

Modified mincut supertrees

Roderic D. M. Page*

DEEB, IBLS, University of Glasgow,
Glasgow G12 8QQ, United Kingdom
r.page@bio.gla.ac.uk

Abstract. A polynomial time supertree algorithm could play a key role in a divide-and-conquer strategy for assembling the tree of life. To date only a single such method capable of accommodate conflicting input trees has been proposed, the MINCUTSUPERTREE algorithm of Semple and Steel. This paper describes this algorithm and its implementation, then illustrates some weaknesses of the method. A modification to the algorithm that avoids some of these problems is proposed. The paper concludes by discussing some practical problems in supertree construction.

1 Introduction

A central goal of systematics is the construction of a “tree of life,” a tree representing the relationships among all living things. One approach to this goal is to assemble a “supertree” from many smaller trees, inferred for different sets of taxa using a variety of data types and (in many cases) a range of phylogenetic methods [1]. Steel et al. [2] list five desirable properties of a supertree method:

1. Changing the order of the trees in the set of input trees \mathcal{T} does not change the resulting supertree.
2. Relabelling the input species results in the corresponding relabelling of species in the supertree.
3. If there are one or more parent trees with which every tree in \mathcal{T} is compatible then the supertree is one of those parent trees.
4. Any leaf that occurs in at least one input tree occurs in the supertree.
5. The supertree can be computed in polynomial time.

Currently the most widely used supertree algorithm in phylogenetics is Matrix Representation using Parsimony (MRP) [3, 4]. MRP encodes the input trees into binary characters, and the supertree is constructed from the resulting data matrix using a parsimony tree building method. The MRP supertree method

* I thank David Bryant, James Cotton, Mike Steel, and Joe Thorley for comments on the manuscript. Computational resources were provided by a grant from the Wolfson Foundation.

does not satisfy property 5 above because the problem of finding the most parsimonious tree is NP-complete [5]. Hence the task of constructing an MRP supertree is of the same order of complexity as inferring individual phylogenies. Similarly, supertrees by flipping [6] is NP-complete.

To date the only supertree method that satisfies all five desirable properties is the mincut supertree algorithm of Semple and Steel [7]. This algorithm is very attractive because it can potentially scale to handle large problems. This is particularly important if supertrees are going to be an effective tool for synthesizing very large trees from many smaller trees, such as those contained in TreeBASE [8] (<http://www.treebase.org>). A polynomial time supertree algorithm could play a key role in a divide-and-conquer strategy for assembling the tree of life.

In this paper I start by outlining the mincut supertree method, and discussing its implementation. I then note some limitations of the method, and propose modifications that increase its practical utility.

2 Supertrees

2.1 Tree terminology

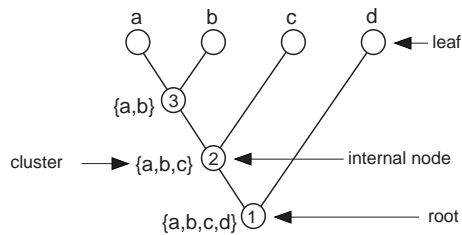


Fig. 1. The tree terminology used in this paper

The terminology used in this paper is illustrated in figure 1. I will consider only rooted trees. The *leaves* of the tree represent species. Let $\mathcal{L}(T)$ be the set of leaves for tree T . The set of leaves that descend from an internal node in a tree comprise that node's *cluster*. In the tree shown in figure 1, node 3 has the cluster $\{a, b\}$. The most recent common ancestor (MRCA) of a set of nodes in a tree is the node furthest from the root of the tree whose cluster contains all those nodes. For example, in figure 1 $\text{MRCA}(b, c)$ is node 2. Given a tree T , if A and B are subsets of $\mathcal{L}(T)$ and $\text{MRCA}(A)$ is a descendant of $\text{MRCA}(B)$, then $A <_T B$, that is, A nests in B [9]. Examples of nestings in figure 1 include $\{a, b\} <_T \{a, b, c\}$ and $\{b, c\} <_T \{a, b, c, d\}$.

Suppose we have a set of k trees \mathcal{T} with different, overlapping leaf sets. Let $S = \cup_{i=1}^k \mathcal{L}(T_i)$ (i.e., the set of all species which are in at least one of the trees in \mathcal{T}). A supertree method takes \mathcal{T} as input and returns a supertree with the leaf set S (see fig. 2).

