# Fast Construction of Near Parsimonious Hybridization Networks for Multiple Phylogenetic Trees

Sajad Mirzaei and Yufeng Wu

**Abstract**—Hybridization networks represent plausible evolutionary histories of species that are affected by reticulate evolutionary processes. An established computational problem on hybridization networks is constructing the most parsimonious hybridization network such that each of the given phylogenetic trees (called gene trees) is "displayed" in the network. There have been several previous approaches, including an exact method and several heuristics, for this NP-hard problem. However, the exact method is only applicable to a limited range of data, and heuristic methods can be less accurate and also slow sometimes. In this paper, we develop a new algorithm for constructing near parsimonious networks for multiple binary gene trees. This method is more efficient for large numbers of gene trees than previous heuristics. This new method also produces more parsimonious results on many simulated datasets as well as a real biological dataset than a previous method. We also show that our method produces topologically more accurate networks for many datasets.

**Index Terms**—Phylogenetics, Reticulate evolution, Algorithm, Combinatorial optimization

## 1 Introduction

A recent trend in phylogenetics is incorporating reticulate evolutionary processes (such as horizontal gene transfer or hybrid speciation) in phylogenetic models. This leads to new phylogenetic models for reticulate evolution. In this paper, we focus on the so-called "hybridization network" model [9], [13]. Suppose we are given a set of phylogenetic trees $T_1, \ldots, T_K$. Each $T_i$ represents the evolutionary history of a gene and thus is called a gene tree. Each gene tree is inferred from DNA sequences collected at the gene. It is well known that these $T_i$'s may not have the same topology. One possible cause is reticulate evolution [10], [13]. If this is the case, then we can no longer represent the evolutionary history with a tree model. Instead, we need a model that allows reticulate evolution. Hybridization network model is one such model. A hybridization network (often referred to as a network in this paper) is a directed acyclic graph that "displays" each of the gene trees. We provide more precise definitions in Section 1.1. We refer to Figure 1 for an illustration of a hybridization network. Our goal is inferring hybridization networks from the given gene trees. Like most current approaches for hybridization network inference, we follow the parsimony principle [9], [10]. That is, our goal is to find the hybridization networks with the fewest reticulation events. Note that we require that the full gene trees are contained in the network. There are approaches for building hybridization networks from parts of gene trees (e.g. [14], [15]).

The problem of constructing the most parsimonious hybridization network is NP-hard even for two gene trees [5]. Nevertheless, there are exact and FPT methods for building hybridization networks that allow two input gene trees (e.g. [2], [4], [13], [20], and also see [17]). For multiple gene trees, there are currently an exact [19] and an FPT method [16], as well as several heuristics [7], [12], [18]. However, the exact method can only construct the most parsimonious networks for relatively small data. The existing heuristics can often work with larger data, but they may produce less accurate hybridization networks and can also be slow. For example, the method $PIRN$ in [18] becomes slow when the number of gene trees increases.

In this paper, we develop an algorithm (called $PIRN_S$) for inferring near parsimonious hybridization networks from multiple gene trees. $PIRN_S$ takes a set of rooted binary gene trees as input and constructs a hybridization network that displays each of the gene trees. The main advantage of our work is that $PIRN_S$ produces more parsimonious hybridization networks than $PIRN$ in many datasets and also scales better than $PIRN$ in the number of gene trees. $PIRN_S$ also produces networks that are usually topologically more accurate than $PIRN$ especially when networks are more complex. On the other hand, $PIRN_S$ scales less well in the number of taxa than in the number of gene trees. Our experiments show that $PIRN_S$ works well for gene trees with up to

• *S. Mirzaei and Y. Wu are with Department of Computer Science and Engineering, University of Connecticut, Storrs, CT 06269, U.S.A. E-mail: {sajad,ywu}@engr.uconn.edu*

*Manuscript received xx xx, 2014; revised xx xx, 2015.*

20 taxa. We show that $PIRN_S$ performs well on simulated datasets and real biological datasets which one of them has been analyzed by several previous approaches [7], [12], [18]. Here, we only compare with $PIRN$ because the method in [7] does not build networks for multiple gene trees. Also, the method in [12] is faster but according to the results in [12], it does not build more parsimonious networks than $PIRN$ in some simulated datasets and a biological dataset. Also, their method is on paper and they have not provided their implementation. We compare with their results on a real biological dataset.

### 1.1 Definitions

In this paper, we assume that a gene tree is a rooted tree and leaf-labeled by a set of taxa. In a tree, in-degrees of all vertices are one except the root, while out-degrees are at least two for all vertices except leaves with out-degree zero. For a binary tree, out-degrees of internal nodes must be two. In this paper, we assume that the gene trees are binary, and the set of gene trees $T_i$ is the input. We assume that the root of each gene tree $T_i$ is attached to an outgroup species $o$. More information about this can be found in [12].

We use the same definition of hybridization network as [18], [19]. A hybridization network is a directed acyclic graph. It contains a vertex set $V$ including leaf nodes and edge set $E$. $V$ can be divided to $V_T$ (called tree nodes) and $V_H$ (called hybridization nodes). $E$ can also be divided to $E_T$ (called tree edges) and $E_H$ (called hybridization edges). Also,

1) Except the root, which has no incoming and two outgoing edges, no nodes with total (in and out) degree of two are allowed, and each node must have at least one incoming edge.
2) $V_H$ contains nodes whose in-degrees are two or more. $V_T$ contains nodes whose in-degrees are one.
3) $E_H$ contains edges that go into hybridization nodes. $E_T$ contains edges that go into tree nodes.
4) A node is labeled by some taxon iff its out-degree is zero.

