding edge e is labeled precisely by C. For example, *cluster networks*, which we introduced in Huson and Rupp (2008), represent clusters in the hardwired sense, and can be computed efficiently.

We say that a rooted phylogenetic network N represents $\mathcal{C}$ in the *softwired sense*, or N is a *softwired network*, if for every cluster $C \in \mathcal{C}$ there exists some tree edge e in N and some *switching* of all reticulate nodes in N such that the set of leaves reachable below e is labeled precisely by C. By a *switching* of a reticulate node r, we mean a choice of exactly one ingoing edge of r that is considered 'turned on', whereas all other ingoing edges are considered 'turned off'. In the following, when we say that a cluster is represented by a tree edge in a network then we will always mean this in the softwired sense.

The basic idea behind the concept of a softwired network is that it describes the evolution of multiple genes or loci: the evolutionary history of any given gene or locus is described by a tree, embedded in the network and containing exactly one of the ingoing edges for each reticulate node of the network.

Let $\mathcal{C}$ be a set of clusters on $\mathcal{X}$. In practice, such a set of clusters is usually given as the set of all clusters present in a collection of rooted phylogenetic trees, or may come from an alignment of binary sequences. In a number of papers mentioned above, the authors invoke the parsimony principle to argue that the best network to compute to represent a set of clusters (in the softwired sense) is one that minimizes the number of reticulation nodes contained in the network, as the underlying evolutionary events, for example, speciation by hybridization, are considered 'expensive'. As we will show later, there may be cases where saving a reticulation results in two *completely unrelated* parts of the phylogeny being linked together, so we will slightly alter the aim of the minimization so as to address this problem.

The problem of computing a miminal rooted phylogenetic network N that represents a given set of clusters $\mathcal{C}$ in the softwired sense is known to be hard. In fact, even just the *cluster containment* problem of determining whether a given cluster C is represented by some given network N, is already NP-complete, see (Kanj *et al*., 2008).

Due to the computational difficulties associated with the general problem, researchers have studied rooted phylogenetic networks that fulfill certain topological constraints. In Gusfield and colleagues (2003) introduced the concept of a *galled tree*, based on a condition introduced in Ma *et al*. (1998) and Wang *et al*. (2001), in which each biconnected component (as defined below) of the network contains at most one reticulate node. Other authors have generalized this to *level k* networks (van Iersel *et al*., 2008), allowing at most k reticulate nodes in any biconnected component. A practical drawback of these definitions is that for a given set of clusters there might not exist such a network (for a fixed level k). So, any tool based on these concepts is liable to disappoint its users on some datasets.

In Huson and Kloepper (2007) we started developing an alternative generalization of galled trees called *galled networks*.

DEFINITION 1. *A rooted phylogenetic network N is called a* galled network*, if for every reticulation r in N and every pair of reticulation edges p and q with target node r there exists a* tree cycle*, that is, an undirected cycle in N that passes through p followed by q and otherwise contains only tree edges.*

The main idea behind this definition is that a galled network may be obtained by attaching one or more reticulations to a rooted phylogenetic 'backbone' tree, see Figure 1. A useful and easy consequence of this is that every reticulate node r in a galled network N is a *cut node* that separates the set of all nodes that are descendants of r from all other nodes in N (Huson and Kloepper, 2007). This corresponds to a biological scenario in which reticulate events (such hybridization or horizontal gene transfer events) are quite rare.

As already mentioned, one advantage of galled networks over other restricted classes of rooted networks is that for any input set of clusters there always exists a corresponding galled network, unlike

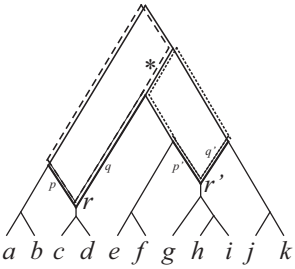


Fig. 1. A galled network containing two reticulations r and r'. The tree cycles associated with r and r' are emphasized using dashed and dotted lines, respectively. In contrast to a galled tree, in which all cycles are edge-disjoint, these two tree cycles share an edge, namely the one marked $*$.

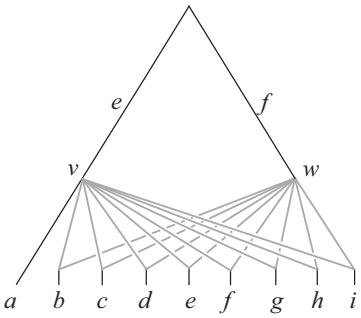


Fig. 2. Any set of clusters on $\mathcal{X}$ can be represented by a galled network using a construction that is illustrated here for $\mathcal{X}=\{a,\ldots,i\}$. For all leaves except the one labeled by the first taxon (in this example, a), the source of each leaf edge is a reticulate node that is connected to two parent nodes v and w. Any cluster C of $\mathcal{X}$ that contains the first taxon is represented by the tree edge e, whereas any cluster that does not contain the first taxon is represented by the tree edge f, in the softwired sense.

in the case of galled trees or level k networks (Fig. 2). Moreover, the cluster containment problem is easy to solve:

LEMMA 1. (Cluster containment for galled networks). The *cluster containment for galled networks* problem can be efficiently solved.

PROOF. We need to show how to determine whether a given tree edge $e=(v,w)$ in a galled network N can represent a given cluster C. Let L_e denote the set of all leaves below the node w that are reachable from w via a path of tree edges, and let R_e denote the set of all reticulate nodes below w that are reachable from w via a path of tree edges, followed by a single reticulate edge. The taxa that label the nodes in L_e are contained in every cluster represented by e. As every reticulate node r in a galled network N is a cut node, we have two possibilities. Either the set Y_r of taxa below r is contained in every cluster represented by e, which happens when all parent nodes of r are descendants of w. Or Y_r is an optional subset in the sense that a cluster represented by e must either contain all of Y_r or none of Y_r. Finally, e represents C if and only if there exists a set $R\subseteq R_e$ such that $C=L_e\cup\bigcup_{r\in R} Y_r$. ■

3 DECOMPOSITION OF PHYLOGENETIC NETWORKS

Let $\mathcal{C}$ be a set of clusters on $\mathcal{X}$. Two clusters $A,B\in\mathcal{C}$ are called *compatible*, if they either are disjoint, or one contains the other, and are called *incompatible*, otherwise. We call $\mathcal{C}$ *pairwise compatible*, if all pairs of clusters in $\mathcal{C}$ are compatible. A basic result of phylogenetics states that a set of clusters $\mathcal{C}$ on $\mathcal{X}$ can be represented by a rooted phylogenetic tree, if and only if $\mathcal{C}$ is pairwise compatible.

