Fast Computation of the Exact Hybridization Number of Two Phylogenetic Trees

Yufeng Wu and Jiayin Wang

Department of Computer Science and Engineering
University of Connecticut
Storrs, CT 06269, U.S.A.
ywu,jiw09003@engr.uconn.edu

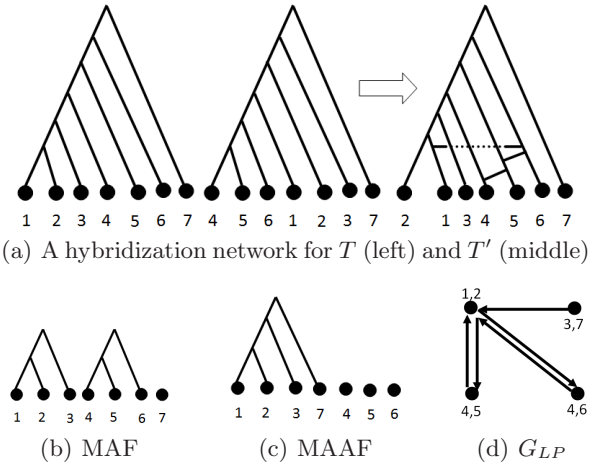
Abstract. Hybridization is a reticulate evolutionary process. An established problem on hybridization is computing the minimum number of hybridization events, called the hybridization number, needed in the evolutionary history of two phylogenetic trees. This problem is known to be NP-hard. In this paper, we present a new practical method to compute the exact hybridization number. Our approach is based on an integer linear programming formulation. Simulation results on biological and simulated datasets show that our method (as implemented in program *SPRDist*) is more efficient and robust than an existing method.

1 Introduction

Recently, *reticulate* evolutionary models have been actively studied in Phylogenetics. Several models have been proposed to address different reticulate processes (e.g. hybridization, lateral gene transfer and recombination). The literature on various aspects of reticulate evolution is growing rapidly. Refer to [12, 11, 18, 15] for surveys of reticulate evolution. In this paper, we focus on *hybridization*. Hybridization refers to hybrid speciation, where hybrid species with mixed genetic composition from different species are created. Hybridization is believed to occur in many species [18].

Imagine we have two phylogenetic trees (called gene trees), each for a gene of a (same) set of species. Due to reticulate evolution, the two gene trees are related but different. In this case, the evolutionary history of the two gene trees can not be represented by a single tree, but rather should be modeled as a *network*. This network is called "hybridization" network [5, 18], which is closely related to the phylogenetic network or reticulate network [12, 11, 18, 15]. Hybridization network for two gene trees is a directed acyclic graph, which is a compact representation of the two trees in the sense that it "displays" the two trees in a compact way. See Figure 1(a) for an example of hybridization networks.

Since there exist many feasible hybridization networks for two gene trees, a common formulation is to find the one with the *fewest* hybridization events. This is motivated by the belief that hybridization is relatively rare and thus the number of hybridization events is likely to be small. We call the *minimum*



(a) A hybridization network for T (left) and T' (middle)

(b) MAF (c) MAAF (d) G_{LP}

Fig. 1. (a): A hybridization network with three hybridization events. (b): maximum agreement forest of size three for T and T'. (c): maximum acyclic agreement forest of size four. (d): *part* of the leaf pair graph $G_{LP}(T, T')$.

number of hybridization events in any hybridization network the *hybridization number* of the two gene trees. For a hybridization network of two gene trees, its hybridization number is equal to the number of nodes with in-degree two in the network. For example, the network in Figure 1(a) has three nodes with in-degree two, and thus its hybridization number is three. One should note that although hybridization number is only a quantity of the most parsimonious hybridization network, algorithms for computing the hybridization number can allow easy reconstruction of a parsimonious network itself as a by-product.

An established computational problem on hybridization networks, the *hybridization number problem*, is: given two trees, compute the hybridization number of the two trees. It is known that the hybridization number problem is NP-hard [8]. Nonetheless, computing the hybridization number of two trees has been studied in several papers [2, 1, 5, 14]. A fundamental result in [1] showed that the hybridization number of two trees is equal to the size of the so-called Maximum Acyclic Agreement Forest (or MAAF) for the two trees minus one. See Section 2 for the definition of the MAAF formulation. There is a program, called *HybridNumber* [5], that can compute the exact hybridization number for two trees of moderate size or topologically similar. Program *HybridNumber* has worst-case exponential running time but is practical for certain range of data. The initial version of program *HybridNumber* was quite slow [5]. A later version of program *HybridNumber* appears to be much faster and more scalable [13]. However, our experience shows that program *HybridNumber* is still slow for larger data and also unstable in some cases: it crashed for some trees we tested due to software error. This greatly limits its application for larger biological data.