In addition there is one more restriction:

$R_1$ For a network $\mathcal{N}$, when only one of the incoming edges of each hybridization node is kept and the rest are deleted, we always derive a leaf labeled tree $T'$.

Hybridization network can be viewed as a compact representation of a set of gene trees. The definition of "display" is critical. Suppose we keep only a single edge at each hybridization node in a hybridization network $\mathcal{N}$ and obtain a tree $T'$. Further suppose that we remove non-labeled leaves and contract edges to remove nodes with degree two (called cleanup) in $T'$. We will derive a phylogenetic tree $T$ with the same set of taxa as in network $\mathcal{N}$. We say $T$ is *displayed* in network $\mathcal{N}$. See Figure 1 for an illustration.

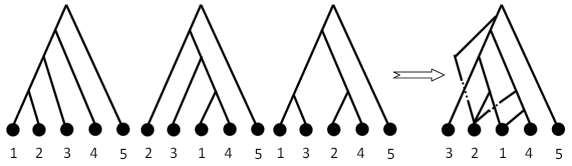



Fig. 1. An illustration of a hybridization network with three reticulation events for three trees. Each tree is displayed in the network: the tree can be obtained by keeping one incoming edge at each hybridization node.

Suppose we have a set of taxa $\hat{S}$ for all the gene trees. For example, $\hat{S} = \{1, 2, 3, 4, 5\}$ for input trees in Figure 1. A *subtree* of an input tree $T$ for a subset of taxa $S \subseteq \hat{S}$ is a tree which can be derived from the original tree by removing the taxa not in $S$ from $T$ (and then performing cleanup afterwards). The derived subtree (denoted as $T(S)$) only contains the taxa in $S$. For example, the tree containing three nodes including one node as a root and two leaves 1 and 2, is a subtree for the subset $S = \{1, 2\}$ for each gene tree in Figure 1.

For a hybridization node, we define the *hybridization number* as its in-degree minus one. For a hybridization network $\mathcal{N}$, we define the hybridization number ($H_{\mathcal{N}}$) as the summation of hybridization numbers of each of its hybridization nodes. For example, in Figure 1, the hybridization node 2 has three incoming edges, and the hybridization node 1 has two incoming edges. Thus, $H_{\mathcal{N}} = (3-1) + (2-1) = 3$. This is the same as the hybridization number defined in [13]. Now we formulate the central problem in this paper.

**The most parsimonious hybridization network problem**. Given $K$ rooted and binary gene trees $T_1, T_2, \ldots T_K$ (with the same $n$ taxa), construct the hybridization network $\mathcal{N}_{min}$ such that (i) each gene tree $T_i$ is displayed in $\mathcal{N}_{min}$, and (ii) $H_{\mathcal{N}_{min}}$ is *minimized* among all possible such networks. We call $H_{\mathcal{N}_{min}}$ the hybridization number of $T_1, \ldots, T_K$.

Constructing parsimonious hybridization networks or computing the hybridization number for a set of $K$ gene trees is computationally challenging. Even the two-gene-tree (i.e. $K = 2$) case is known to be NP-complete [5]. For the two-gene-tree case of the hybridization network problem, there are several exact methods [2], [4], [20]. Although the worst case running time of these practical methods is exponential, these methods may work reasonably well in practice. The case of three or more gene trees ($K \geq 3$) is more difficult. There are currently an exact method and a few heuristic methods for either estimating the hybridization number or reconstructing near optimal networks for trees $T_1, \ldots, T_K$ when $K \geq 3$ [7], [12], [18]. The exact method in [19] uses a backward in time approach. This method can find optimal hybridization networks with hybridization number up to six. It becomes impractical when the hybridization number

is larger than six. The heuristic (called $PIRN$) in [18] (and [12]) is based on the following idea: make the current hybridization network $\mathcal{N}$ equal to one gene tree (say $T_1$); for each additional tree $T_i$, modify $\mathcal{N}$ by adding additional hybridization nodes to $\mathcal{N}$ if needed to make $T_i$ displayed in $\mathcal{N}$. Usually this implies that we need to add as few hybridization nodes in $\mathcal{N}$ as possible in order to reconstruct a more parsimonious network. Since it is unclear in which order $T_i$ should be "added" to $\mathcal{N}$, $PIRN$ tries all possible order of gene trees. So, $PIRN$ works well when $K$ is small but becomes slow when $K$ is larger. In this paper, we use a different approach for building near parsimonious hybridization networks that is much faster than $PIRN$ when $K$ is larger.

## 2 METHOD

### 2.1 The high-level approach

Recall that $PIRN$ [18] builds hybridization networks incrementally: at each step, the network is modified to ensure a new gene tree is displayed in the network. Our new method $PIRN_S$ is also an incremental approach. The difference from $PIRN$ is that, instead of "adding" a gene tree each time, $PIRN_S$ adds a *taxon* each time. That is, we start with the trivial networks for subsets of two taxa; then we add other taxa gradually to the networks; the procedure stops when all taxa are in the network. We now present more details of this approach.