We define the *incompatibility graph* $\mathrm{IG}(\mathcal{C})=(V,E)$ for $\mathcal{C}$ as the graph with node set $V=\mathcal{C}$ and edge set E such that any two clusters $A,B\in\mathcal{C}$ are connected by an edge if and only if they are incompatible with each other.

Recall from basic graph theory that a *connected component* of an undirected graph G is defined as any maximal set of nodes U in G with the property that any two nodes in U are connected by a path in G. Similarly, a *biconnected component* is a subgraph of G induced by a maximal set of nodes U such that any two nodes $v,w\in U$ are connected by *two* different (undirected) paths that are node-disjoint (except at v and w).

Let N be a rooted phylogenetic network that represents a set of clusters $\mathcal{C}$. Because a cluster C may be represented by more than one edge in N, we define an *edge assignment* ϵ to be a mapping that chooses for each cluster $C\in\mathcal{C}$ a single tree edge $\epsilon(C)$ in N that represents it.

We say that N is a *decomposable representation* of $\mathcal{C}$, or simply that N is *decomposable*, if there exists an edge assignment ϵ such that:

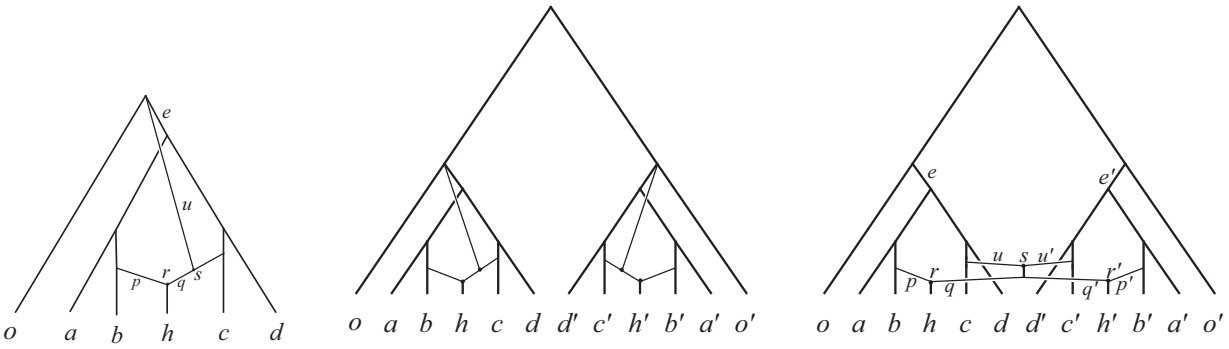
- for all pairs of clusters $A,B\in\mathcal{C}$ we have: the two edges $\epsilon(A)$ and $\epsilon(B)$ lie in the same biconnected component of N if and only if A and B lie in the same connected component of the incompatibility graph $IG(\mathcal{C})$.

By definition, cluster networks (Huson and Rupp, 2008) and split networks (Bandelt and Dress, 1992) are always decomposable. However, an example due to Yun Song shows that this property does not hold for more general rooted phylogenetic networks (Gusfield *et al*., 2007). As we illustrate in Figure 3, it is sometimes possible to save one reticulation by replacing two completely unlinked and unrelated configurations, each containing two reticulations, by one configuration containing only three reticulations. This construction does not reflect biology as it may link together totally unrelated and arbitrarily distant parts of a phylogeny, thus resulting in misleading networks, possibly bringing together nematodes and apes, for example.

To avoid this problem, but also for computational reasons, we propose to focus on decomposable representations. As a consequence, we may *decompose* our task as follows (Gusfield and Bansal, 2005; Huson *et al*., 2005). Suppose we are given a set of clusters $\mathcal{C}$ on $\mathcal{X}$ as input. We first compute the connected components of $\mathrm{IG}(\mathcal{C})$ [either naively in time $O(|\mathcal{C}|^2)$ by building the incompatibility graph, or directly, in subquadratic time (Charbit *et al*., 2008)]. Then, for each non-trivial connected component $\mathcal{C}'\subseteq\mathcal{C}$ of $\mathrm{IG}(\mathcal{C})$ we compute a rooted phylogenetic network N' for $\mathcal{C}'$. All these will then be fitted together to produce as output a final network N for $\mathcal{C}$.

We say that two taxa $x,y\in\mathcal{X}$ are *separated* if there exists a cluster $A\in\mathcal{C}$ with $|\{x,y\}\cap A|=1$. When considering such a subproblem $\mathcal{C}'$, we identify or collapse all taxa that are not separated by any cluster in $\mathcal{C}'$.

As a consequence, throughout the next sections of the article, we may assume that for a set of clusters $\mathcal{C}$ on $\mathcal{X}$ the two following properties hold.



(**a**) Phylogenetic network (**b**) Decomposable representation (**c**) Non-decomposable representation

Fig. 3. (**a**) A minimum rooted network N that represents the clusters $\mathcal{C} = \big\{\{a\},\{b\},\{c\},\{d\},\{o\},\{h\},\{a,b\},\{a,b,h\},\{b,h\},\{c,d\},\{c,d,h\},\{a,b,c,d,h\},\{a,b,c,d\}\big\}$ using two reticulate nodes r and s. Note the role of the reticulate edge labeled u is to switch off the taxon h so that the cluster $\{a,b,c,d\}$ can be represented by the tree edge e. (**b**) Two copies of N embedded into a decomposable network requiring four reticulations to represent all clusters in $\mathcal{C}$ and a second set $\mathcal{C}'$ of corresponding ones on $\{o',a',b',c',d',h'\}$. (**c**) This network also represents all described clusters, but uses only three reticulations. However, we clearly see that this improvement is gained at an undesirable price: decomposability is abandoned (all clusters in $\mathcal{C}$ are compatible with all clusters in $\mathcal{C}'$, yet they are all represented in the same biconnected component of the network) and so two completely unrelated parts of the phylogeny are linked together via reticulation edges.

(P1) The incompatibility graph IG($\mathcal{C}$) has only one connected component.

