AN OPTIMIZATION BASED APPROACH FOR COMBINING SEMI-LABELED ROOTED PHYLOGENETIC TREES

M. A. Hai Zahid, Ankush Mittal and R. C. Joshi

Department of E&CE Indian Institute of Technology Roorkee, Uttaranchal, India

{zaheddec,ankumfec,rcjosfec}@iitr.ernet.

Abstract: There are two approaches, agreement and optimization, for combining the small phylogenetic rooted trees with overlapping taxa into a single tree. Almost all the existing supertree methods, irrespective of the underlying approach, are developed based on the implicit assumption that only leaf nodes are labeled in the input trees. Recently, an interesting problem of combing the input trees is posted, in which the leaf nodes and some internal nodes are labeled [1]. The phylogenies constructed based on morphological studies often contains the labeled internal nodes, thus needing a more generalized supertree approach. The existing methods solving this problem take an agreement approach and fail to return any tree if the input trees are incompatible. In this paper we propose an optimization based divide and conquer method to combined semi-labeled trees. This method will return a tree even for (descendent level) incompatible input trees based on least square optimization criterion. The algorithm is applied on two data sets, one consists of semi labeled phylogenetic trees of spiders, and the second data set consists of hypothetical fully labeled tress.

Introduction

Phylogenetic supertree methods are used to combine a collection of small phylogenetic trees with overlapping set of taxa into a single tree, representing the branching information carried by each input tree. Supertree methods have emerged as the inevitable branch of computational biology. For a comprehensive survey of the supertree methods the readers are referred to [2]. Very few supertree algorithms exist, which satisfy the following properties [3].

1. The method should run in polynomial time.
2. If input trees are compatible, the resulting supertree preserves the branching information carried by each tree. If more than one, such trees exist then any one of them is produced.
3. The output is independent of the order, and relabelling of leaf node of the input trees.
4. All the leaf nodes that occur in the input trees occur in supertree.

Almost all the existing supertree methods, irrespective of underlying approach, are developed based on the implicit assumption that only leaf nodes are labeled in the input trees. Recently, Page [1] posted an interesting problem of combing the input trees, in which the leaf nodes and some internal nodes are labeled. The phylogenies constructed based on morphological studies often contain the labeled internal nodes, requiring a more generalized supertree approach. Till date, ANCESTRALBUILD [4] and NESTEDSUPERTREE [5] are the only known algorithms for constructing supertree of the small phylogenetic input trees with labeled internal nodes.

Both the algorithms are all-or-nothing algorithms. NESTEDSUPERTREE is a generalization of the ANCESTRALBUILD, both the algorithms gives either a tree compatible to all the input trees or a message indicating that the input trees are not ancestrally compatible or there is ancestor descendent conflict. The formal definitions for ancestral compatibility and ancestor descendent conflict are given in the coming sections.

In this paper we propose an optimization based divide and conquer method to combined semi-labeled trees. This method will return a tree even for incompatible input trees. The conflict is removed by deleting the conflicting edges based on least square error estimates. Once the conflict is resolved, the Adam's consensus algorithm [6] is used to construct the supertree. The algorithm is applied on two data sets, one consists of semi labeled phylogenetic trees of spiders, and the second data set consists of hypothetical fully labeled tress.

This is one of the most exciting challenges for constructing Tree of Life consisting of 1.7million described species, which provides a framework to facilitate biological information retrieval and predication. Many phylogenetic trees published in the early phylogenetic literature and they were constructed based on the morphology, thus often contains some of the internal nodes labelled but unfortunately no good algorithm exists to address this issue. In this paper we give a robust supertree method for an extremely important problem in classification.

In the following sections we discuss the background and preliminaries, algorithm with two examples followed by the conclusions.

Background and Preliminaries

In this paper we follow the terminology analogous to [4] and [5]. Here we present some of the basic concepts which are sufficient to understand the problem as well as proposed solution.

A rooted phylogenetic tree, on label set *S*, is a tree *T* with a function *f*, in which all the node have degree three or more except the root node and leaf node, which have degree at least two and one respectively. The function *f* maps from *S* to leaf nodes of the tree *T*, $f : S \rightarrow set\ of\ leaf\ nodes$. Semi-labelled phylogenetic tree on label set *S* is a generalisation of a simple phylogenetic tree with all the leaf nodes as well as some of the internal nodes are labelled. Let *T* is a semi-labelled tree with vertex set *V*, then the function $f : S \rightarrow V$, assigns the labels to node with the degree one and two. Moreover some of the internal nodes can even be labelled but the leaf nodes must be labelled.