In this paper, we present a new method for computing the exact hybridization number of two gene trees based on an integer linear programming (ILP)

formulation. We also use a divide-and-conquer approach (developed in [5] and also used by program *HybridNumber*) in order to reduce the size of the problem. Our method is implemented in program *SPRDist*, which also outputs the corresponding maximum acyclic agreement forest and allows easy reconstruction of the most parsimonious hybridization network. To demonstrate the performance of our method, we provide simulation results on biological and simulated datasets. Comparing with program *HybridNumber*, our program *SPRDist* is more *efficient* for large trees, and also more *robust*.

2 Background

In this paper, we let a phylogenetic tree T be a binary *rooted* leaf-labeled tree. The set of the leaf labels of T is denoted as $L(T)$, its set of branches as $E(T)$, and its set of vertices as $V(T)$. Given a tree T, we can create a forest of trees $F(T) = \{T_1, T_2, \ldots, T_k\}$ from T by deleting a subset of $E(T)$. The forest $F(T)$ induces a partition of $L(T)$, and any two trees T_i and T_j are node disjoint. Conversely, we say a list of trees T_i form a forest for T if (a) for any tree T_i, $L(T_i) \subseteq L(T)$ and the union of $L(T_i)$ is equal to $L(T)$; (b) for each T_i, the (unique) *minimal* subtree connecting the nodes in $L(T_i)$, denoted as $S(T_i)$, is identical to T_i when nodes with degree two of $S(T_i)$ are contracted (called cleanup); if this holds, we say T_i *appears* in T; and (c) for any two trees T_i and T_j, $S(T_i)$ and $S(T_j)$ are node disjoint. The size of a forest is the number of trees in the forest.

Throughout this paper, we let two phylogenetic trees T and T' with the same set of leaf labels be the input data. Without loss of generality, we assume leaves are labeled with distinct integers from 1 to n, where n is the number of species. For convenience, we assign a distinct integer to each (leaf or internal) node in T and T', and thus we use the integer to refer to the node. When no confusion is caused, we assign integer i to a leaf labeled with i in both T and T'.

The concept of agreement forest has been used in many previous papers (e.g. [10, 16, 4, 6]) and is also crucial to the current work. An agreement forest $F(T, T')$ is a set of trees $T_1, T_2, \ldots, T_k$ that is a forest for *both* T and T'. See Figure 1(b) for an illustration of agreement forest. Intuitively, an agreement forest is derived by cutting the same number of branches of T and T' which leads to the same set of trees (after cleanup). Agreement forest always exists for any two phylogenetic trees with the same set of leaves: a forest with n trees, each with a distinct leaf, is an agreement forest. We are interested in the agreement forest with the *smallest* number of trees (called maximum agreement forest or MAF). It is easy to see that the agreement forest in Figure 1(b) is also a MAF.

A useful observation [10, 7] is that the rooted subtree prune and regraft (rSPR) distance between two trees is equal to the number of trees in a MAF minus one [10, 7]. Recently, we developed an integer linear programming formulation (ILP), which can find the MAF of two trees and thus also compute the exact rSPR distance for many data [20]. Note that the two trees should be preprocessed to add a dummy leaf out of the root of each tree to ensure correctness,

which we perform throughout this paper. We now briefly describe the ILP formulation for the rSPR distance problem, since we will extend this formulation to the hybridization number problem.