(P2) Every pair of taxa in $\mathcal{X}$ is separated.

In effect, our algorithm processes all connected components of the incompatibility graph sequentially.

4 DETERMINING A MINIMUM SET OF RETICULATIONS

As discussed above, one way to think of galled networks is that they are constructed by attaching a number of reticulations to a phylogenetic tree. Hence, to compute a galled network that uses a minimum number of reticulations, we will first determine a minimum set of reticulations [the Maximum Compatible Subset (MCS) problem, discussed in this section], which will then be attached optimally to the tree part of the network (the MAP, discussed in the next section).

PROBLEM 1. (MCS). *Let $\mathcal{C}$ be a set of clusters on $\mathcal{X}$. The* Maximum Compatible Subset *(MCS) problem is to remove a minimum set of taxa R from $\mathcal{X}$ so that the set of clusters $\mathcal{C}|_{\mathcal{X}\setminus R}$ induced by $\mathcal{X}\setminus R$ is compatible.*

For a general set of clusters, this is equivalent to the 'Maximum Compatible Tree' (MCT) problem, which is known to be NP-hard (Steel and Hamel, 1996). We refer to the MCS problem in which the instances are restricted to sets of clusters $\mathcal{C}$ on $\mathcal{X}$ for which properties (P1) and (P2) hold as the RMCS problem. We next state that the RMCS problem is hard:

LEMMA 2. (RMCS problem is hard). *Let $\mathcal{C}$ be a set of clusters on $\mathcal{X}$ with properties (P1) and (P2). Solving the MCS problem is NP-hard for such input.*

A proof is given in the Appendix A.

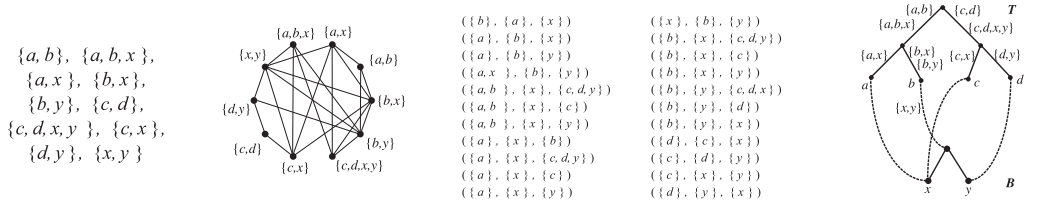
In Berry and Nicolas (2006), an FPT algorithm is presented for solving the MCT problem for a set of rooted phylogenetic trees on $\mathcal{X}$. This algorithm can be employed to solve the RMCS problem, too, simply by encoding each cluster C as an appropriate rooted phylogenetic tree.

In this article, we formulate a new algorithm for solving RMCS directly, which we call the *seed-growing algorithm*. This algorithm goes straight to the heart of the problem and thus performs very well in practice. For each pair of incompatible clusters $A,B\in\mathcal{C}$ we define an *incompatibility statement* consisting of the three terms $A\setminus B, A\cap B, B\setminus A$, and we use L to denote the list of all such statements for $\mathcal{C}$ (Fig. 4a–c). To solve the RMCS problem, we must find a minimum set of taxa $R\subset\mathcal{X}$ that *resolves* each incompatibility statement in L, that is, for each incompatibility statement in L at least one term is a subset of R .

The algorithm maintains a set S of candidate solutions, called *seeds*. Each seed $s\in S$ is labeled by the number rank(s) of incompatibility statements that it has been shown to resolve in succession, starting from the beginning of the list L. Initially, the three parts of the first incompatibility statement are chosen as seeds, and we set rank(s)$=1$ for each such seed s. The algorithm then proceeds by repeatedly choosing among all seeds of minimum size a seed s^* that maximizes rank(s^*). If rank(s^*)$=|L|$, then s^* is an optimal solution of the RMCS problem and the algorithm halts. Otherwise, if s^* resolves the (rank(s^*)$+1$)-th incompatibility statement X,Y,Z, we increment rank(s^*) by 1. Otherwise, we define three new seeds $s_1 = s^*\cup X$, $s_2 = s^*\cup Y$ and $s_3 = s^*\cup Z$, with rank(s_1)$=$rank(s_2)$=$rank(s_3)$=$rank(s^*)$+1$, add these to S and remove s^* from S.

Note that this algorithm is more than just a greedy heuristic, as it is guaranteed to find an optimal solution. Moreover, although the MCS problem has the flavor of a hitting set-, vertex cover- or edge cover problem, it is, in fact, a new problem.

LEMMA 3. (Performance of the seed-growing algorithm). *If a solution to the Maximum Compatible Subset problem has size k for a given set of incompatibility statements H, then the seed growing will find it by considering at most 3^{k+1} seeds, and the algorithm has a worst-case time complexity of $O(k|H|3^k)$.*



(a) Set of clusters $\mathcal{C}$ (b) Incompatibility graph (c) Incompatibility statements (d) Galled network for $\mathcal{C}$

Fig. 4. (**a**) A set of clusters $\mathcal{C}$ on $\mathcal{X}=\{a,b,c,x,y\}$, (**b**) the incompatibility graph $IG(\mathcal{C})$, (**c**) the corresponding list L of incompatibility statements solved by choosing $\{x,y\}$ as the solution of the MCS problem; and (**d**) a galled network N obtained by solving the MAP. Here, the top and bottom parts of the network are labeled T and B, respectively. The edges of the top part are labeled by the non-trivial clusters that they represent and the leaves are labeled by their taxa. Link edges are shown as dashed lines.

A proof is given in the Appendix A.

5 THE MAP

Let $\mathcal{C}$ be a set of clusters on $\mathcal{X}$ and $R\subsetneq\mathcal{X}$ a minimum subset of taxa such that the restriction of $\mathcal{C}$ to $\mathcal{X}\setminus R$, denoted by $\mathcal{C}|_{\mathcal{X}\setminus R}$, is compatible. Let T be the rooted phylogenetic tree on $\mathcal{X}\setminus R$ that represents $\mathcal{C}|_{\mathcal{X}\setminus R}$, and let $L(e)$ denote the cluster in $\mathcal{C}|_{\mathcal{X}\setminus R}$ represented by an edge e of T. For each tree edge e of T, let $\mathcal{C}(e)=\{C\in\mathcal{C}\,|\,C\setminus R=L(e)\}$ denote the set of all clusters in $\mathcal{C}$ that are mapped to $L(e)$ under the restriction of $\mathcal{C}$ to $\mathcal{X}\setminus R$, and let $R(e)=\{r\in R\cap C\,|\,C\in\mathcal{C}(e)\}$ denote the set of all reticulate taxa that map to the cluster $L(e)$. We will refer to T as the *top* part.