Let *T* be the rooted phylogenetic tree with the label set *S*. Given the label set *S'*, such that $S' \subseteq S$, the topological restriction of *T* to *S'* is the tree obtained by deleting the nodes which are not in the path from root to any node in *S'* and then contracting the internal edges whose degree two. The topological restriction is represented as *T'*=*T*|*S'*, an example is shown in Figure 1. *T'* is called the induced subtree of *T* by *S'*. A rooted tree *T* is said to display another rooted tree *T'* if $S' \subseteq S$ and *T*|*S'* is isomorphic to *T'*. A set of semi-labeled trees *G* is said to be compatible if there exists a semi-labeled tree *T*, which display every tree in *G*.

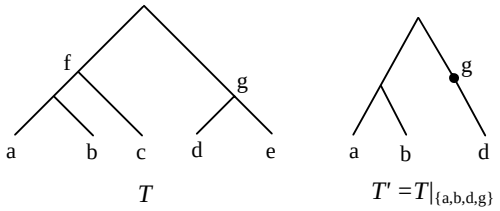

Figure 1: Two rooted semi-labeled trees. *T'* is induced subtree or restriction of *T* on the labels {a,b,d,g}.

A node *a* is said to be descendent of another node *b* if the path from root to *a* includes the node *b*. A rooted tree *T* is said to ancestrally display another rooted tree *T'* if $S' \subseteq S$ and *T*|*S'* is restricted in such a way that when ever *a* is strict descendent of *b* in *T* then *a* is also a strict descendent of *b* in *T'*. A collection of semi labeled trees *Z* is ancestrally compatible if a semi-labeled tree *R* ancestrally displays all the trees in *Z*. Two labels, *a* and *b*, are said to be pairwise consistent if, when ever *a* is a strict descendent of *b* in some tree in *Z* then *a* is always a strict descendent of *b* in every tree of *Z* whose label contain both *a* and *b*.

The ResolvedSupertree Algorithm

In this section we define the ResolvedSupertree algorithm for semi-labeled collection of input trees. The algorithm is based on the construction of conflict graph. The conflicts are identified and resolved using least square error criterion. Then the Adam's consensus tree methods is used for the construction of the semi-labeled supertrees.

As a first step, new distinct labels are assigned to the root nodes of the input trees if the root node is not labeled. The input trees are then divided into multiple trees by cutting the tree till its internal label. The internal labeled node is considered as the leaf node label for the parent tree and root node label for the child tree. The division of the semi-labeled input trees is shown in figure 2.

The conflict graph consists of only weighted arcs (directed edges) and may have cycles in it. Let *Z'* be the collection of semi-labeled rooted trees and their child trees. The arcs from node labeled *a* to the node labeled *b* is added to the conflict graph if *a* is an ancestor of *b* in any of the tree in *Z'*. The number of trees that represent the same child parent relationship of the nodes are assigned as the arc weight. The example is shown in figure 3.

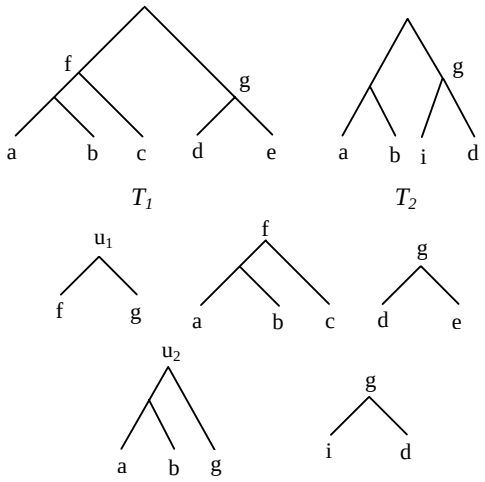

Figure 2: Two rooted semi-labeled trees T_1 and T_2 and the resulting trees after dividing from internal labels.

The root nodes of the trees T_1 and T_2 are not labeled, for the construction of conflict graph we assigned new distinct labels u_1 and u_2 to the root nodes of the trees T_1 and T_2 respectively. The conflict graph for the trees T_1 and T_2 (shown in figure 2) is given in figure3.

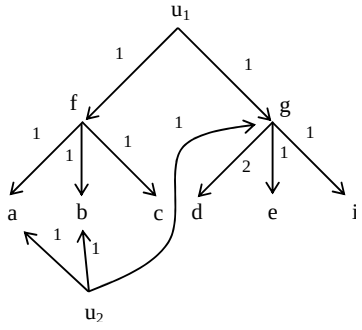

Figure 3: Conflict graph for semi-labeled trees T_1 and T_2 (shown in Figure 2).

Once the conflict graph (*CG*) is ready, it is searched for different conflicts and the following algorithm ConflictResolve is applied to resolve the conflicts. The algorithm results in a directed tree and parent table. The directed tree, with all the nodes having indegree exactly one, except the root node have indegree zero, is also

called resolved graph. Based on the resolved graph the subtrees are modified.