We denote a subset of taxa as $S$. For one $S$, we maintain one or a small number of (say $c$) hybridization networks. We denote one such network as $\mathcal{N}(S)$. $\mathcal{N}(S)$ has the taxa in $S$ as leaves. Usually there are many choices for $\mathcal{N}(S)$. We denote the subtree of an input gene tree $T_i$ for $S$ as $T_i(S)$. We first require that each $T_i(S)$ is displayed in $\mathcal{N}(S)$ for each $1 \leq i \leq K$. Since our goal is finding parsimonious hybridization networks, $\mathcal{N}(S)$ should have as small hybridization number as possible for the set of subtrees $T_i(S)$. For example, Figure 2(a) shows the subtrees and a hybridization network for subset $S = \{2, 3, 4, 5\}$ of the original set of taxa $\hat{S} = \{1, 2, 3, 4, 5\}$, where the gene trees are displayed in Figure 1. Obviously, $\mathcal{N}(\hat{S})$ displays all gene trees $T_i$. This constructed $\mathcal{N}(\hat{S})$ is the result of our method.

Suppose we have constructed a hybridization network $\mathcal{N}(S)$ for a subset of taxa $S$. We let $s$ be a taxon not in $S$, and denote $S' = S \cup \{s\}$. We can construct $\mathcal{N}(S')$ by modifying $\mathcal{N}(S)$. $H_{\mathcal{N}(S')}$ is equal to $H_{\mathcal{N}(S)}$ plus the cost of adding the new taxon $s$ to $\mathcal{N}(S)$ such that $\mathcal{N}(S')$ displays the subtrees $T_i(S')$. We define the cost $C(\mathcal{N}(S), s)$ of adding a taxon $s$ to a network $\mathcal{N}(S)$ as the number of new hybridization edges that we have to add to $\mathcal{N}(S)$. Therefore, in order to reduce $H_{\mathcal{N}(S')}$, we need to make $C(\mathcal{N}(S), s)$ as small as possible.

The high-level procedure of our method $PIRN_S$ is as follows. Here, for simplicity, suppose we only store a single best hybridization network $\mathcal{N}(S)$ for each subset $S$ of taxa. It is straightforward to modify this procedure to store $c$ networks for each $S$.

---

Initialize $\mathcal{N}(S_0)$ of all possible subsets $S_0$ of taxa of size 2.
**for** $2 \leq k \leq n-1$ **do**
  **for all** saved networks $\mathcal{N}(S)$ for subsets (say $S$) of size $k$ **do**
    **for all** taxa that are not present in $S$ **do**
      a. Add the taxon to the network and reconstruct a network $\mathcal{N}'(S')$ of subsets $S'$ of size $k+1$
      b. Store $\mathcal{N}'(S')$ as $\mathcal{N}(S')$ for $S'$ if $H_{\mathcal{N}'(S')} < H_{\mathcal{N}(S')}$.
    **end for**
  **end for**
**end for**
**return** $\mathcal{N}(\hat{S})$ as the network for the whole set of taxa $\hat{S}$.

---

Note that when $|S_0| = 2$ (i.e. only two taxa), each $T_i(S_0)$ has the same cherry topology for all $1 \leq i \leq K$. Thus, $\mathcal{N}(S_0)$ is just the cherry. The main step is how to grow a network by adding a new taxa $s$. We will explain this procedure in Section 2.2. In Section 2.3, we will provide some analysis for the algorithm.

### 2.2 Adding a taxon

The cost $C(\mathcal{N}(S), s)$ of adding a new taxon $s$ is based on the number of new hybridization edges that we must add to the network $\mathcal{N}(S)$ to make it display the subtrees $T_i(S')$ (where $S' = S \cup \{s\}$). Recall that $\mathcal{N}(S)$ is the best network found for $S$. $\mathcal{N}(S)$ displays all the subtrees $T_i(S)$ of subset $S$. The key is finding what parts of $\mathcal{N}(S)$ we need to change such that each subtree $T_i(S')$ is displayed in the changed network $\mathcal{N}(S)$, and fewest new reticulation events are added to $\mathcal{N}(S)$. Our approach aims to find a minimal set of *edges* $E(s)$ of $\mathcal{N}(S)$, and then attach the new taxon $s$ to each edge $e \in E(s)$. Here, we attach $s$ to $e$ by creating a new node $v(e, s)$ that subdivides $e$ into two parts and connecting $s$ to $v(e, s)$. We refer to Figure 2(c) for an illustration. The network $\mathcal{N}(\hat{S})$ in Figure 2(c) is obtained from $\mathcal{N}(S)$ in Figure 2(b) by attaching taxon 1 to edges $(2, j)$ and $(4, k)$.

To find the best edge set $E(s)$, we examine input subtree $T_i(S)$ and $T_i(S')$ for each $1 \leq i \leq K$. Let $T$ be one of input trees, and $T(S)$ and $T(S')$ be the subtrees derived from $T$ with taxa $S$ and $S'$ respectively. From restriction $R_1$ in Section 1.1, we can derive a tree $T'(S)$ from network $\mathcal{N}(S)$ (but without doing the cleanup step). Note that there can be multiple valid $T'(S)$ for some $T$ and $S$. In our current implementation, we choose the particular $T'(S)$ that is obtained during the

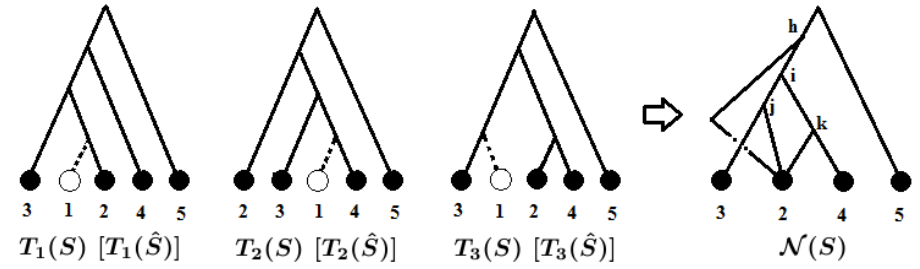
construction of $\mathcal{N}(S)$. Comparing $T'(S)$ to $T(S')$, we can find proper edges to attach the new taxon $s$ such that $T'(S')$ and $T(S')$ become topologically equivalent after cleanup. The process of finding candidate edges and subdividing some of them is the most critical part of our method. We illustrate this procedure using the following example.