Let $\mathcal{C}|_R$ denote the restriction of $\mathcal{C}$ to the taxon set R and let $\hat{\mathcal{C}}|_R$ denote the set of all maximal clusters (by containment) in $\mathcal{C}|_R$. We now define a graph B associated with $\hat{\mathcal{C}}|_R$, as follows: each cluster $C\in\hat{\mathcal{C}}|_R$ is represented by a node $v(C)$ and each taxon $r\in R$ is represented by a node $v(r)$ and we place an edge from $v(C)$ to $v(r)$ for all taxa r contained in the cluster C. We will refer to B as the *bottom* part (Fig. 4d).

Problem 2. (Attachment problem). *The* Attachment *problem is to define a set of* link *edges from nodes in the top part T to nodes in the bottom part B such that the resulting graph is a galled network that represents the input set of clusters* $\mathcal{C}$.

More precisely, we aim at representing all clusters in $\mathcal{C}(e)$ by the edge e in T and all clusters in $\mathcal{C}|_R$ by the in-edges of the nodes of the form $v(C)$, with $C\in\hat{\mathcal{C}}|_R$. To ensure this, the *link set* must fulfill the following properties.

- (A1) For every edge e of T and every taxon $r\in R(e)$ there exists a link from some descendent node of e in T to either $v(r)$ or to some node of the form $v(C)$ in B, where $C\in\hat{\mathcal{C}}|_R$ contains r.
- (A2) For every node of the form $v(C)$ in B, with $C\in\hat{\mathcal{C}}|_R$, there exists exactly one link from some node in T to $v(C)$.
- (A3) For every edge e in T and $r\in R$ such that $\mathcal{C}(e)$ contains some $C\in\mathcal{C}$ that does not contain r, there exists a path from some node in T, that is not a descendant of e, to $v(r)$.

Property (A1) ensures that we can reach $v(r)$ from any edge e in T that has a cluster $C\in\mathcal{C}(e)$ that contains r. Property (A2) ensures that all nodes of the form $v(C)$ in B obtain in-degree 1. We do not allow an in-degree >1 to ensure that only nodes of the form $v(r)$ for $r\in R$ will be reticulate nodes. For example, in Figure. 4d, the node above x and y is only attached to the node labeled b, and not to the node labeled c or to the parent of the two nodes labeled c and d, etc. Finally, property (A3) ensures that the node $v(r)$ can be avoided from any edge e in T that must represent a cluster that does not contain r.

Our goal is to minimize the number of edges used to solve the following Attachment problem.

Problem 3. (MAT). *Find a collection of links that has all properties (A1)–(A3) and is of minimum size.*

In general, this is hard to solve.

Lemma 4. (The MAT Problem is hard). *The decision problem whether there exists a solution to the Attachment problem that uses at most k edges is NP-complete.*

A proof is given in the Appendix A.

The above reduction also proves that the MAT problem is W[2]-hard, remaining intractable even when parameterized by k, and so it is unlikely that an $O(f(k)\cdot poly(n))$ algorithm can be obtained. The instances of this problem that are of interest in practice are usually quite small and so a branch-and-bound approach is adequate. Alternatively, the problem can be posed as an ILP with binary variables that determine whether a possible link edge is used or not and inequalities that ensure that the properties (A1)–(A3) hold. The optimization goal is then to minimize the sum of the binary variables.

The following complication must also be taken into account. After solving the MAP for a set of clusters $\mathcal{C}$ on $\mathcal{X}$ and a given reticulation set R, assume there exists an edge e along the path from the root of the top part T to a node $v(r)$, with $r\in R$, that represents a cluster C that does not contain r. Assume further that e has the property that the 'lowest single ancestor' of $v(r)$ (Huson and Rupp, 2008) lies below e. Then we need to create an additional edge to connect $v(r)$ to a node above e to be able to turn off the taxon r in the representation of the cluster C.

6 IMPLEMENTATION IN DENDROSCOPE

As our goal is not only to solve the theoretical aspects of galled networks, but also to provide a robust and easy to use tool so that biologists can benefit immediately from these results, we have implemented all presented algorithms in a new version 2.0 of our program Dendroscope (Huson *et al*., 2007).

Originally designed as a tool for drawing phylogenetic trees, this new version of Dendroscope is geared toward analyzing multiple trees using both consensus trees and rooted networks. When asked

to produce a rooted network, the program provides the user with the choice of either computing a rooted 'cluster network', as recently introduced in (Huson and Rupp, 2008), or of computing an 'optimal galled network', as described in this article.

Input is a set of rooted trees on $\mathcal{X}$. The program proceeds by extracting the set of all clusters $\mathcal{C}$ from the input, or, alternatively, all clusters contained in at least a fixed percentage of the input trees. If the given phylogenetic trees are on overlapping, but non-identical taxon sets, our implementation uses the Z-closure method (Huson *et al*., 2004) to infer clusters on the full taxon set.

The incompatibility graph is then computed and the list of connected components is generated. Each such component is 'compacted', i.e. every maximal subset of taxa which is not separated by any cluster in the component is represented by only one of the taxa in the subset. For each compacted component $\mathcal{C}'$ on $\mathcal{X}' \subseteq \mathcal{X}$, we determine the smallest set $R \subset \mathcal{X}'$ such that $\mathcal{C}'|_{\mathcal{X}'-R}$ is compatible, and then produce an optimal solution of the MAP for the component. The solution found is encoded as a set of clusters with the property that the Hasse diagram of the set of clusters is precisely the rooted network structure that we are looking for. This is a useful trick, because it makes 'uncompacting' clusters and putting together the results from different components straightforward to do.

However, the structure of a galled network cannot always be completely described in terms of a set of generating clusters in this way: it may be necessary to add a direct edge from a node v to one of its descendants w, namely to be able to completely turn off a reticulate node in the representation of some cluster. To address this, we maintain a list of pairs of clusters that describe pairs of such nodes that are to be joined by additional edges after the Hasse diagram of the modified cluster set has been computed.