Let the node *a* is ancestor for nodes {*b,c,d,e*} in divided collection of trees. The resolved graph show that the node *a* has child list {*b,c,d*}, then all the trees in the divided collection with the root node labeled *a* are refined for child list {*b,c,d*}. Therefore the final trees having root as *a*, which are combined to get the supertree, have {*b,c,d*} its subset as its descendants. The parent table gives the information about the label of the least common ancestor of the set of labels given in the table.

Algorithm: ConflictResolve (*CG*)
Input: Conflict Graph constructed for the given input semi-labeled trees.
Output: resolved graph and parent table.

begin
for each node *n* in *CG* having indegree two or more **do**
If the originating node has indegree 0 (zero) **then** add the *n* to the child list of originating node and remove the arc between them.
else
Delete all minimum weight incoming edges.
In case of tie or more than one incoming edges, the following rule can be applied:
If *n* has O_1 and O_2 as originating nodes then if O_1 is the descendent of O_2 then the link between O_2 and *n* is removed.
end (if-else)
end (for)
for each directed cycle in *CG* **do**
remove the arc with least weight. In case of tie remove each minimum weighted arc and calculated the error using Least Square error criterion. Remove the edge with minimum least square error.
end (for)
return *MCG* (modified conflict graph) and parent table;
end (algo)

The algorithm ConflictResolve returns the modified conflict graph and parent table. The modified conflict graph is obtained by removing the conflicting arcs which leads to minimum error. All the nodes in the resulting modified graph have indegree equal to one. The immediate descendents of each node *n* in modified conflict graph represents the set of labels the trees with *n* as root label can have. Finally the Adams consensus tree is used for the construction of the supertree.

Algorithm: ResolveSupertree (*Z, MCG, parent table*)
Input: divided collection if semi-labled input trees (*Z*), Modified Conflict Graph and parent table.
Output: Supertree for semi-labelled input trees.

begin
for each node *n* in modified conflict graph **do**
make a set, *S*, of all immediate descendents of node *n*.
find the restrictions of all the trees in Z with root *n* on *S*.
for all modified trees with root node *n* **do**
find the restriction of each tree on common nodes
apply Adams consensus tree for the modified trees.
Add remaining taxa to appropriate edge.
end (inner for)
end (outer for)
merge all the trees to make single tree. // as if tree *T* // has a leaf node label *x* same the root label of *T'* // then T' is attached to T at the leaf node *x*.
for each entry *e* in parent table **do**
add a label, *e*, to the most recent common ancestor of the set element in entry *e*.
end (for)
remove all the new distinct labels assigned to the roots of the unlabeled roots of the input trees.
end (algo)

To illustrate the algorithm, ResolvedSupertree is applied on two data sets, one consists of semi labeled phylogenetic trees of spiders, and the second data set consists of hypothetical fully labeled tress. The two spider trees shown in figure 4 are taken from [1].

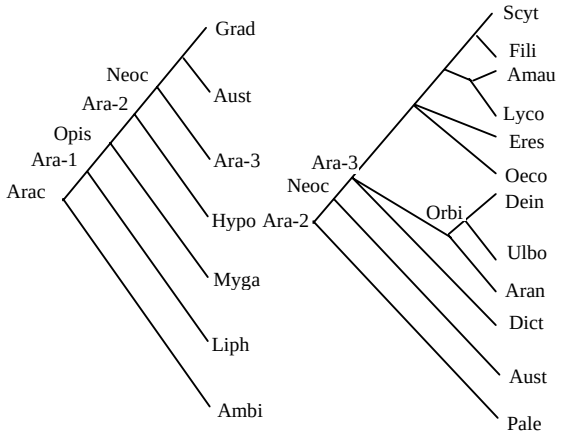

Figure 4: Two trees for spiders and related taxa. These can be obtained from study S1x6x97c14c42c30 from TreeBase.

We have taken the first four letters of the taxa in the figure 4, the full names of the taxa can be obtained from study S1x6x97c14c42c30 from TreeBase [7] or [1]. As a first step, labels are assigned to the root nodes of the input trees with unlabeled root nodes, but both the trees are root labeled, so this step is avoided. Now the trees are divided into child and parent tree, through its internal label node. The internal label node, in master tree, is now leaf node in parent tree and root node label in child tree. The collection of trees divided from input trees (figure 4) is shown in figure 5.

The conflict graph is constructed based on the guidelines given in the algorithm. The conflict graph for input trees (figure 4) is shown in figure 6. The collection of divided trees and the conflict graph is given as the input to the ConflictResolve algorithm. As

the input conflict graph does not have any directed or underlying cycles, the algorithm return the same conflict graph with only Arac and relative arcs from the graph, this is added as the most recent common ancestor of the labels Ara-1 and Ambi, in parent table.