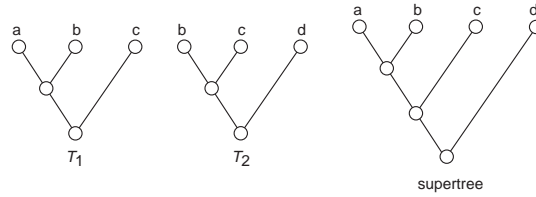


Fig. 2. Two input trees T_1 and T_2 and a supertree

2.2 ONETREE algorithm

Given a set of rooted trees, the question of whether they are compatible – that is, whether there is a supertree with which each input tree is compatible – can be answered in polynomial time using algorithms that owe their origin to [10]. To appreciate how Semple and Steel’s algorithm works, let us first consider Bryant and Steel’s [11] version of algorithm ONETREE [12]. Given three species a, b, c , let the triplet $ab|c$ be the rooted tree where a and b are more closely related to each other than either is to c . A binary tree can be decomposed into a set of triplets. Bryant and Steel construct the graph $[R, S]$ where the nodes S correspond to the species, and there is an edge between nodes a and b if there are any triplets of the form $ab|c$ in the set of input trees. The algorithm ONETREE takes as input a set of triplets and a set of species $S = \{x_1, \dots, x_n\}$.

procedure OneTree(R, S)

1. **if** $n = 1$ **then return** a single node labelled by x_1 .
2. **if** $n = 2$ **then return** a tree with two leaves labelled by x_1 and x_2 .
3. Otherwise, construct $[R, S]$ as described.
4. **if** $[R, S]$ has only one component **then return** ‘no tree’.
5. **for** each component S_i of $[R, S]$ **do**
 if ONETREE(R, S_i) returns a tree **then** call it T_i **else** return ‘no tree’.
6. Construct a new tree T by connecting the roots of the trees T_i to a new root r .
7. **return** T

end

The key step in this recursive algorithm is step 4. Bryant and Steel [11] showed that a set of rooted triplets R is compatible if and only if, for any subset S' of S of size at least 2 the graph $[R, S']$ is disconnected. If at any point in the execution of ONETREE the graph $[R, S]$ is connected then the algorithm exits without returning a tree. Hence ONETREE offers a polynomial time test for tree compatibility.

If there is a supertree with which all the input trees are compatible, then ONETREE will return that tree, otherwise the algorithm returns the result that no such tree exists. This severely limits its application to the supertree problem in phylogenetics. Any non-trivial set of phylogenetic trees is likely to be

incompatible for a range of reasons, including sampling error, inaccuracies, or biases in tree building algorithms. What is needed is an algorithm that will find a supertree even if the input trees are incompatible. Given that ONETREE exits if $[R, S]$ is connected, one approach to extending the algorithm would be to disconnect $[R, S]$ whenever it is connected. This is the rationale behind Semple and Steel's [7] mincut algorithm.

2.3 MINCUTSUPERTREE

Semple and Steel's MINCUTSUPERTREE algorithm modifies ONETREE so that it always returns a tree. Instead of constructing $[R, S]$ from a set of triplets, MINCUTSUPERTREE constructs a graph $S_{\mathcal{T}}$, where the nodes are species, and nodes a and b are connected if a and b are in a proper cluster in at least one of the input trees (i.e., if there is a tree in which $\text{MRCA}(a, b)$ is not the root of the tree). Semple and Steel's algorithm takes a set of k rooted trees \mathcal{T} and a set of species $S = \cup_{i=1}^k \mathcal{L}(T_i) = \{x_1, \dots, x_n\}$.

procedure MinCutSupertree(\mathcal{T})