In Section 2, we showed that the cluster containment problem is easy for galled networks. Based on this, our implementation contains an algorithm that verifies that a computed galled network does indeed contain all clusters that are present in the input.

Dendroscope was designed to handle very large trees. In this vein, we have been careful to ensure that our implementation of galled networks works well on large datasets, too. The running time of the algorithm depends on the maximum number of reticulations required by any compacted component of the incompatibility graph. If this number is reasonable, then our implementation will happily work well on trees containing hundreds or thousands of taxa.

As discussed above, to obtain an optimal galled network we must solve a number of instances of two problems that are both computationally hard, the Maximum Compatible Subset problem and the MAP. To avoid frustrating the user, when attempting to solve an instance of either problems, the user is presented with a cancelable progress bar dialog. If he or she decides to cancel a computation, then the program takes the best partial solution found so far and then greedily extends it to a full solution in very short time. In this way, the program can always be used to produce a valid galled network, even when the user does not have the time to wait for a guaranteed optimal one.

7 APPLICATION TO REAL DATA

7.1 Application to grasses

An algorithm for computing the optimal number of hybridization nodes in a rooted phylogenetic network N that displays both of two given input trees is presented by Bordewich *et al*. (2007), a computationally hard problem, as they show. At present, their implementation does not provide an actual network that attains the number, but this is under development (C.Semple, personal communication).

Table 1. For all possible pairs of five trees on grasses (Grass Phylogeny Working Group, 2001), we report the number of taxa shared by both trees, the number of hybridization nodes and run-time reported in (Bordewich *et al*., 2007) and the number of reticulate nodes and run-time obtained using the algorithm described in this article.

First tree	Second tree	Common taxa	Hybrid nodes	Time	Reticulate nodes	Time(s)
ndhF	phyB	40	14	11 h	9	<2s
ndhF	rbcL	36	13	11.8 h	8	<2s
ndhF	rpoC2	34	12	26.3 h	10	<2s
ndhF	waxy	19	9	320 s	6	<2s
ndhF	ITS	46	>15	2days	23	<2s
phyB	rbcL	21	4	1s	6	<2s
phyB	rpoC2	21	7	180 s	4	<2s
phyB	waxy	14	3	1 s	3	<2s
phyB	ITS	30	8	19 s	9	<2s
rbcL	rpoC2	26	13	29.5 h	9	<2s
rbcL	waxy	12	7	230 s	4	<2s
rbcL	ITS	29	>9	2days	15	<2s
rpoC2	waxy	10	1	1 s	2	<2s
rpoC2	ITS	31	>10	2days	14	<2s
waxy	ITS	15	8	620 s	5	<2s

Bordewich and colleagues study five different trees on a grass (*Poaceae*) dataset provided by the Grass Phylogeny Working Group (Grass Phylogeny Working Group, 2001), ranging in size from 19 to 65 taxa. They apply their algorithm on all pairs of trees in the input set, in each case restricting the input trees to the set of taxa that both trees have in common. They report the optimal number of hybridization nodes and the computational time required. In Table 1, we list their results and compare them with the results obtained by running our new algorithm on the same datasets. We show the networks obtained in Figure 5.

There are two important observations to be made. The first is that the number of hybridization nodes and reticulate nodes is not always the same. The networks considered by Bordewich and colleagues are more general, so a smaller number of hybridization nodes is possible. On the other hand, they search for a network that contains both trees as subtrees, whereas our algorithm seeks only to represent their clusters. In all cases where we report a lower number of reticulations, it is true that at least one of the two input trees is not contained in the network as a tree (although all its clusters are). At first glance, this may seem a weakness of our approach. However, biologists are usually focused on the clades or monophyletic groups rather than on the trees that contain them, and so in practice this is not a problem.

The second observation is that the runtime required by our algorithm was <2 seconds in all cases, whereas the algorithm by Bordewich and colleagues took a number of days in some cases, without completion.

7.2 Application to prokaryotes dataset

To test the algorithm on a larger dataset, we considered nine different gene trees on a set of 279 prokaryotes (Auch *et al.,* manuscript in preparation). To evaluate the running time on an extremely hard