We refer to Figure 2 where $S = \{2,3,4,5\}$, and $\hat{S} = \{1,2,3,4,5\}$. One can see all subtrees $T_i(S)$ (ignoring the dotted edges) and $T_i(\hat{S})$ (keeping the dotted edges) in Figure 2(a) (on the left) and the network $\mathcal{N}(S)$ (on the right). Figure 2(b) shows how each $T_i(S)$ is displayed in $\mathcal{N}(S)$: the dotted edges are those to remove from $\mathcal{N}(S)$ to display each $T_i(S)$. Tree $T_i(\hat{S})$ has an extra taxon (i.e. 1) not in $T'_i(S)$ and also one extra edge that connects the taxon 1 to the rest of it. Now we need to find a position in network $\mathcal{N}(S)$ to add the taxon 1 to. The proper position for the taxon 1 is where the taxon 1 has the same sibling in the changed network as it has in tree $T_i(\hat{S})$. For example, in Figure 2(a) in $T_2(\hat{S})$ the taxon 1 has the taxon 4 as sibling, so the taxon 1 should have the taxon 4 as sibling when we change $T'_2(S)$ in Figure 2(b). Thus, it can be added somewhere between node 4 and node $i$ in $\mathcal{N}(S)$ in Figure 2(b). One should note that the taxon 1 can also be added to the edge $\{2,k\}$ (the edge connecting the two nodes 2 and $k$) even if it is not part of the tree derived from the network. This is because if we subdivide the edge $\{2,k\}$ and add the taxon 1 to it, the taxon 1 can still have the taxon 4 as sibling, so the edge $\{2,k\}$ is also a valid choice.

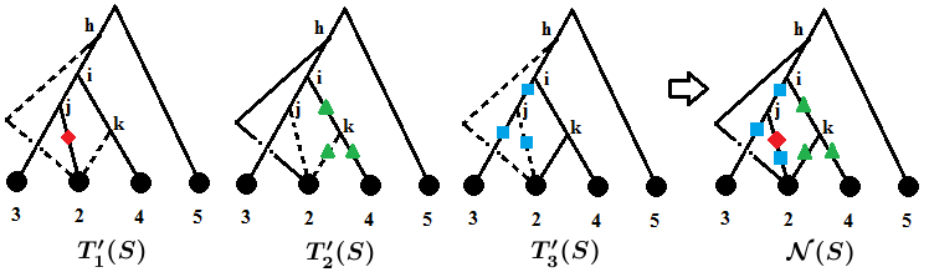
For each input subtree $T_i(S)$, we identify the set of edges $E_i(s)$ in $\mathcal{N}(S)$ that the new taxon $s$ can be added to, which we call the candidate edges. We say the subtree $T_i(S)$ is *covered* if at least one edge in $E_i(s)$ is chosen to be subdivided. Clearly each $T_i(S)$ needs to be covered. Note that the more candidate edges we subdivide and attach to $s$, the higher the hybridization number of the new network will be. This is because $s$ will attach to each subdivided edge and increases the hybridization number by one. Thus, we need to choose the *minimum* number of candidate edges to attach to $s$ such that $T_i(S)$ is covered for $1 \leq i \leq K$. Also note that one edge in $\mathcal{N}(S)$ can appear in several $E_i(s)$. To reduce the number of edges to subdivide, we want to choose edges that appear in multiple $E_i(s)$. This is precisely the classic hitting set problem. For example, in Figure 2(b), the edge $(2,j)$ is in both $E_1(s)$ and $E_3(s)$ where $s = 1$. Thus, we pick the edge $(2,j)$ to attach to the taxon 1. This edge $(2,j)$ allows us to display both $T_1(\hat{S})$ and $T_3(\hat{S})$ by subdividing only one edge. We then pick another edge in $E_2(s)$ (say $(4,k)$ as in Figure 2(c)) to display $T_2(\hat{S})$. This allows us to use only two new hybridization events when adding the taxon 1. One can verify that some other choices of edges to subdivide will need three new hybridization events to add the taxon 1. Thus, the choices made in Figure 2(c) reduce the number of new reticulation events. This part of our algorithm is somewhat similar to the "Minimum Attachment Problem" mentioned in chapter 8 of [9], which is for clusters.