Original ILP formulation for the MAF problem The objective of the MAF problem is to find how to divide T and T' by cutting the *fewest* edges to derive an agreement forest. We define a binary variable C_i for each edge e_i in T (with m edges), where $C_i = 1$ if edge e_i is cut and 0 otherwise. We only place branch cuts in T and use T' as a reference. The objective of the ILP formulation is to minimize $\sum_{i=1}^{m} C_i$. We consider three leaves in both T and T'. We call the subtree connecting the three leaves in T the *triple* in T. Some triples do not agree in T and T'. For example, in Figure 1(a), the topology of triple of 1, 2 and 4 in T is different from that in T'. We call such triple "incompatible", meaning that this triple in T conveys different topological information from that in T'. To ensure the resulting forest to be an agreement forest, we require at least one edge of each incompatible triple is cut. Moreover, for two leaves i and j, we say the edges in T connecting i and j form a path between i and j. In Figure 1(a), the path between leaf pair $(1, 2)$ intersects that of leaf pair $(3, 4)$ in T' of Figure 1(a), but is disjoint in T. We call such two leaf pairs "incompatible", since the two paths for the two leaf pairs can not both be left uncut in an agreement forest. To ensure the resulting forest is an agreement forest, we require at least one edge along the two paths of two incompatible leaf pairs is cut. Both types of constraints can be easily expressed in ILP. The correctness of the ILP formulation for the MAF problem was established in [20]. See [20] for more details.

A useful observation on the hybridization number problem is made in [1], which concerns the so-called maximum *acyclic* agreement forest (MAAF). Maximum acyclic agreement forest is a special kind of agreement forest with one additional constraint, which concerns the topological order of the trees in the agreement forest. For agreement forest $F(T, T') = \{T_1, T_2, \ldots, T_k\}$, we say T_i is ancestral to T_j if the root of T_i is ancestral to the root of T_j in *either* T *or* T'. Here, a node v is ancestral to a node v' in T if v is on the (unique) path from the root to v' in T. Suppose we create a directed graph $G(F(T, T'))$ whose nodes are in one-to-one correspondence with $\{T_i\}$ (and thus we use T_i to refer both the node in the graph and the corresponding tree). To simplify notations, we write $G(F(T, T'))$ as $G(F)$. We create an edge from T_i to T_j if T_i in $G(F)$ is ancestral to T_j. An agreement forest $F(T, T') = \{T_1, T_2, \ldots, T_k\}$ is *acyclic* if the graph $G(F)$ is acyclic. As an example, the forest in Figure 1(b) is not acyclic, while the one in Figure 1(c) is. This is because in Figure 1(b), the tree with leaves 1, 2 and 3 is ancestral to the tree with leaves 4, 5 and 6 in T', and ancestral relationship is reversed in T. This leads to a cycle of two trees in $G(F)$. There are no cycles for the forest in Figure 1(c). We say a forest is maximum acyclic agreement forest (MAAF) if the forest is a MAF and acyclic. The forest in Figure 1(c) is a MAAF. The following theorem is proved in [1].

Theorem 1. *[1] The hybridization number of T and T' is equal to the size of the MAAF for T and T' minus one.*

3 Computing the Hybridization Number with ILP

Now we present an ILP formulation for finding the maximum acyclic agreement forest (MAAF) for T and T', which also allows the computation of the hybridization number due to Theorem 1. This ILP formulation also finds the MAAF, which can be used to reconstruct the most parsimonious hybridization network. We also apply a divide and conquer approach to speed up the computation.

3.1 New ILP formulation for the MAAF problem

Our ILP formulation is an extension of the ILP formulation for the MAF problem, as described in Section 2. We uses binary variable C_i to indicate whether edge e_i in T is cut or not. As in [20], we create binary variable $M_{i,j}$ for two (leaf or internal) nodes i and j in T. $M_{i,j} = 1$ iff *none* of the edges along the path between i and j is cut and 0 otherwise. Since a MAAF is also a MAF, we create the same set of ILP constraints as in the MAF formulation to ensure the resulting forest is an agreement forest. We now focus on how to ensure the resulting agreement forest $F(T, T')$ is acyclic in the ILP formulation.

Suppose the resulting forest $F(T, T')$ is known. Then it is straightforward to construct the corresponding graph $G(F)$ and test whether $G(F)$ is acyclic or not. A major problem here is that $G(F)$ depends on the agreement forest $F(T, T')$, which is exactly what we want to find. Without knowing $F(T, T')$, we can not explicitly construct $G(F)$. One may consider enumerating all possible agreement forests and then impose ILP constraints to forbid cycles. But enumerating agreement forests is impractical in most cases.

To get around this difficulty, our main idea is to consider *leaf pairs* in T. First note that the number of leaf pairs is much smaller than the number of possible agreement forests: there are $O(n^2)$ leaf pairs for n leaves. What is more important is that the acyclicity of $G(F)$ can be enforced using leaf pairs selected from the trees in $F(T, T')$ as we will explain later. Note that we do not know which leaf pairs i and j are in the same T_i without knowing $F(T, T')$. We do know, however, for each leaf pair of i and j, i and j are in the same tree of the forest iff $M_{i,j} = 1$. We now introduce the key tool of our approach: the leaf pair graph.