1. **if** $n = 1$ **then return** a single node labelled by x_1 .
2. **if** $n = 2$ **then return** a tree with two leaves labelled by x_1 and x_2 .
3. Otherwise, construct $S_{\mathcal{T}}$ as described.
4. **if** $S_{\mathcal{T}}$ is disconnected **then**
 Let S_i be the components of $S_{\mathcal{T}}$.
 else Create graph $S_{\mathcal{T}}/E_{\mathcal{T}}^{\text{max}}$ and delete all edges in $S_{\mathcal{T}}/E_{\mathcal{T}}^{\text{max}}$ that are in a minimum cut set of $S_{\mathcal{T}}$. Let S_i be the resulting components of $S_{\mathcal{T}}/E_{\mathcal{T}}^{\text{max}}$.
5. **for** each component S_i **do**
 $T_i = \text{MinCutSupertree}(\mathcal{T}|S_i)$, where $\mathcal{T}|S_i$ is the set of input trees with any species not in S_i pruned.
6. Construct a new tree \mathcal{T} by connecting the roots of the trees T_i to a new root r .
7. **return** T

end

In step 4 Semple and Steel ensure that $S_{\mathcal{T}}$ yields more than one component by using minimum cuts. Given a connected graph G , a set of edges whose removal disconnects the graph is a *cut set*. If each edge in G has a weight assigned to it, then a cut set with the smallest sum of weights is a *minimum cut* of the graph. Semple and Steel do not find minimum cuts of $S_{\mathcal{T}}$, but rather of an associated graph $S_{\mathcal{T}}/E_{\mathcal{T}}^{\text{max}}$ which is constructed as follows::

1. Weight each edge (a, b) in $S_{\mathcal{T}}$ by the number of trees in \mathcal{T} in which a and b are in the same proper cluster.
2. Let $E_{\mathcal{T}}^{\text{max}}$ be the set of edges that have weight k , where k is the number of trees in \mathcal{T} .
3. Merge any nodes in $S_{\mathcal{T}}$ that are connected by edges in $E_{\mathcal{T}}^{\text{max}}$.

For example, given the two input trees in figure 3, the edge (a, b) in $S_{\mathcal{T}}$ has a weight of 2, and hence the nodes a and b are merged. This procedure ensures that any nesting found in all of the input trees \mathcal{T} will be in the supertree returned by MINCUTSUPERTREE¹.

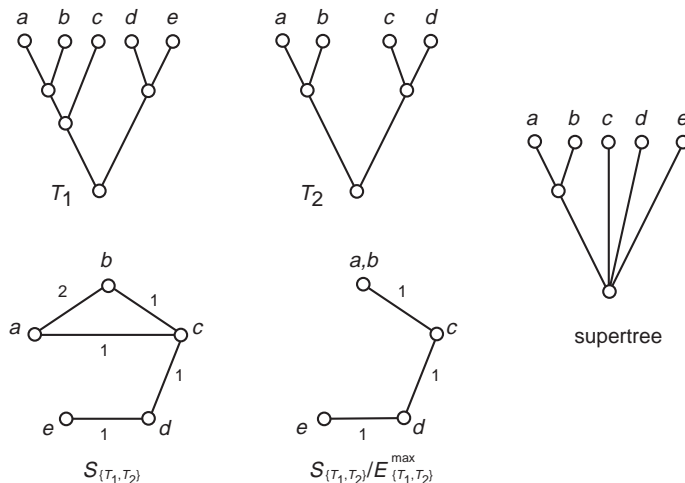


Fig. 3. An example of the MINCUTSUPERTREE algorithm showing the two input trees T_1 and T_2 , and the graphs $S_{\mathcal{T}}$ and $S_{\mathcal{T}}/E_{\mathcal{T}}^{\max}$. The graph $S_{\mathcal{T}}/E_{\mathcal{T}}^{\max}$ has three minimum cut sets, which yield the components $\{a, b\}$, $\{c\}$, $\{d\}$, and $\{e\}$, which in turn yield the supertree (from [7])

2.4 Implementation

Because the mincut problem can be solved in polynomial time, so can MINCUTSUPERTREE. The crucial computational step in the MINCUTSUPERTREE algorithm is determining for each edge in $S_{\mathcal{T}}/E_{\mathcal{T}}^{\max}$ whether that edge is in a minimum cut set of $S_{\mathcal{T}}/E_{\mathcal{T}}^{\max}$. Semple and Steel suggest an algorithm for doing this that requires computing the minimum cut for all the subgraphs of $S_{\mathcal{T}}/E_{\mathcal{T}}^{\max}$ that result from deleting a single edge of that graph. Doing this for all the edges of $S_{\mathcal{T}}/E_{\mathcal{T}}^{\max}$ rapidly becomes tedious for all but fairly small examples. We can improve on this by using Picard and Queryanne's [13] algorithm to find all minimum cuts of the graph. Firstly all (s, t) minimum cuts of $S_{\mathcal{T}}/E_{\mathcal{T}}^{\max}$ are found [14], where s and t are nodes in $S_{\mathcal{T}}/E_{\mathcal{T}}^{\max}$ that are disconnected by the cut. For