### 2.3 Analysis of the bottom up approach

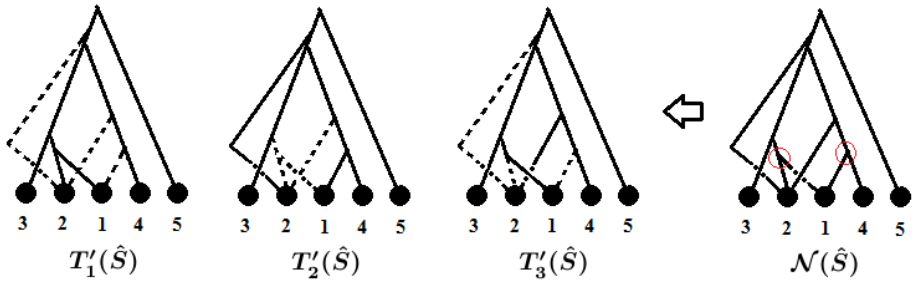
Our method considers all subsets $S$ of $n$ taxa with at least two taxa. For each $S$, we construct a network for $S$. There are $O(2^n)$ subsets of $n$ taxa. When $n$ increases (to say 30), our method becomes very slow since the number of subsets becomes very large. On the other hand, our method scales much better with respect to $K$, the number of gene trees. $K$ has no effect on the number of subsets. $K$ affects the time for inserting a new taxon $s$ to a network $\mathcal{N}(S)$ for a subset $S$. There are two aspects: (i) time to find candidate edges in $\mathcal{N}(S)$ for each subtree $T_i(S)$, and (ii) time to select which candidate edges to connect $s$ such that all subtrees are covered. The time for finding candidate edges for one subtree is linear to the number of edges of the network $\mathcal{N}(S)$. In theory, $\mathcal{N}(S)$ may have many hybridization edges. But in practice, $\mathcal{N}(S)$ is usually a simple network and it is not very far from being a tree, so it is often reasonable to assume that the number of edges in $\mathcal{N}(S)$ is $O(n)$. Thus, the time of part (i) is $O(nK)$. The problem for selecting which edges to subdivide is equivalent to solving the hitting set problem, which is well known to be NP complete. Our current implementation takes a simple enumeration-based approach to find the smallest set of edges to subdivide. That is, we enumerate all subsets of candidate edges of the increasing cardinality to find the candidate edges that cover all subtrees. This is because it is critical to obtain optimal solutions here in order to construct near parsimonious networks. Since there are $O(n)$ candidate edges for each of the $K$ subtrees, our simple enumeration may need to enumerate $O(n^K)$ subsets of these candidate edges. This is because we need to choose one candidate edge for each subtree. This seems to suggest that our method will also become slow when $K$ increases. In practice, however, our implementation appears to scale well with $K$. There are several reasons. First, usually there are only a small number of candidate edges for each subtree. We also remove redundant candidate edges that cover the exact same subtrees. That is, if there are two candidate edges $x$ and $y$ that are candidates for the same subtree(s), so we can remove one (say $y$) from consideration. At last, usually the number of edges to choose is rather small and so we don't need to enumerate subsets of large size. In fact, if we need to choose a large number of candidate edges to subdivide, this usually indicates that the network will not be near parsimonious and should not be constructed at all. Therefore, in practice, we may assume that the number of candidate edges to choose is bounded by a constant $R$. Then the time for




(a) A hybridization network subset $S = \{2, 3, 4, 5\}$




(b) Finding candidate edges for adding the taxon 1




(c) Solution for subset $\hat{S} = \{1, 2, 3, 4, 5\}$

Fig. 2. (a): Denote the subtrees $T_1(S)$, $T_2(S)$, and $T_3(S)$ for subset $S = \{2, 3, 4, 5\}$ from original trees $T_1(\hat{S})$, $T_2(\hat{S})$, and $T_3(\hat{S})$: $T_i(S)$ is obtained by removing dotted edges from $T_i(\hat{S})$. The rightmost part shows a network $\mathcal{N}(S)$ for the subset $S$. (b): How subtrees $T_1(S)$, $T_2(S)$, and $T_3(S)$ are displayed in the network $\mathcal{N}(S)$ from the restriction $R_1$: each $T'_i(S)$ shows how $T_i(S)$ is displayed in the network without cleanup. We mark the edges of $\mathcal{N}(S)$ to which taxon 1 can be added. $T'_1(S)$ and $T'_3(S)$ share edge $(2, j)$ (marked by a diamond and a square) and we can use any of the edges $(4, k)$, $(2, k)$, or $(k, i)$ (marked by triangles) for $T'_2(S)$ (here we choose $(4, k)$). (c): Adding the taxon 1 by subdividing candidate edges $(2, j)$ and $(4, k)$ to build the network $\mathcal{N}(\hat{S})$ for set $\hat{S} = \{1, 2, 3, 4, 5\}$.

finding no more than $R$ candidate edges is $O(Kn^R)$. Under this assumption, the running time of $PIRN_S$ is polynomial in $K$.

We note that when $n$ increases, it will be slow to enumerate all subsets of $n$ taxa. There are various heuristics for speedup. We may drop some subsets that do not have good networks when compared to other subsets with the same size. Our implementation uses another way to drop some subsets. We use parameter $a$ to determine what subsets to drop. As described in our high level algorithm, we iterate through all possible subsets with size 2 to $n - 1$. Among all subsets of size $k$, if the best solution has the hybridization number $r$, we remove all networks with hybridization number greater than $r + a$ for speedup because these networks may not lead to a good final network. Also, there might be more than one parsimonious network for each subset which will make the program even slower. Here we use parameter $c$ to limit the number of parsimonious network to save for each subset. Note that, however, these speedup heuristics may lead to less parsimonious networks. So there is a trade-off between speed and accuracy.

## 3 Results

We have implemented our method $PIRN_S$ (where "S" stands for subset) for building near-parsimonious networks. $PIRN_S$ is available for download from: http://www.engr.uconn.edu/~ywu/. $PIRN_S$ is written in Java. By default, we set the parameters $a$ and

$c$ to be both three. These default parameters are used in our simulation to evaluate $PIRN_S$ on simulation and biological data.