Leaf pair graph. We denote the leaf pair of two distinct leaves i and j as $lp(i, j)$. We say $lp(i, j)$ is connected if $M_{i,j} = 1$ (i.e. no branch is cut along the path between i and j) in T and also in T'. We say $lp(i, j)$ is from tree $T_i \in F(T, T')$ if $i, j \in L(T_i)$. Each connected leaf pair must be from some T_i of the forest. For two leaves i and j, we denote $MRCA_T(i, j)$ (respectively $MRCA_{T'}(i, j)$) as the most recent common ancestor of i and j in T (respectively T'). We say leaf pair $lp(i, j)$ is ancestral to leaf pair $lp(p, q)$ in T if $MRCA_T(i, j)$ is ancestral to $MRCA_T(p, q)$. We now construct the leaf pair graph $G_{LP}(T, T')$ (or simply G_{LP}), which is a directed graph. The nodes in G_{LP} are in one-to-one correspondence to the leaf pairs in T, and so we can use the leaf pairs to refer to the nodes of G_{LP}. For two leaf pairs $lp(i, j)$ and $lp(p, q)$ in G_{LP}, we create an edge from $lp(i, j)$ to $lp(p, q)$ if the following two conditions are *both* satisfied:

a. the path between i and j is disjoint with that of p and q in *both* T and T',
b. $lp(i,j)$ is ancestral to $lp(p,q)$ in *either* T *or* T'.

As an example, Figure 1(d) shows part of the leaf pair graph G_{LP} for the two trees in Figure 1(a). There is an edge from $lp(1,2)$ to $lp(4,5)$ because the MRCA of leaves 1 and 2 is ancestral to the MRCA of leaves 4 and 5 in T'. There is a degenerate case not covered by leaf pairs: leaf singletons in $F(T,T')$. But leaf singletons will not be part of any cycles in $G(F)$ because they can not be ancestral to any other trees: the root of a singleton tree is a leaf. Therefore, we only need to consider trees with at least two leaves from now on.

The reason that we require the two paths of two leaf pairs sharing an edge in G_{LP} to be disjoint is that we will only use one *realized* leaf pair per tree in the forest in our method. We say a leaf pair is realized in an agreement forest $F(T,T')$ if the two leaves are connected in $F(T,T')$. For a given agreement forest, some nodes (i.e. leaf pairs) of G_{LP} may not be realized. In this case, we remove from G_{LP} all leaf pairs that are not realized in $F(T,T')$ along with any edges incident to these leaf pairs. We denote the reduced G_{LP} as $G_{LP}(F)$, which is a sub-graph of G_{LP}. That is, $lp(i,j)$ appears in $G_{LP}(F)$ if i and j belong to the *same* tree in $F(T,T')$. We now show $G_{LP}(F)$ and $G(F)$ convey the same information on whether the agreement forest is acyclic or not.

Suppose $G_{LP}(F)$ contains a cycle, whose nodes are the (realized) leaf pairs. We first note that a cycle in $G_{LP}(F)$ can not contain only leaf pairs from a single tree of $F(T,T')$.

Lemma 1. *Cycles in $G_{LP}(F)$ contain leaf pairs from at least two trees in $F(T,T')$.*

Proof. For contradiction, suppose there exists a cycle in $G_{LP}(F)$ where all leaf pairs along the cycle are from a single tree in $F(T,T')$. Among these leaf pairs, we consider the leaf pair lp with the highest MRCA (i.e. closest to the root of the tree). Clearly, there exists no edge in G_{LP} that originates from some leaf pair on the cycle and points to lp. This contradicts the assumption that lp is on the cycle. □

Lemma 2. *For an agreement forest F, $G(F)$ is acyclic iff $G_{LP}(F)$ is acyclic.*

Proof. We will show that if $G(F)$ contains a cycle, then $G_{LP}(F)$ also contains a cycle, and vice versa. First suppose there is a cycle in $G(F)$ and suppose the cycle contains tree $T_{i_1},\ldots,T_{i_c}$. Then there exists one connected leaf pair from each T_{i_j} such that their MRCAs are the same as the *roots* of the trees which they belong to. These leaf pairs thus have the same ancestral relationship as T_{i_j}. Since the leaf pairs belong to different trees, the leaf pairs are pairwise disjoint. By the definition of $G_{LP}(F)$, these leaf pairs form a cycle in $G_{LP}(F)$.