¹ Note that in Semple and Steel's original algorithm each input tree in \mathcal{T} can have a weight $w(T)$ assigned to it, and in step 1 above they weight each edge by the sum of the weights of the trees in which a and b are in the same proper cluster. The set of edges E^{\max} comprises those edges for which $w_{sum} = \sum_{T \in \mathcal{T}} w(T)$. For simplicity here I consider only the case where all trees have the same unit weight.

each (s, t) cut, the graph $S_{\mathcal{T}}/E_{\mathcal{T}}^{\max}$ is transformed into a directed graph, and the maximum $s - t$ flow computed. The strongly connected components of the residual graph R for this flow are then computed. Any edge in R which connects nodes belonging to different strongly connected components is a member of a minimum cut set of $S_{\mathcal{T}}/E_{\mathcal{T}}^{\max}$ [13].

Another shortcut concerns the recursion in MINCUTSUPERTREE. As originally described the algorithm keeps calling itself recursively until $\mathcal{T}|S$ contains less than three leaves (steps 1 and 2 above), at which point the algorithm returns either a single-leaf or a two-leaf tree. However, if at any point there is only a single tree T in $\mathcal{T}|S_i$ with any leaves, then that tree can be grafted directly onto the growing supertree.

3 Properties of MINCUTSUPERTREE

MINCUTSUPERTREE has at least two attractive properties: it is quick to compute, and any nesting found in all the input tree will appear in the supertree [7]. However, when applied to some examples it can yield slightly disconcerting results. Fig. 4 shows two trees that share only three leaves (a , b , and c), and disagree on the relationships among those leaves. Both trees have unique sets of leaves with fully resolved relationships. In the tree produced by MINCUTSUPERTREE for these two trees the relationships among $x_1 - x_3$ are unresolved, although there is no information in the input trees that contradicts this grouping. In contrast, relationships among $y_1 - y_4$ are fully resolved. Furthermore, even though the two trees disagree about the relations between a , b , and c , the supertree contains the triplet $ab|c$.

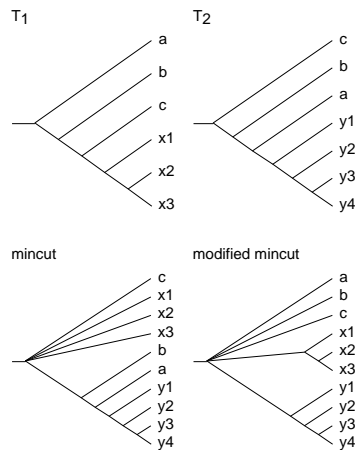


Fig. 4. Two source trees that share the leaves a , b , and c , and their mincut and modified mincut supertrees

These features of the supertree are due, at least in part, to the different sizes of the two input trees. Fig. 5 shows the graph $S_{\mathcal{T}}/E_{\mathcal{T}}^{\max}$ for these two trees. Because there is no triplet common to the two subtrees, no edge has a weight greater than one. In this case, minimum-weight cuts sets will involve edges connected to nodes with the lowest degree. These are all nodes in the smaller of the two source trees (T_1).

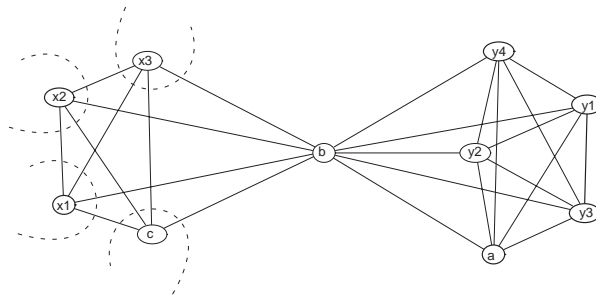


Fig. 5. The graph $S_{\mathcal{T}}/E_{\mathcal{T}}^{\max}$ for the trees shown in figure 4. The four minimum weight cuts of the graph are indicated by dashed lines