### 3.1 Simulation Data

We use the same simulation method in [18]. This simulation is a two step approach. First, we simulate a hybridization network, and then generate a fixed number of trees displayed in the network. For a number of taxa $n$, the simulation starts with isolated lineages and simulate reticulation backwards in time. At each step, there are two possible events: (a) lineage merging, which occurs at rate 1; (b) lineage splitting, which occurs at rate $r$. It chooses the next event according to relative probabilities of all feasible events. Lineage merging generates speciation events, while lineage splitting generates reticulation events. The parameter $r$ dictates the level of reticulation in the simulated network: larger $r$ will lead to more reticulation events in simulation [18]. We refer to [18] for more details.

To test the performance of $PIRN_S$, we compare $PIRN_S$ with program $PIRN$ [18]. We test our program on datasets with various settings of parameters $K$ (numbers of trees), $n$ (number of taxa), and $r$ (level of reticulation). For each setting we simulate 100 datasets. We can not completely compare with the exact method in [19] because the exact method is slow for many datasets that we test here. We were only able to compare with the exact method of $PIRN$ for $n = 5$, $r = 3.0$ for 100 datasets with two different values of $K$ which $PIRN_S$ found an optimal network for 96% of datasets with $K = 5$ and 81% of datasets with $K = 10$. There is a new unpublished exact method called *Hybroscale* [1], which scales better than the exact method of $PIRN$ . This program did not finish for some cases after weeks for settings of $n = 10$, $r = 3.0$, and $K >= 4$. $PIRN_S$ was able to find optimal networks for 100% of datasets For settings $n = 10$, $r = 1.0$, and $K = 3$–5, and for 99% of datasets for setting $n = 10$, $r = 3.0$, and $K = 3$. Further comparisons in this paper are made with the heuristic method of $PIRN$ . Also recall that the method in [12] is faster but does not build more parsimonious networks than $PIRN$ [12].

We first compare the parsimony of the constructed networks from the two approaches. By parsimony, we refer to the hybridization number of a constructed network and the one with the smaller hybridization number is more parsimonious and better. For each program, we report the percentage of networks with smaller hybridization number than that of the other one among 100 datasets for each setting. As shown in Figure 3, $PIRN_S$ outperforms $PIRN$ in most tests especially when the numbers of trees $K$ is small. In a few settings (for example, $n = 15$, $r = 5.0$) when $K$ increases, $PIRN$ produces better networks. However,

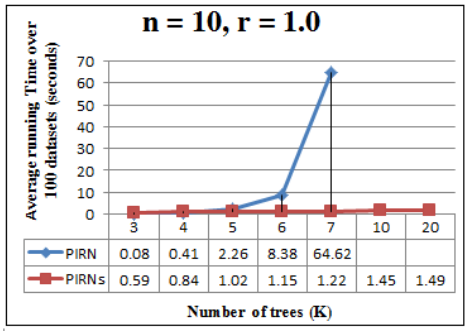



Fig. 4. Average running time of $PIRN_S$ and $PIRN$ over 100 datasets with fixed $n$ and $r$ by increasing $K$. X axis: number of trees $K$. Y axis: time (in seconds). $PIRN$ is unable to finish for $K \geq 8$.

$PIRN$ cannot scale to larger numbers of trees, and becomes very slow. Also, when the data becomes more complicated (i.e. reticulation level $r$ increases), $PIRN_S$ performs better. Simulations show that when the number of taxa $n$ increases, the hybridization number gap between $PIRN_S$ and $PIRN$ becomes larger and $PIRN_S$ usually outperforms $PIRN$.

TABLE 1
The topological accuracy of the networks for $n = 10$, and different values of $r$ and $K$. We report the average distance and the percentage of more accurate networks for each method over 100 datasets.

| $n = 10$ | | $r = 1.0$ | | $r = 3.0$ | | $r = 5.0$ | |
|---|---|---|---|---|---|---|---|
| | | PIRN | $PIRN_S$ | PIRN | $PIRN_S$ | PIRN | $PIRN_S$ |
| $K = 3$ | Avg Distance | 14.72 | 14.24 | 23.04 | 22.57 | 20.32 | 16.37 |
| | % of better solutions | 11 | 30 | 23 | 48 | 17 | 72 |
| $K = 4$ | Avg Distance | 15.65 | 15.37 | 23.25 | 23.01 | 21.29 | 15.05 |
| | % of better solutions | 15 | 27 | 32 | 40 | 9 | 88 |
| $K = 5$ | Avg Distance | 16.91 | 16.01 | 23.02 | 23.39 | 21.98 | 15.22 |
| | % of better solutions | 12 | 46 | 45 | 31 | 14 | 82 |

We also compare the running time of the two approaches. Increasing the numbers of trees has a significant effect on $PIRN$'s running time as expected, but does not have significant effect on the running time of $PIRN_S$. $PIRN$ becomes impractical when the number of trees reaches 8 or more, while $PIRN_S$ can scale to larger numbers of trees. Figure 4 shows the average running time for various numbers of trees $K$. In this simulation, we fixed $n = 10$ and $r = 1.0$. $PIRN$ has many unfinished simulations for cases $K \geq 8$. So we do not report its running time for these cases. Overall, our method $PIRN_S$ is much faster than $PIRN$.