Now suppose $G_{LP}(F)$ contains a cycle C. Due to Lemma 1, this cycle contains leaf pairs from at least two trees of $F(T,T')$. Note that each leaf pair on C must belong to some tree of $F(T,T')$, but there can be multiple leaf pairs belong to the same tree. We let $T'_1,T'_2,\ldots,T'_c$ be the *distinct* trees, which is ordered along C. Here $c \geq 2$ due to Lemma 1. We now show $T'_1,T'_2,\ldots,T'_c$ forms a cycle.

Consider two leaf pairs $lp(i,j)$ and $lp(p,q)$ that are consecutive along C. Moreover, they are from two different trees T'_i and T'_{i+1} respectively, and there is an edge from $lp(i,j)$ to $lp(p,q)$ in $G_{LP}(F)$. Then, in $G(F)$, there is an edge from T'_i to T'_{i+1}. To see this, without loss of generality assume $MRCA_T(i,j)$ is ancestral to $MRCA_T(p,q)$. Since T'_i and T'_{i+1} are disjoint, the root of T'_{i+1} must be on the path from $MRCA_T(i,j)$ to $MRCA_T(p,q)$ in T. Thus, there is a path from the root of T'_i to the root of T'_{i+1} in T, which leads to an edge from T'_i to T'_{i+1}. Since each T'_i has at least one leaf pair in C, there is an edge in $G(F)$ from each T'_i to T'_{i+1}. Therefore, trees $T'_1, \ldots, T'_c$ form a cycle in $G(F)$. □

Lemma 2 implies that if $G_{LP}(F)$ is acyclic, the resulting forest is acyclic. We will create ILP constraints on the leaf pairs that ensure acyclicity for the resulting $G_{LP}(F)$. Note that the proof of Lemma 2 suggests that we only need one leaf pair from a single tree of the forest.

If G_{LP} is acyclic, then *any* agreement forest created by some branch cuts will be acyclic. So we assume in the following that G_{LP} contains cycles. Imagine we enumerate all cycles in G_{LP}. If a leaf pair on a cycle is not realized by the resulting forest, the cycle will be absent from $G_{LP}(F)$. To ensure an agreement forest to be acyclic, we enforce for each cycle C, at least one node in C is not realized (i.e. the corresponding leaf pair is not connected in $F(T,T')$). Unfortunately, experimental study shows that G_{LP} can contain many cycles. This makes enumeration of cycles impractical. However, an *empirical* finding is that if we remove the so-called infeasible twin-pairs (defined later) from G_{LP}, then the reduced G_{LP} often contains a small number of cycles (and for many biological datasets, G_{LP} becomes acyclic).

Consider two leaf pairs $lp(i,j)$ and $lp(p,q)$, where there is an edge from $lp(i,j)$ to $lp(p,q)$ and an edge from $lp(p,q)$ to $lp(i,j)$ in G_{LP}. Here, $lp(i,j)$ and $lp(p,q)$ form a cycle of two nodes in G_{LP}. We call these two leaf pairs in G_{LP} *infeasible* twin-pairs. As an example, in Figure 1(d), $lp(1,2)$ and $lp(4,5)$ form an infeasible twin-pair, and so do $lp(1,2)$ and $lp(4,6)$. The two leaf pairs are called infeasible because to achieve an MAAF, at least one of the two leaf pairs is not realized. We now create ILP constraints to ensure at least one leaf pair of an infeasible twin-pair is not realized in $G_{LP}(F)$. We then remove from G_{LP} the two edges between two leaf pairs forming an infeasible twin-pair. This is valid since one of the two leaf pairs will not be realized and edges incident to both leaf pairs will always be removed.

In general, after deleting the edges between the infeasible twin-pairs, the reduced G_{LP} can still contain cycles. But our experience shows that for biological data, the reduced G_{LP} often contains a small number of cycles. This permits us to simply *enumerate* all possible elementary cycles in the reduced G_{LP}. A cycle is called elementary if it does not contain a smaller cycle. To enumerate elementary cycles in the directed graph G_{LP}, we use the algorithm developed by Tarjan [19], which appears to work well in our simulation study.