This example shows that MINCUTSUPERTREE can be sensitive to the size of the input trees, and can fail to include information that is not contradicted in the set of input trees. Indeed, the original illustration of Semple and Steel’s method (fig. 3) provides an example of the latter problem. The edge connecting nodes d and e in the graph $S_{\mathcal{T}}/E_{\mathcal{T}}^{\max}$ represents the uncontradicted nesting $\{d, e\} <_{T_1} \{a, b, c, d\}$. This nesting is not present in the supertree, and as a result the supertree has less information than the input trees. Although there is no consensus method that can always display all the uncontradicted information in a set of trees [2], it would be desirable to maximise in some sense the amount of uncontradicted information a supertree displays.

If mincut supertrees are to be useful then we need ways to avoid these problems. The next section explores one way of modifying the way the graph $S_{\mathcal{T}}/E_{\mathcal{T}}^{\max}$ is created so as to minimise the impact of these problems.

3.1 Modifications

We can refine Semple and Steel’s approach by distinguishing between “unanimous” and “uncontradicted” nestings in $\mathcal{T}|S$. A unanimous nesting occurs in every tree in $\mathcal{T}|S$. These nestings correspond to edges with weight $w(e) = k$, where k is the number of trees in $\mathcal{T}|S$. Semple and Steel’s algorithm ensures that all unanimous nestings are preserved in the mincut supertree. I define an uncontradicted nesting to be a nesting found in a subset of trees in $\mathcal{T}|S$, but which no tree in $\mathcal{T}|S$ contradicts. Let m be the number of trees in $\mathcal{T}|S$ in which leaves i and j both occur. Let r be the number of trees containing both i and

j for which $\text{MRCA}(i, j)$ is not the root. If all trees in $\mathcal{T}|S$ are binary, then if $m > 0$ and $r = 0$ then the nesting is uncontradicted.

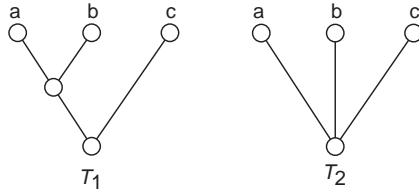


Fig. 6. A triplet (T_1) and a fan (T_2)

Fans For nonbinary trees the situation is a little more complicated. Consider the two trees shown in figure 6. Tree T_1 contains the nesting $(a, b) <_T (a, b, c)$, whereas T_2 other does not. Whether we regard T_2 as disagreeing with T_1 depends on our interpretation of the polytomy in T_2 . Polytomies can be true (“hard”) or apparent (“soft”) [15, 16]. A true polytomy results when three or more lineages diverge from a single ancestor at the same time. In a tree derived from a single data set an apparent polytomy represents a lack of information, such as no substitutions occurring along a branch on the tree [16]. If the tree is a consensus tree that summarises a set of input trees, then an apparent polytomy is used to depict conflict between the original trees. If the polytomy in tree T_2 is true, then T_1 and T_2 conflict, if the polytomy is only apparent then T_2 is consistent with any of the three rooted trees for a, b, and c, and hence T_1 and T_2 are consistent.

The distinction between true and apparent polytomies corresponds to different use of the term “compatible” [17, 12]. Ng and Wormald [12] regard two trees are only compatible only if one is a subtree of the other, whereas for Steel [17] two trees will also be considered compatible if one can be derived from the other by removing edges. In figure 6 T_2 is not a subtree of T_1 , but by removing an edge from T_1 we can collapse T_1 into the fan T_2 .

Given that true polytomies are likely to be rare [16], in most cases any polytomies in the trees in $\mathcal{T}|S$ will be apparent (soft), and hence we need to take this into account when deciding whether a nesting is contradicted. Specifically, let f be the number of trees in $\mathcal{T}|S$ for which $\text{MRCA}(i, j)$ is the root of that tree, and the root has degree > 2 . A nesting that occurs in m trees in $\mathcal{T}|S$ is uncontradicted if $r - f = 0$.