**Topological accuracy**. While parsimony has been the main evaluation criterion for evaluating hybridization networks in the literature, evaluating the topological accuracy of inferred networks is also desirable. There are several known metrics for comparing hybridization networks from restricted classes [6], [11]. We measure the topological accuracy of the inferred networks by $PIRN_S$ using the metric distance defined

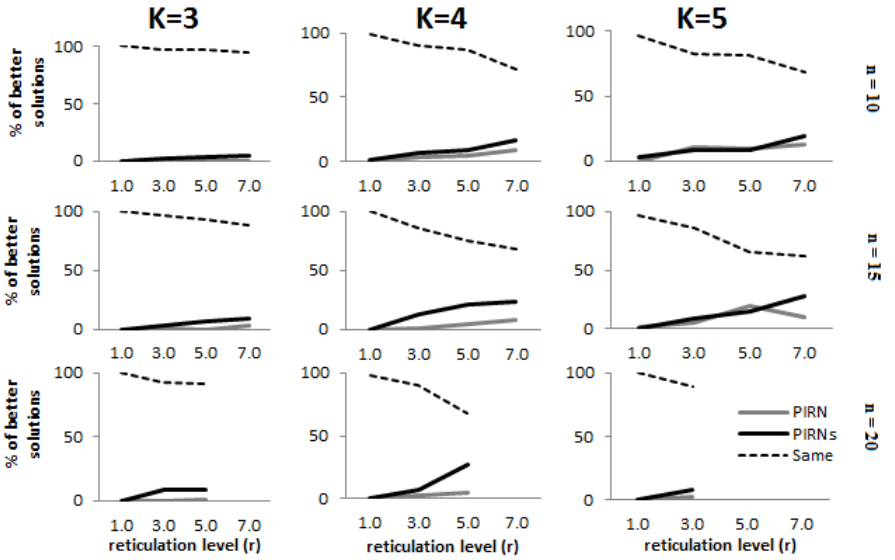



Fig. 3. Comparing parsimony of the networks for various $n$, $K$ and $r$. The vertical axis shows the percentage of datasets for which a method has a better network (more parsimonious) when compared to the other method. Horizontal axis is the reticulation level $r$. The dotted line shows the percentage of data that both methods have the same hybridization number.

in [6] called generalized Robinson-Foulds distance. Generally it defines a cluster set for each node in the network which contains its descendants leaves, then the total distance is the difference between the number of clusters that each network covers. We measure this distance between our output network and the original simulated network that we obtained our input trees from. Table 1 shows this distance for different values of $r$ and $K$ for both methods. $PIRN_S$ works better in terms of clusters for most of the settings especially when the reticulation level is higher.

### 3.2 Biological Data

To evaluate our program for real biological data, we use the Poaceae dataset which is originally from the Grass Phylogeny Working Group [8]. 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 $\beta''$ (rpoC2); and granule bound starch synthase I (waxy). Pairwise comparison of gene trees for these loci were performed in [4], [20]. In [18], a hybridization network with hybridization number of 13 was found for five gene trees (see the Appendix of [18] for these trees). In [12] they found a network with hybridization number of 14.

As shown in Table 2, $PIRN_S$ and $PIRN$ find networks with the same hybridization number (and also optimal based on lower bounds given in [18]) for two datasets with three gene trees. For the five tree case, however, $PIRN_S$ finds a network with hybridization number of 12 for five trees (one less than the one found by $PIRN$), and also runs much faster than $PIRN$ on these five gene trees. Figure 5 shows the found network for these five trees. We note that a lower bound of 11 is given for these five trees in [18]. Thus, although we obtain a more parsimonious network here, it is still unknown whether this is the most parsimonious network for this data.

We also used another biological dataset in [3]. It contains sequences for three loci: chloroplast trnL/F intron; nuclear rDNA internal transcribed spacer (ITS); and nuclear rDNA external transcribed spacer (ETS). Both $PIRN$ and $PIRN_S$ found 9 hybridization for this dataset. There is one biologically meaningful hybrid in this dataset named Mimulus evanescens. This is a hybrid between Mimulus latidens and Mimulus breviflorus, which was among those that $PIRN_S$ found, but $PIRN$ found this hybrid between Mimulus latidens and another hybrid node.

## 4 Conclusions and Discussions

In this paper, we developed a new method for constructing parsimonious hybridization networks from multiple gene trees. Constructing parsimonious hybridization networks from multiple gene trees is computationally challenging. As a heuristic, our method takes the natural incremental approach for building networks as in previous approaches [18], [12]. The key contribution of our work is showing that building networks by incrementally adding taxa performs well. In particular, our method scales better in the number of gene trees than existing methods. It also provides more parsimonious networks than previous methods

TABLE 2
Results for biological data. Hybridization number and running time for the grass data from $PIRN_S$ and $PIRN$. $n$: number of taxa. $H_{\mathcal{N}}$: the hybridization number of the network.

| Trees | n | $PIRN$ ($H_{\mathcal{N}}$) | $PIRN_S$ ($H_{\mathcal{N}}$) | $PIRN$ (Time) | $PIRN_S$ (Time) |
|---|---|---|---|---|---|
| rpoC2, waxy, ITS | 10 | 7 | 7 | 1s | 4s |
| ndhF, phyB, rbcL | 21 | 8 | 8 | 1s | 99s |
| ndhF, phyB, rbcL, rpoC2, ITS | 14 | 13 | **12** | 26m,38s | 117s |