We now give the details on how our ILP formulation ensures the resulting forest to be acyclic. Recall that we have a binary variable $M_{i,j}$ for each pair of leaves i and j, where $M_{i,j} = 1$ if i and j are connected in the forest. We first

consider the infeasible twin-pairs in G_{LP}. For an infeasible twin-pair with $lp(i,j)$ and $lp(p,q)$, we impose an ILP constraint so that at least one of the leaf pairs is not realized:

$$M_{i,j} + M_{p,q} \leq 1$$

Second, we create one constraint for each enumerated elementary cycle C in the reduced G_{LP}. Suppose leaf pairs $(i_1, j_1), \ldots (i_c, j_c)$ are the nodes of C. Then we need to ensure at least one of these nodes is not realized:

$$\sum_{p=1}^{c} M_{i_p,j_p} \leq c - 1$$

These additional constraints are necessary and sufficient for the original MAF ILP formulation to obtain a MAAF, which we omit the detailed proof due to the space limit. To find the MAAF, we need to find which edges e_i with $C_i = 1$ are in the ILP solution. Empirical results show that our ILP formulation can often be solved efficiently in practice for many data (see Section 4).

3.2 Speed Up Computation by Divide and Conquer

For larger trees, it is known that computing the hybridization number can be made faster by preprocessing the input trees [5]. There are three preprocessing rules known to reduce the size of input trees while preserving the *optimality* of the solution. As implemented in program *HybridNumber*, these rules sometimes greatly reduce the running time. So we also use these rules by preprocessing the input trees before solving the ILP formulation.

The preprocessing applies to both T and T'. The first rule is: when there exists a common subtree T_0 for T and T' with at least two leaves, we delete T_0 from both T and T' and replace it with a single node $v(T_0)$. The reduced trees have the same hybridization number as the original trees [5]. The second rule is called *Chain Reduction* in [5], which is targeted to a special type of tree topology called maximal chain. Our experience suggests often this rule can not be applied to trees we tested. We refer the readers to [5] for more details.

The last rule, called cluster reduction in [5], is potentially more useful. Suppose there exists subtree T_1 of T, and also subtree T_2 of T', such that T_1 and T_2 may be topologically different but have the *same* set of leaves. Then we cut T_1 from T and T_2 from T' so that T_1 and T_2 become two new phylogenetic trees. We also add a new leaf, s, to T and T' at the same positions where T_1 and T_2 are previously attached. As shown in [5], the hybridization number of the original T and T' is equal to the summation of the hybridization number of the updated T and T' and that of T_1 and T_2. This rule can be effective because it can divide a larger problem into two smaller problems in a divide and conquer manner. However, not all input trees can be reduced by this rule.

To apply these preprocessing rules, we search for pairs of subtrees T_1 and T_2 with identical leaves in T and T'. Once the subtrees T_1 and T_2 are found, we cut

T_1 from T and T_2 from T', and then use the ILP approach described in Section 3.1 to compute the hybridization number of T_1 and T_2. We continue to divide the reduced T and T' until the two trees are small enough.

4 Results

We have implemented our method in an ILP based software tool called *SPRDist*. Program *SPRDist* was originally designed to compute the rSPR distance for two trees. We have updated program *SPRDist* to allow computation of the hybridization number for two trees. The tool is written in C++ and uses the GNU GLPK integer linear programming solver or the commercial CPLEX solver. Our experience shows CPLEX (a commercial ILP solver) is often faster and more robust than GLPK. To test the effectiveness of our method, we compute the hybridization number for a number of tree pairs from both simulated data and biological datasets. The experiment was performed on a 3192 MHz Intel Xeon workstation.

4.1 Simulated Data

The simulated data is from Beiko and Hamilton [3]. The trees are generated as follows: a random tree is first selected, and then a small number of random rSPR operations are applied to obtain the second tree. We tested ten pairs of trees, each with 100 leaves and the number of rSPR operations is equal to ten. In Table 1, we give results of these ten datasets. It can be seen that program SPRDist is efficient in computing the hybridization number of these trees: the running time is usually less than one minute. The CPLEX solver gives faster results for more difficult cases. Also the preprocessing helps to reduce the running time. As a comparison, program *HybridNumber* runs for very long time for most datasets: for only one of the ten datasets, program *HybridNumber* finds the solution within one hour.