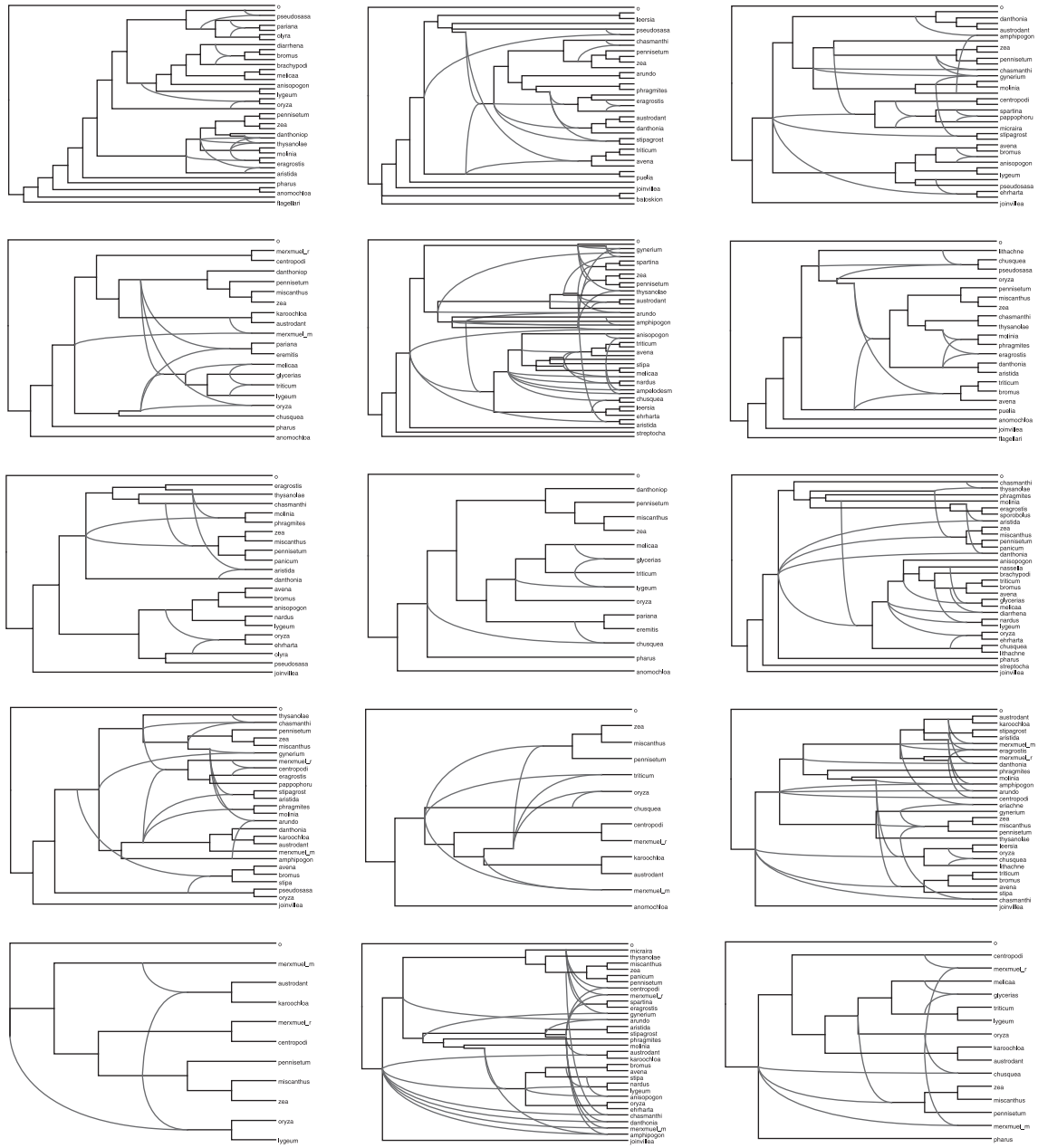


Fig. 5. Galled networks computed by Dendroscope, using the algorithms presented in this article, for 15 different pairs of trees on grasses (Grass Phylogeny Working Group, 2001), see Table 1.

dataset, we ran our algorithm on the set of all 1334 clusters contained in the nine trees. As it turned out, most clusters are involved in one large incompatibility component with 13 337 incompatibility statements. Our code for solving the Maximum Compatible Subset problem required < 10 min. to solve this component, whereas the code for solving the Minimum Attachment problem was unable to find a guaranteed minimum solution within an acceptable time. The network produced used 179 reticulate nodes. For a less difficult dataset, we ran the algorithm on all clusters contained in more than one input tree. This gave rise to a dataset of 655 clusters and the most difficult incompatibility component had 29 incompatibility statements, which required 4 reticulations to resolve. In this case, the galled network was obtained in less than 2 seconds, and we display it in Figure 6.

ACKNOWLEDGEMENTS

This work was initiated by DHH and RR during the Phylogeny Programme at the Newton Institute of Cambridge University in 2007. We would like to thank Magnus Bordewich, Vincent Moulton and Charles Semple for many helpful discussions, and Johannes

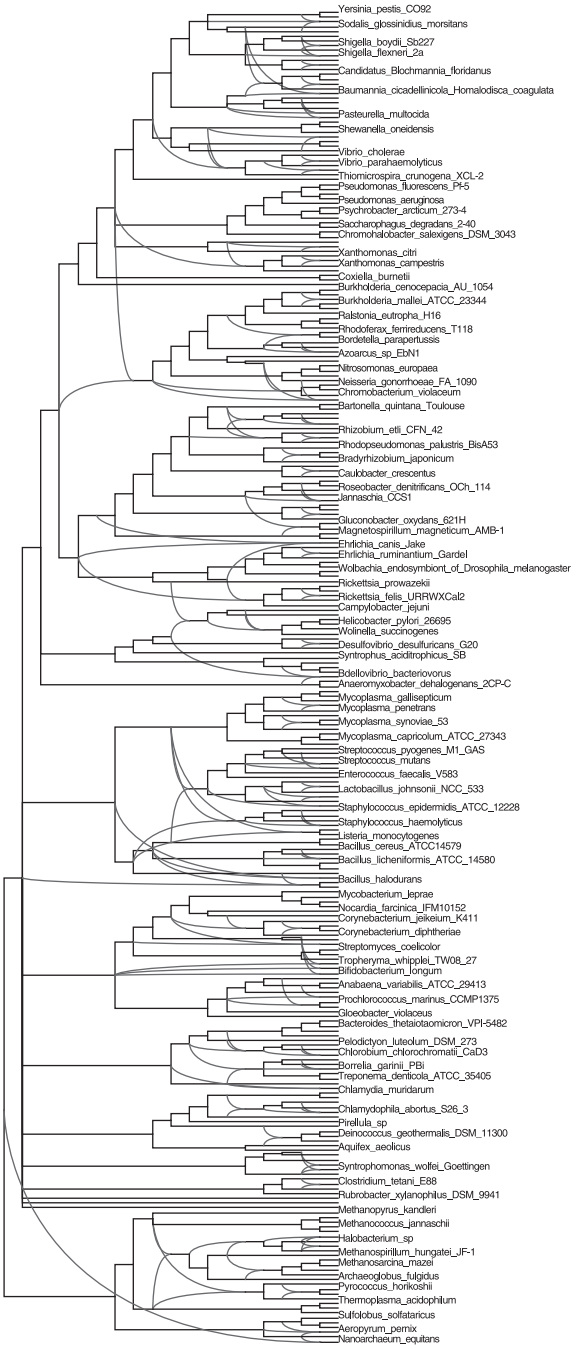


Fig. 6. A galled network on 279 prokaryote species, computed from all clusters that are contained in more than one of nine different gene trees.

Fischer for giving us feedback on the article. Thanks to Alexander Auch for providing the nine gene trees on prokaryotes.

Funding: DFG (to R.R.)grant Hu 566/5-1. French ANR project ANR-08-EMER-011-01.

Conflict of Interest: none declared.

REFERENCES

Bandelt,H.J. and Dress,A.W.M. (1992) A canonical decomposition theory for metrics on a finite set. *Adv. Math.*, **92**, 47–105.

Berry,V. and Nicolas,F. (2006) Improved parameterized complexity of the maximum agreement subtree and maximum compatible tree problems. *IEEE/ACM TCBB*, **3**, 289–302.

Bordewich,M. *et al*.(2007) A reduction algorithm for computing the hybridization number of two trees. *Evol. Bioinform.*, **3**, 86–98.