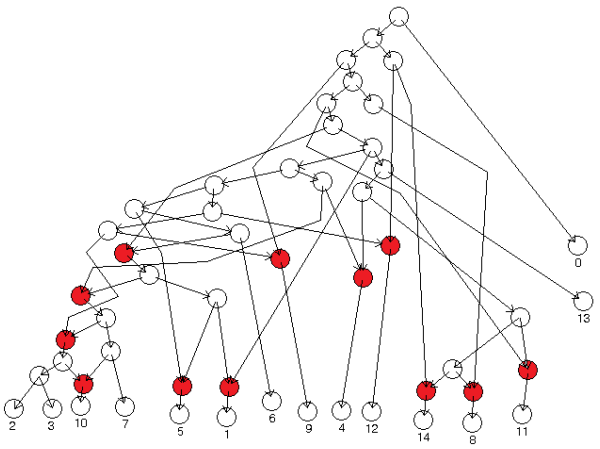



Fig. 5. A hybridization network for the five trees of a grass dataset found by program $PIRN_S$. Filled circles represent hybridization nodes.

in most of our experiments especially for real biological data. It also provides topologically more accurate networks than an existing method.

We note that much more research is still needed on the general subject of hybridization network reconstruction. While the parsimonious hybridization network formulation captures some key aspects of reticulate evolution, extensions and revisions may be needed to address the complexity when applying this formulation to real biological data. For example, we assume the (binary) gene trees are given in this paper, while in practice there is usually uncertainty in the gene trees. Future research will be focused on developing methods for biologically more realistic models and also on finding more applications to real biological data.

## ACKNOWLEDGMENTS

This work is partly supported by U.S. National Science Foundation grants IIS-0953563 and CCF-1116175.

## REFERENCES

[1] B. Albrecht. Computing Hybridization Networks for Multiple Rooted Binary Phylogenetic Trees by Maximum Acyclic Agreement Forests. *arXiv*:1408.3044, 2014.

[2] B. Albrecht, C. Scornavacca, A. Cenci, and D. H. Huson. Fast computation of minimum hybridization networks. *Bioinformatics*, 28:191–197, 2012.

[3] P. M. Beardsley, S. E. Schoenig, J. B. Whittall and R. G. Olmstead. Patterns of evolution in western North American Mimulus (Phrymaceae). *American Journal of Botany*, 91: 474–489, 2004.

[4] 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.

[5] M. Bordewich and C. Semple. Computing the minimum number of hybridization events for a consistent evolutionary history. *Discrete Applied Mathematics*, 155:914–928, 2007.

[6] G. Cardona, M. Llabrés, F. Rosselló and G. Valiente. Metrics for Phylogenetic Networks I: Generalizations of the Robinson-Foulds Metric. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 6(1):46-61, 2009.

[7] Z. Chen and L. Wang. Algorithms for reticulate networks of multiple phylogenetic trees. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 9 (2):372–384, 2012.

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

[9] D. H. Huson, R. Rupp, and C. Scornavacca. *Phylogenetic Networks: Concepts, Algorithms and Applications*. Cambridge University Press, Cambridge, U. K., 2010.

[10] D. A. Morrison. *Introduction to Phylogenetic Networks*. RJR Productions, Uppsala, Sweden, 2011.

[11] L. Nakhleh. *A Metric on the Space of Reduced Phylogenetic Networks*. IEEE/ACM Trans. Comput. Biology Bioinform. 7(2): 218-222, 2010.

[12] H. J. Park and L. Nakhleh. MURPAR: A fast heuristic for inferring parsimonious phylogenetic networks from multiple gene trees. In *Proc. of Bioinformatics Research and Applications (ISBRA 2012)*, pages 213–224. Springer, LNCS 7292, 2012.

[13] C. Semple. Hybridization networks. In O. Gascuel and M. Steel, editors, *Reconstructing Evolution: New Mathematical and Computational Advances*, pages 277–309. Oxford, 2007.

[14] L. van Iersel, J. Keijsper, S. Kelk, L. Stougie, F. Hagen, and T. Boekhout. Constructing level-2 phylogenetic networks from triplets. In M. Vingron and L. Wong, editors, *Proc. of RECOMB 2008: The 12th Ann. International Conference Research in Computational Molecular Biology(RECOMB 08)*, pages 450–462. Springer, LNBI 4955, 2008.

[15] L. van Iersel, S. Kelk, R. Rupp, and D. Huson. Phylogenetic networks do not need to be complex: using fewer reticulations to represent conflicting clusters. *Bioinformatics (supplement issue for ISMB 2010 proceedings)*, 26:i124–i131, 2010.

[16] L. van Iersel, and S. Linz. A quadratic kernel for computing the hybridization number of multiple trees. *Information Processing Letters*, 113 (9) 318 – 323, 2013.

[17] C. Whidden and N. Zeh. A unifying view on approximation and FPT of agreement forests. In *Proc. of Algorithms in Bioinformatics (WABI 2009)*, pages 390–402. Springer, LNCS 5724, 2009.

[18] Y. Wu. Close lower and upper bounds for the minimum reticulate network of multiple phylogenetic trees. *Bioinformatics (supplement issue for ISMB 2010 proceedings)*, 26:140–148, 2010.

[19] Y. Wu. An Algorithm for Constructing Parsimonious Hybridization Networks with Multiple Phylogenetic Trees. Journal of Computational Biology, 20:792–804, 2013.

[20] Y. Wu and J. Wang. Fast computation of the exact hybridization number of two phylogenetic trees. In *Proc. of International Symposium on Bioinformatics Research and Applications (ISBRA 2010)*, pages 203–214, Springer, LNCS 6053, 2010.