4.2 Biological data

To demonstrate that our method works for true biological data, we also test our method on the following biological data: tree pairs for a Poaceae dataset. The dataset is originally from the Grass Phylogeny Working Group [9]. The dataset contains sequences for six loci: internal transcribed spacer of ribosomal DNA (ITS); NADH dehydrogenase, subunit F (ndhF); phytochrome B (phyB); ribulose 1,5-biphosphate carboxylase/oxygenase, large subunit (rbcL); RNA polymerase II, subunit β'' (rpoC2); and granule bound starch synthase I (waxy). The Poaceae dataset was previously analyzed by Heiko Schmidt [17], who generated the inferred rooted binary trees for these loci. Bordewich, et al. [5] computed the minimum hybridization number for each of the fifteen pair of trees. Previously, we computed the exact rSPR distance for each pair of trees. To test how well our method performs on these biological trees, we compute the exact hybridization number for the same fifteen pairs of trees. See Table 2 for the results.

Table 1. Computing exact hybridization number on simulated data from Beiko and Hamilton (2006). For each pairs of trees, ten random rSPR operations are performed when the datasets are applied. h_S: the hybridization number computed by program *SPRDist*. Each pair is computed with CPLEX and GLPK. We also compare the program with or without divide-and-conquer preprocessing. The columns labeled as no prep. are for the results without performing preprocessing. Time is measured in seconds (s), hours (h) and days (d).

h_S	CPLEX	no prep.	GLPK	no prep.	HybridNumber
10	37 s	37 s	57 s	57 s	2 d 9 h
10	81 s	81 s	167 s	179 s	2 d 19 h
9	10 s	12 s	38 s	47 s	21 h 33 s
9	0 s	3 s	15 s	17 s	3205 s
10	15 s	16 s	66 s	82 s	2 d 6 h
9	0 s	5 s	1 s	26 s	2 d 20 h
10	13 s	13 s	53 s	58 s	2 d 7 h
10	18 s	24 s	318 s	379 s	2 d 8 h
10	16 s	376 s	72 s	146 s	2 d 5 h
10	3 s	10 s	29 s	77 s	2 d 9 h

Table 2. Performance of program SPRDist on fifteen pairs of trees for the Poaceae data. For comparison, we also list those of program *HybridNumber*. h_S: hybridization number from program SPRDist. rSPR: rSPR distance. h_{HN}: the minimum hybridization number computed in [5]. We list the running time of program *SPRDist* using either CPLEX and GLPK. We only give results for trees preprocessed with divide-and-conquer preprocessing. Time is measured in seconds (s) and hours (h).

	Data			SPRDist			HybridNumber	
Pair	1	2	# taxa	h_S(rSPR)	**CPLEX**	**GLPK**	h_{HN}	Time
1	ndhF	phyB	40	14 (12)	5 s	65 s	14	3 s
2	ndhF	rbcL	36	13 (10)	10 s	84 s	13	3 s
3	ndhF	rpoC2	34	12 (11)	7 s	77 s	12	6 s
4	ndhF	waxy	19	9 (7)	1 s	2 s	9	1 s
5	ndhF	ITS	46	19 (19)	**51** s	666 s	19	667 s
6	phyB	rbcL	21	4 (4)	0 s	1 s	4	1 s
7	phyB	rpoC2	21	7 (6)	3 s	2 s	7	1 s
8	phyB	waxy	14	3 (3)	1 s	1 s	3	1 s
9	phyB	ITS	30	8 (8)	1 s	2 s	8	1 s
10	rbcL	rpoC2	26	13 (11)	14 s	134 s	13	16 s
11	rbcL	waxy	12	7 (6)	1 s	1 s	7	1 s
12	rbcL	ITS	29	14 (13)	**80** s	1140 s	14	4 h 2716 s
13	rpoC2	waxy	10	1 (1)	0 s	0 s	1	1 s
14	rpoC2	ITS	31	15 (14)	**115** s	1469 s	15	7 h 776 s
15	waxy	ITS	15	8 (7)	1 s	9 s	8	2 s

The experimental results in Table 2 indicate that our program *SPRDist* is more efficient than program *HybridNumber* for more difficult datasets. In fact, our program only takes seconds or minutes to compute the *exact* hybridization number when CPLEX solver is used. The CPLEX solver usually leads to less running time than the GLPK solver, especially for more difficult input data. Even the GLPK solver can lead to faster running time than program *HybridNumber* for more difficult cases (pairs 12 and 14). On the other hand, program *HybridNumber* appears to perform better when the input trees allow significant reduction through preprocessing; when preprocessing is less effective, it usually performs poorly for larger trees. This can also be seen in Table 1. We also compare the hybridization number with the rSPR distance. It appears that the two values tend to be more different when the size of trees and the hybridization number grow.