Bryant,D. and Moulton,V. (2002) NeighborNet: An agglomerative method for the construction of planar phylogenetic networks. In Guigó,R. and Gusfield,D. (eds), *Algorithms in Bioinformatics, WABI 2002*. vol. 2452, *Lecture Notes in Computer Science*, Springer Berlin/Heidelberg, pp. 375–391.

Charbit,P. *et al*. (2008) A note on computing set overlap classes. *Inform. Process. Lett.*, **108**, 186–191.

Cohen,J.E. (2004) Mathematics is Biology's next microscope, only better; Biology is Mathematics' next Physics, only better. *PLoS Biol.*, **2**, e439.

Doolittle,W.F. and Bapteste,E. (2007) Pattern pluralism and the tree of life hypothesis. *Proc. Natl Acad. Sci. USA*, **104**, 2043–2049.

Grass Phylogeny Working Group. (2001) Phylogeny and subfamilial classification of the grasses (poaceae). *Ann. M. Bot. Gard.*, **88**, 373–457.

Gusfield,D. and Bansal,V. (2005) A fundamental decomposition theory for phylogenetic networks and incompatible characters. In *Proceedings of the Ninth International Conference on Research in Computational Molecular Biology (RECOMB)*. Cambridge, MA, USA, pp. 217–232.

Gusfield,D. *et al*. (2003) Efficient reconstruction of phylogenetic networks with constrained recombination. In *Proceedings of the IEEE Computer Society Conference on Bioinformatics*. Stanford, CA, USA, pp. 363.

Gusfield,D. *et al*. (2007) A decomposition theory for phylogenetic networks and incompatible characters. *J. Computat. Biol.*, **14**, 1247–1272.

Hallett,M. and Lagergren,J. (2001) Efficient algorithms for lateral gene transfer problems. In *5th Annual RECOMB Montreal, April 22–25*, pp. 149–156.

Hein,J. (1993) A heuristic method to reconstruct the history of sequences subject to recombination. *J. Mol. Evol.* **36**, 396–405.

Huber,K.T.*et al*.(2007) Reconstructing the evolutionary history of polyploids from multilabeled trees. *Mol. Biol. Evol.*, **23**, 1784–1791.

Huson,D.H. and Kloepper,T.H. (2005) Computing recombination networks from binary sequences. *Bioinformatics*, **21 (Suppl. 2)**, ii159–ii165. ECCB.

Huson,D.H. and Bryant,D. (2006) Application of phylogenetic networks in evolutionary studies. *Mol. Biol. Evol.*, **23**, 254–267.

Huson,D.H and Kloepper,T.H. (2007) Beyond galled trees - decomposition and computation of galled networks. In *Proceedings of the Eleventh International Conference on Research in Computational Molecular Biology (RECOMB)*. Vol. 4453 of *Lecture Notes in Computer Science*, Oakland, CA, USA, pp. 211–225.

Huson,D.H. and Rupp,R. (2008) Summarizing multiple gene trees using cluster networks. In Crandall,K. and Lagergren,J. (eds), *Algorithms in Bioinformatics, WABI 2008*. Vol. 5251 in *Lecture Notes in Bioinformatics (LNBI)*. Springer, Berlin/Heidelberg, pp. 211–225.

Huson,D.H. *et al*. (2004) Phylogenetic super-networks from partial trees. *IEEE/ACM TCBB*, **1**, 151–158.

Huson,D.H. *et al*.(2005) Reconstruction of reticulate networks from gene trees. In *Proceedings of the Ninth International Conference on Research in Computational Molecular Biology (RECOMB)*, Vol. 3500 of *Lecture Notes in Computer Science*. Springer Verlag, Heidelberg, pp. 233–249.

Huson,D.H. *et al*. (2007) Dendroscope: an interactive viewer for large phylogenetic trees. *BMC Bioinformatics*, **8**, 460.

Kanj,I.A. *et al*. (2008) Seeing the trees and their branches in the network is hard. *Theor. Comput. Sci.*, **401**, 153–164.

Linder,C.R. and Rieseberg,L.H. (2004) Reconstructing patterns of reticulate evolution in plants. *Am. J. Bot.*, **91**, 1700–1708.

Lyngsø,R.B. *et al*. (2005) Minimum recombination histories by branch and bound. In *Proceedings of the Workshop on Algorithms in Bioinformatics (WABI)*. Mallorca, Spain, pp. 239–250.

Ma,B.*et al*. (1998) Fixed topology alignment with recombination. In *Proceedings of the 9th Annual Symposium on Combinatorial Pattern Matching (CPM'98)*. Vol. 1448 of *Lecture Notes in Computer Science*. pp. 174–188.

Mirkin,B.G. *et al*. (2003) Algorithms for computing parsimonious evolutionary scenarios for genome evolution, the last universal common ancestor and dominance of horizontal gene transfer in the evolution of prokaryotes. *BMC Evol. Biol.*, **3**, 2.

Steel,M. and Hamel,A. (1996) Finding a maximum compatible tree is NP-hard for sequences and trees. *Appl. Math. Lett.*, **9**, 55–60.

Song,Y.S. and Hein,J. (2005) Constructing minimal ancestral recombination graphs. *J. Comp. Biol.*, **12**, 147–169.

Than,C. *et al*. (2008) Phylonet: a software package for analyzing and reconstructing reticulate evolutionary relationships. *BMC Bioinformatics*, **9**, 1–16.

van Iersel,L. and Kelk,S. (2008) Constructing the simplest possible phylogenetic network from triplets. In *Proceedings of ISAAC'08*, Vol. 5369, Springer, Berlin/Heidelberg, Gold Coast, Australia, pp. 472–483.

van Iersel,L. *et al*. (2008) Constructing level-2 phylogenetic networks from triplets. In *International Conference on Computational Molecular Biology (RECOMB)*. Vol. 4955 of *Lecture Notes in Computer Science*, Springer Verlag, Berlin/Heidelberg, pp. 450–462.

Wang,L. *et al*. (2001) Perfect phylogenetic networks with recombination. In *Proceedings of the 16th ACM Symposium on Applied Computing (SAC'01)*. Las Vegas, NV, USA, pp. 46–50.