We can modify Semple and Steel’s algorithm by assigning to each edge in $S_{\mathcal{T}}$ one of three colours: unanimous, uncontradicted, or contradicted. Our goal is to modify S in such a way as to minimise the number of uncontradicted edges that are cut. One way to do this is to extend Semple and Steel’s approach of collapsing nodes linked by unanimous edges to include nodes linked by uncontradicted edges:

1. Construct $S' = S_{\mathcal{T}}/E_{\mathcal{T}}^{\max}$ as described in MINCUTSUPERTREE.
2. Let C be the set of contradicted edges in S' .
3. Remove all contradicted edges from S' to yield the graph S'/C .
4. **if** S'/C is disconnected **then**
 S' can be cut without cutting an uncontradicted edge. Find the components of S'/C and merge the nodes in each component.
5. Restore the edges in C (removing any parallel edges resulting from nodes merged in step 4). The graph S' now comprises nodes representing unanimous and uncontradicted nestings, connected by edges representing contradicted nestings.

When then find all edges in at least one cutset of S' and proceed as in MINCUTSUPERTREE.

Example Applying the modified algorithm to the trees in figure 4 yields a more resolved supertree that includes the resolved subtrees for $x_1 - x_3$ from T_1 and $y_1 - y_4$ from T_2 . The relationships between the shared leaves a , b , and c are represented as a basal polytomy. It should be noted here that polytomies in mincut supertrees are interpreted in a similar fashion to polytomies in Adams consensus trees [9]. This means that groupings in the supertree tree are not necessarily clusters (i.e., monophyletic groups of species).²

3.2 How good is a supertree?

In addition to computational complexity, another criterion for evaluating the efficacy of a supertree method is the degree to which the supertree agrees with the input trees \mathcal{T} [19]. This can be measured using a tree comparison metric. For example, we can measure the agreement between an input tree T and a supertree by the size of the maximum agreement subtree (MAST) for the two trees [6]. Although efficient algorithms for MAST are available [20], it has limitations as a measure of tree similarity, particularly if the input trees are not fully resolved [21]. Here I use triplet measures that are the rooted equivalent of the quartet measures described by [22]. For each input tree T we compare that tree with the subtree of the supertree that results when any taxa not in T are pruned. By analogy with quartets [22], for any pair of triplets (one from each tree) there are five possible outcomes: the triplet is resolved in both trees is identical (d), or different (s); the triplet is resolved in one tree ($r1$) or the other ($r2$) but not both, or the triplet is unresolved in both trees (x). For our purposes, only d , s , and $r2$ are relevant. Triplets that are resolved only in the supertree ($r1$) reflect information present in other input trees rather than T , and triplets not resolved in either T or the supertree (x) are not counted as “agreement” because they

² Although the notion that Adams consensus trees are “difficult” to interpret has become part of taxonomic folklore, some biologists have argued that this way of interpreting polytomies “merges with, or is implicit within, routine taxonomic practice” [18].

reflect a shared lack of information. Hence, for each tree T the fit to the supertree is given by:

$$1 - \frac{d + r2}{d + s + r2}$$

The average of this value over all input trees is a measure of fit between \mathcal{T} and the supertree.

3.3 Empirical example



Fig. 7. Comparison of the modified mincut supertree and the MRP supertree for the two trees shown in [1, fig. 2]. The four species in common to the two input trees are shown in **boldface**

Figure 7 shows the modified mincut supertree for the two trees used by [1] to illustrate the MRP supertree method. For comparison, figure 7 also shows the strict consensus tree of the 8 equally parsimonious trees obtained for the MRP matrix for the two input trees. The two input trees share only four taxa (shown in **boldface** in fig. 7), and disagree about their relationships for three of the species: one tree has the triplet *Astragalus alpinus* | (*Oxytropis deflexa*, *Cilianthus puniceus*), the other tree has the triplet (*Astragalus alpinus*, *Oxytropis deflexa*) | *Cilianthus puniceus*. Given we have only two sources of information

about the relationships among these three species, and that these sources conflict, it is a little disconcerting that the MRP tree resolves this triplet in favour of the second tree. In contrast, the mincut supertree displays a fan for these three species.