The simulation results show that our program *SPRDist* is more scalable than program *HybridNumber*. Also, program *SPRDist* is more robust than program *HybridNumber*. With our new method, computing the hybridization number between two large and topologically far apart trees is still challenging, but feasible.

Constructing the consistent history. In addition to computing the hybridization number in this paper, our method also finds the corresponding maximum acyclic agreement forest. It is straightforward to apply the algorithm given in [18] to construct a most parsimonious phylogenetic history from the forest.

Acknowledgment. Research is supported by National Science Foundation [IIS-0803440] and the Research Foundation of University of Connecticut.

References

1. M. Baroni, S. Grunewald, V. Moulton, and C. Semple. Bounding the number of hybridization events for a consistent evolutionary history. *J. Math. Biol.*, 51:171–182, 2005.
2. M. Baroni, C. Semple, and M. Steel. A framework for representing reticulate evolution. *Annals of Combinatorics*, 8:391–408, 2004.
3. R. G. Beiko and N. Hamilton. Phylogenetic identification of lateral genetic transfer events. *BMC Evolutionary Biology*, 6:15, 2006.
4. M. Bonet, K. S. John, R. Mahindru, and N. Amenta. Approximating subtree distances between phylogenies. *J. of Comp. Biology*, 13:1419-1434, 2006.
5. M. Bordewich, S. Linz, K. S. John, and C. Semple. A reduction algorithm for computing the hybridization number of two trees. *Evolutionary Bioinformatics*, 3:86-98, 2007.
6. M. Bordewich, C. McCartin, and C. Semple. A 3-approximation algorithm for the subtree distance between phylogenies. *J. Discrete Algorithms*, 6:458–471, 2008.
7. M. Bordewich and C. Semple. On the computational complexity of the rooted subtree prune and regraft distance. *Annals of Combinatorics*, 8:409–423, 2004.
8. M. Bordewich and C. Semple. Computing the minimum number of hybridization events for a consistent evolutionary history. *Discrete Applied Mathematics*, 155:914–928, 2007.

9. Grass Phylogeny Working Group. Phylogeny and subfamilial classification of the grasses (poaceae). *Ann. Mo. Bot. Gard.*, 88:373–457, 2001.
10. J. Hein, T. Jiang, L. Wang, and K. Zhang. On the complexity of comparing evolutionary trees. *Discrete Appl. Math*, 71:153–169, 1996.
11. D. Huson and D. Bryant. Application of phylogenetic networks in evolutionary studies. *Molecular Biology and Evolution*, 23:254–267, 2006.
12. C. R. Linder, B. M. E. Moret, L. Nakhleh, and T. Warnow. Network (reticulate) evolution: biology, models, and algorithms, 2004.
13. S. Linz. Personal communications.
14. S. Linz and C. Semple. Hybridization in nonbinary trees. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 6:30–45, 2009.
15. L. Nakhleh. Evolutionary phylogenetic networks: models and issues. In L. Heath and N. Ramakrishnan, editors, *The Problem Solving Handbook for Computational Biology and Bioinformatics*. Springer, 2010. In press.
16. E. M. Rodrigues, M. F. Sagot, and Y. Wakabayashi. Some approximation results for the maximum agreement forest problem. In *Proceedings of RANDOM-APPROX 2001*, page 159-169, 2001.
17. H. Schmidt. *Phylogenetic trees from large datasets*. PhD thesis, Heinrich-Heine-Universitat, Dusseldorf, 2003.
18. C. Semple. Hybridization networks. In O. Gascuel and M. Steel, editors, *Reconstructing Evolution: New Mathematical and Computational Advances*, pages 277–309. Oxford, 2007.
19. R. Tarjan. Enumeration of the elementary circuits of a directed graph. *SIAM J. on Computing*, 2:211-216, 1973.
20. Y. Wu. A practical method for exact computation of subtree prune and regraft distance. *Bioinformatics*, 25:190–196, 2009.