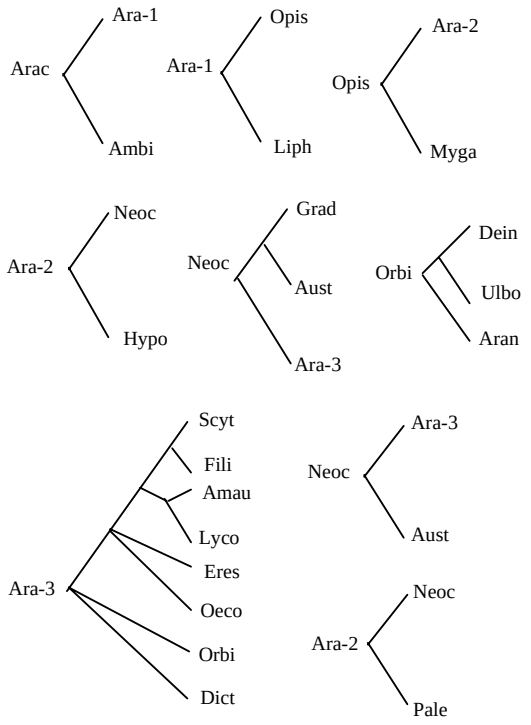

Figure 5: Trees obtained from dividing the input trees shown in figure 4.

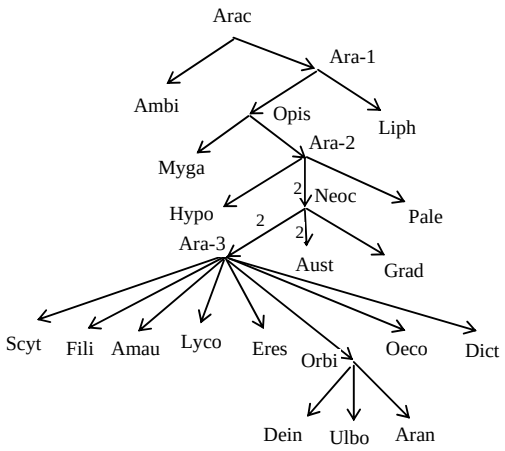

Figure 6: Conflict graph for the input trees shown in figure 4.

The conflict graph shown in figure 6 does not have any cycles and underlying cycles in it. The arcs, Ara-2 to Neoc, Neoc to Ara-3, Neoc to Aust, are weighted 2 as they appear in both the input trees, rest of the arcs carry unit weight. Now it is straightforward to costruct the supertree for each smaller tree with same root label and merge them for the final supertree. The final supertree for the input trees is shown in figure 7.

The supertree can be constructed for the trees shown in figure 2 using the conflict graph shown in figure 3.

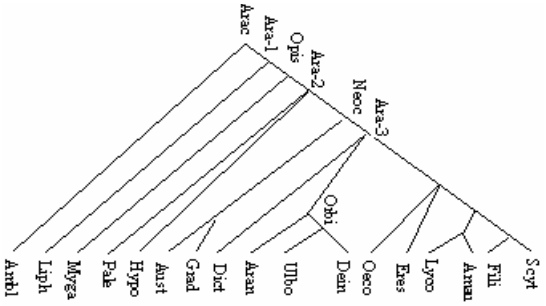

Figure 7: Supertree for the spider trees shown in figure 4.

Conclusions

The proposed algorithm addresses an extremely important problem in classification. The ResolvedSupertree outperforms the NESTEDSUPERTREE and ANCESTRALBUILD as it preserves all the nesting information common to all the input trees due to underlying Adams consensus tree construction method. On the other hand, both the NESTEDSUPERTREE and ANCESTRALBUILD may not display the information carried by all the input trees [5].

Above all, this is the first algorithm which resolves the conflicts or incompatibilities and returns the semi-labelled supertree even for incompatible semi-labelled input trees. The conflicts are removed based on minimum error criterion, which is quite reasonable approach for resolving the conflicts.

References

[1] Page, R.D.M. (2004): 'Taxonomy, supertrees, and the Tree of Life', in Bininda-emonds, O. (Ed): 'Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life', Computational Biology Series, Kluwer, pp. 247-265.

[2] Bininda-emonds, O. (2004): 'The evolution of supertrees', *Trends in Ecol. and Evol.*, **19**, pp. 315-322.

[3] Semple, C., Steel, M., (2000): 'A supertree method for rooted trees', *Disc. Appl. Math.*, **105**, pp. 147-158.

[4] Denial, P., Semple, C. (2004): 'Supertree Algorithms for nested taxa', in Bininda-emonds, O. (Ed): 'Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life', Computational Biology Series, Kluwer, pp. 151-171.

[5] Denial, P., Semple, C. (2004): 'A class of general supertree methods for nested taxa', *SIAM Journal of Discrete Maths*, in press.

[6] Adams, E.N., (1972): Consensus techniques and comparison of texanomic trees', Sys. Zool., **21**, pp. 390-397.

[7] http://www.treebase.org