4 Discussion

The modified mincut supertree algorithm presented here retains the desirable property of being able to be computed in polynomial time, as well as retaining more of the information shared among the input trees. However, its properties require further exploration. MRP is a global optimisation method that seeks to find one or more optimal supertrees. The mincut supertree method does not have an explicit global optimality criterion, instead it uses the locally optimal criterion of minimum cuts. Semple and Steel [7] show that it preserves any nesting found in all input trees, but its other properties are unclear. In addition to theoretical work on the properties of modified mincut supertrees, there are practical issues that are relevant to supertrees in general.

4.1 Labels

A supertree method requires that the same taxon be given the same name in all of the input trees containing that taxon. This requirement is obvious, but not always met. Proper application of the rules of taxonomic nomenclature should ensure that the same taxon is called the same thing in different studies but, sadly, this is not always the case. Even in very well known groups such as birds and mammals, there is disagreement about the name given to some taxa.

A further source of difficulty concerns combining trees containing taxa of different taxonomic rank. In phylogenies constructed from sequence data the leaves of the tree are typically individual organisms, such as *Mus musculus*. However, trees constructed from morphological data sets may have higher taxa as terminals, such as “Rodentia”. In the absence of any additional information, a supertree algorithm has no way of knowing that *Mus musculus* in one tree is equivalent to Rodentia in another. One solution would be to include information about higher taxon names in the input trees. TreeBASE [8] has a facility to do this, for example. With this information in hand we could, for example substitute a tree containing *Mus* and *Rattus* species for a leaf labelled Rodentia in another tree. However, this method of “taxon substitution” [23], which has parallels with substitution and adjunction operations in tree adjoining grammars [24], requires that higher taxon names are applied consistently. Given current controversy about the meaning and application of higher taxon names [25], this may be difficult to guarantee. Consequently, any mapping between taxonomic names at different levels would need to be tested for internal consistency.

4.2 Constraints

A potentially serious problem affecting supertree construction is poor overlap between different studies. Sanderson et al. [1] introduced the concept of a “tree-graph,” where each node represents an input tree, and nodes are connected by edges weighted by the number of taxa shared by the corresponding trees. For a supertree to be constructed from two trees, those trees must share a minimum of two taxa in common [1]. If the degree of overlap between input trees is poor then this may compromise the resulting supertree. This is a particular problem if input trees use different “exemplars” of higher taxa. For example, if one study on mammalian phylogeny used a mouse to represent rodents, and another study used a rat, then a supertree for those two studies would not group mouse and rat together. This same problem can be encountered when assembling species trees from trees for gene families [26]. One solution would be to impose a constraint on the set of possible solutions, for example by specifying a constraint tree [27] which must be displayed by the supertree. In the previous example, we could require that any solution grouped mouse and rat.

4.3 Software

The algorithms described here have been implemented in the C++ programming language and compiled using the GNU gcc compiler, and commercial compilers from Borland and Metrowerks. Paul Lewis’ NEXUS Class Library (<http://lewis.eeb.uconn.edu/lewishome/nc12/NexusClassLibrary.html>) is used to read NEXUS format tree files. The code makes extensive use of the Graph Template Library (GTL) (<http://www.infosun.fmi.uni-passau.de/GTL/>) which provides a Standard Template Library (STL) based library of graph classes and algorithms. Triplet tree comparison measures are computed using the quartet algorithm developed by Douchette [28]. The program reads trees in either NEXUS or Newick format, and outputs a supertree. The program can also output the graphs $S_{\mathcal{T}}$ and $S_{\mathcal{T}}/E_{\mathcal{T}}^{\max}$ in GML (<http://www.infosun.fmi.uni-passau.de/Graphlet/>) and dot formats (<http://www.research.att.com/sw/tools/graphviz/>) so the progress of the algorithm can be viewed. The supertree software is available at (<http://darwin.zoology.gla.ac.uk/~rpage/supertree/>).