APPENDIX

Proof of Lemma 2: RMCS is hard

We consider the decision problem of knowing whether there exists a set of k taxa whose removal resolves all the incompatibilities in a set of clusters, shown to be hard in Steel and Hamel (1996).

We then show by reduction of this decision problem for general cluster sets that the decision problem for a set of clusters with properties (P1) and (P2) is NP-hard, which implies the NP-hardness of the MCS problem for such input. Let $\mathcal{C}$ be a set of clusters on $\mathcal{X}$, not necessarily fulfilling properties (P1) and (P2). Set $\mathcal{X}'=\mathcal{X}\cup\{o\}$, where o is some new taxon not contained in $\mathcal{X}$, and let $\mathcal{C}_1$ denote the set of all trivial clusters, i.e. clusters which contain only one taxon. Define $\mathcal{C}'=(\mathcal{C}\setminus\mathcal{C}_1)\cup\{\mathcal{X}\}\cup\{\{o,x\}\,|\,x\in\mathcal{X}\}$. Note that by construction, the set of clusters $\mathcal{C}'$ on $\mathcal{X}'$ has properties (P1) and (P2). We prove that there is a solution R' of size $k+1$ of the restricted problem for $\mathcal{C}'$ iff there is a solution R of size k of the general problem for $\mathcal{C}$.

$\Rightarrow$: Let R' be a solution of the restricted problem for $\mathcal{C}'$ of size $k+1$, there are two cases to consider.

CASE 1. $o\in R'$. Then $R=R'\setminus\{o\}$ is of size k and removes all the incompatibilities from $\mathcal{C}$.

CASE 2. $o\notin R'$. In this case, the set R' must contain all but one element of $\mathcal{X}$, that is, $|R'|=k+1=|\mathcal{X}|-1$, because otherwise there would remain two clusters of the form $\{o,x\}$ and $\{o,y\}$, which are incompatible. Then any subset $R\subset\mathcal{X}$ of size $|\mathcal{X}|-2=k$ is a valid solution for $\mathcal{C}$, because a collection of clusters on only two taxa cannot be incompatible.

$\Leftarrow$: Let R be a solution of the general problem for $\mathcal{C}$ of size k. We consider $R'=R\cup\{o\}$. This set has size $k+1$, removes all incompatibilities between clusters $\{o,x\}$ as it removes taxon o. All other incompatibilities are removed because $R\subset R'$. ■

Proof of Lemma 3: Performance of the seed-growing algorithm

We claim that if a minimum element $s\in S$ contains k taxa, then S contains at most 3^{k+1} sets. Consider the enumeration tree of seeds generated by the algorithm. For the purpose of this proof, let the *level* of a node v be the number of edges in the path from the root of the enumeration tree to v. At the beginning of an iteration, the algorithm chooses a seed s of minimum cardinality $|s|$. By construction, the level of the corresponding node will be at most $|s|$. If s solves the next incompatibility, then no new nodes are added to the enumeration tree. Otherwise, three new nodes are added, each representing some seed whose cardinality is strictly larger than that of s. Thus, the enumeration tree will have at most $k+1$ different levels, and each level will have at most three times as many nodes as the previous level. This implies that the number of seeds considered in S is at most 3^{k+1}. ■

Proof of Lemma 4: The MAT Problem is hard

Here, we sketch a reduction of the *Set Cover (SC)* problem to the *MAT* problem. Further details of the proof will be provided elsewhere. Recall that in the SC problem, we are given a collection of sets $\mathcal{C}$ on a set $\mathcal{X}$ and a number k. The question is: does there exist a subset $\mathcal{S}\subseteq\mathcal{C}$ of k sets that covers $\mathcal{X}$? We will construct an instance of MAT that has a solution using $m+k$ edges if and only if SC has a solution using k sets, with $m=|\mathcal{C}|$. Let d be an auxiliary taxon not contained in $\mathcal{X}$ and define $\mathcal{X}'=\mathcal{X}\cup\{d\}$. We construct a phylogenetic tree T with root ρ and $m+1$ leaves. Set $R(e_0)=\mathcal{X}'$ for the first leaf edge $e_0=(\rho,v_0)$ of T and set $R(e_i)=\{d\}$ for all other leaf edges $e_1=(\rho,v_1),\ldots,e_m=(\rho,v_m)$ of T. This defines the top part of the graph used in the Attachment problem. For the bottom part, B, define one *cluster node* $v(C\cup\{d\})$ for each set $C\in\mathcal{C}$ and one reticulate node $v(r)$ for each $r\in\mathcal{X}'$. Place an edge from $v(C\cup\{d\})$ to $v(r)$ for each taxon r contained in $C\cup\{d\}$. Note that we can assume that each $r\in\mathcal{X}$ is contained in at least two different sets in $\mathcal{C}$, otherwise we can reduce the instance of SC to a smaller one. This observation ensures that any solution to the MAT problem will fulfill property (A3). We will now prove the following *claim:* if there exists a solution of this instance of MAT using $m+k$ edges, then there is one in which the node v_0 is only connected to nodes of the form $v(C\cup\{d\})$, with $C\in\mathcal{C}$. It follows from properties (A1) and (A2) that there are k edges that connect v_0 to B and m edges that connect nodes $v_1,\ldots,v_m$ to B. Since all edges departing from v_0 lead to cluster nodes in B, then this set of cluster nodes defines a subset of clusters $\mathcal{S}$ that covers $\mathcal{X}'$, and thus, also $\mathcal{X}$, providing a solution to the SC problem of size k. Now, to prove the claim, assume one of the edges departing from v_0 leads directly to a node of the form $v(r)$ in B, for some taxon $r\in\mathcal{X}'$. If there is a cluster node $v(C\cup\{d\})$ connected to v_0 with $x\in C\cup\{d\}$, then we can remove the edge from v_0 to $v(r)$, as it is superfluous. Otherwise, there exists a cluster node $v(C\cup\{d\})$ that is attached below some node v_i in T, with $i>0$. In this case, we modify the solution as follows: redirect the edge from v_0 to $v(r)$ so that it leads to $v(C\cup\{d\})$ and redirect the edge from v_i to $v(C\cup\{d\})$ so that it leads to $v(r)$. This is repeated until all children of v_0 are cluster nodes.

Vice versa, it is also not difficult to see that any solution with k sets of an instance of the SC problem leads to a solution with $m+k$ edges of this simplified case of the MAT problem. ■