References

- [1] Sanderson, M. J., Purvis, A., Henze, C.: Phylogenetic supertrees: assembling the trees of life *Trends Ecol. Evol.* **13** (1998) 105–109
- [2] Steel, M., Dress, A., Böcker, S.: Simple but fundamental limitations on supertree and consensus tree methods. *Syst. Biol.* **49** (2000) 363–368
- [3] Baum, B.R.: Combining trees as a way of combining data sets for phylogenetic inference, and the desirability of combining gene trees. *Taxon* **41** (1992) 3–10
- [4] Ragan, M.A.: Phylogenetic inference based on matrix representation of trees. *Mol. Phylogen. Evol.* **1** (1992) 53–58

- [5] Graham, R.L., Foulds, L.R.: Unlikelihood that minimal phylogenies for a realistic biological study can be constructed in reasonable computational time. *Math. Biosci.* **60** (1982) 133–142
- [6] Chen, D., Eulenstein, O., Fernández-Baca, D., Sanderson, M.: Supertrees by flipping. Technical Report TR02-01, Department of Computer Science, Iowa State University (2001)
- [7] Semple, C., Steel, M.: A supertree method for rooted trees. *Disc. Appl. Math.* **105** (2000) 147–158
- [8] Piel, W.: Phyloinformatics and tree networks In: Wu, C.H., Wang, P., Wang, J.T.L. (eds.): *Computational Biology and Genome Informatics*. World Scientific Press (2001)
- [9] Adams, E.N.: N -trees as nestings: complexity, similarity, and consensus. *J. Classif.* **3** (1986) 299–317
- [10] Aho, A.V., Sagiv, Y., Szymanski, T.G., Ullman, J.D.: Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions. *SIAM J. Comput.* **10** (1981) 405–421
- [11] Bryant, D., Steel, M.: Extension operations on sets of leaf-labelled trees. *Adv. Appl. Math.* **16** (1995) 425–453
- [12] Ng, M. P., Wormald, N. C.: Reconstruction of rooted trees from subtrees. *Disc. Appl. Math.* **69** (1996) 19–31
- [13] Picard, J.-C., Queryanne, M.: On the structure of all minimum cuts in a network and applications. *Math. Prog. Study* **13** (1980) 8–16
- [14] Stoer, M., Wagner, F.: A simple min cut algorithm. *J. ACM* **44** (1997) 585–591
- [15] Maddison, W. P.: Reconstructing character evolution on polytomous cladograms. *Cladistics* **5** (1989) 365–377
- [16] Slowinski, J.B. Molecular polytomies. *Mol. Phylogen.Evol.* **19** (2001) 114–120
- [17] Steel, M.: The complexity of reconstructing trees from qualitative characters and subtrees. *J. Classif.* **9** (1992) 91–116
- [18] Nelson, G., Platnick, N.I. Multiple branching in cladograms: two interpretations. *Syst. Zool.* **29** (1980) 86–91
- [19] Day, W.H.E.: The role of complexity in comparing classifications. *Math. Biosci.* **66** (1983) 97–114
- [20] Farach, M., Przytycka, T.M. and Thorup, M.: On the agreement of many trees. *Info. Proc. Letters* **55** (1995) 297–301
- [21] Swofford, D.L.: When are phylogeny estimates from molecular and morphological data incongruent? In: Miyamoto, M. M., Cracraft, J. (eds.): *Phylogenetic analysis of DNA sequences*. Oxford University Press, New York (1991) 295–333
- [22] Day, W.H.E.: Analysis of quartet dissimilarity measures between undirected phylogenetic trees. *Syst. Zool.* **35** (1986) 325–333
- [23] Wilkinson, M., Thorley, J.L., Littlewood, D.T.J., Bray, R.A.: Towards a phylogenetic supertree of Platyhelminthes? In: Littlewood, D.T.J., Bray, R.A. (eds.): *Interrelationships of the Platyhelminthes*. Taylor and Francis, London (2001) 292–301
- [24] Joshi, A.K.: An introduction to tree adjoining grammars In: Manaster-Ramer, A. (ed.): *Mathematics of Language*. John Benjamins Publishing Co., Amsterdam (1987) 87–115
- [25] de Queiroz, K., Gauthier, J.: Phylogenetic taxonomy. *Ann. Rev. Ecol. Syst.* **23** (1992) 449–480
- [26] Page, R.D.M.: Extracting species trees from complex gene trees: reconciled trees and vertebrate phylogeny. *Mol. Phylogen. Evol.* **14** (2000) 89–106

- [27] Constantinescu, M., Sankoff, D.: Tree enumeration modulo a consensus. *J. Classif.* **3** (1986) 349–356
- [28] Douchette, C.R.: An efficient algorithm to compute quartet dissimilarity measures. BSc(Hons) dissertation. Department of Computer Science, Memorial University, Newfoundland, Canada (1985)