Supertree Algorithms for Ancestral Divergence Dates and Nested Taxa

Charles Semple[1], Philip Daniel[1], Wim Hordijk[1], Roderic D. M. Page[2], and Mike Steel[1]

[1] Biomathematics Research Centre, Department of Mathematics and Statistics, University of Canterbury, Christchurch, New Zealand
[2] DEEB, IBLS, Graham Kerr Building, University of Glasgow, Glasgow G12 8QP, United Kingdom

Abstract. Motivation: Supertree methods have been often identified as a possible approach to the reconstruction of the 'Tree of Life'. However, a limitation of such methods is that, typically, they use just leaf-labelled phylogenetic trees to infer the resulting supertree.
Results: In this paper, we describe several new supertree algorithms that extend the allowable information that can be used for phylogenetic inference. These algorithms have been recently implemented and we describe here two illustrative applications.
Availability: These new algorithms are freely available for application at `http://darwin.zoology.gla.ac.uk/cgi-bin/build.pl`
Contact: c.semple@math.canterbury.ac.nz

1 Introduction

Rooted phylogenetic trees are used in evolutionary biology to represent the ancestral history of a collection of present-day species. For example, ignoring the numerals, Fig. 1 shows a rooted phylogenetic tree where the labels a, b, c, d, e, and f represent the present-day species. A *supertree* is a rooted phylogenetic tree that is the result of combining a collection of smaller rooted phylogenetic trees on overlapping subsets of species. There are now many techniques ('supertree methods') for constructing supertrees. However, for almost all of these methods, the input is restricted to leaf-labelled phylogenetic trees (without branch lengths or interior node labels or dates), in which case any other related information is ignored. In this paper, we describe several new, and recently implemented, supertree methods which extend the typical input to include some of this additional information.

The new supertree methods divide into two groups depending upon the type of additional information being used as input. The first group allows the input to include ancestral divergence dates which may be either relative or explicit.

For example, in this group the input could include information such as whether one particular divergence event on one side of a tree occurred before or after a divergence event on the other side of the tree, or actual time estimates of certain divergence events. The second group of supertree algorithms takes as its input rooted trees in which some of the interior vertices as well as all of their leaves are labelled. This allows the inclusion of nested taxa in the input.

All of the new supertree algorithms described in this paper have been implemented in Java (and are thus platform independent) and are available for applications at

`http://darwin.zoology.gla.ac.uk/cgi-bin/build.pl`

The implementation requires the input trees to be given in the Newick format[1]. Any tree that is returned by these algorithms is also in the Newick format. All of the algorithms run quickly (that is, require just polynomial time) in the size of the input.

The notation and terminology in this paper follows Semple and Steel (2003). The paper is organised as follows. Each of the new algorithms can be viewed as a variation of BUILD, one of the oldest supertree algorithms. Indeed, the general approach used in each of the algorithms is similar to that used by BUILD. This approach is outlined in the next section. In Sections 3 and 4, we describe the new algorithms for ancestral divergence dates and nested taxa, respectively, and include two applications of these algorithms to some data sets.

We end this section by noting that the formal details of the algorithms described in this paper, including their correctness, are not included here as these details are to appear as chapters (Bryant *et al.*, 2004; Daniel and Semple, 2004) of a forthcoming book on supertrees (Bininda-Emonds, 2004).

2 The BUILD Approach

Originally designed for other purposes, BUILD (Aho *et al.*, 1981) is an exact algorithm in that it outputs a tree precisely if the input collection satisfies a particular compatibility criteria. A rooted phylogenetic tree $\mathcal{T}$ *displays* a rooted phylogenetic tree $\mathcal{T}'$ if the label set X' of $\mathcal{T}'$ is a subset of the label set of $\mathcal{T}$ and, up to suppressing degree-two vertices, $\mathcal{T}'$ is a refinement of the minimal rooted subtree of $\mathcal{T}$ that connects the labels in X'. For the purposes of this paper, 'up to suppressing degree-two vertices' essentially means 'overlooking degree-two vertices'. A collection $\mathcal{P}$ of rooted phylogenetic trees is *compatible*

[1] See for example `http://evolution.genetics.washington.edu/phylip/newicktree.html`

if there exists a rooted phylogenetic tree that displays each of the trees in $\mathcal{P}$. Intuitively, $\mathcal{P}$ is compatible if there is a rooted phylogenetic tree that, up to polytomies, preserves all of the ancestral relationships described by the trees in $\mathcal{P}$. In particular, if the most recent common ancestor of a and b is a descendant of the most recent common ancestor of a and c in a tree in $\mathcal{P}$, then this relationship is also preserved in $\mathcal{T}$.

The algorithms we present in this paper follow the approach of Build. Each algorithm is exact and outputs a tree precisely if the input collection satisfies some particular compatibility criteria. Furthermore, like Build, the descriptions of the algorithms take the following form. Each algorithm attempts to construct a tree $\mathcal{T}$ that satisfies the compatibility criteria by constructing the set of clusters of $\mathcal{T}$. In all cases, this is done by starting with the cluster that is the union of the labels of the trees in $\mathcal{P}$ and successively breaking it down into disjoint subclusters. How the clusters are broken down is determined by an associated graph at each iteration. For each algorithm, this graph as well as the process for breaking up clusters is different. The process continues provided the graph at each iteration satisfies some condition. Eventually, either

(i) the algorithm outputs a tree that satisfies the compatibility criteria, or
(ii) outputs a statement indicating that the input collection does not satisfy this criteria.

3 Supertree Algorithms for Ancestral Divergence Dates

We first describe a supertree algorithm that incorporates relative divergence times. This algorithm is called RankedTree. An extension of this algorithm to include absolute divergence times or intervals on these times is also possible and this is mentioned at the end of this section.

Essentially, RankedTree is an extension of Build and its input consists of rooted phylogenetic trees as well as information detailing the order in which the divergence events of certain different pairs of species occurred. We call the latter type of input a *relative divergence date* and such information is based, for example, on fossil data or molecular dating techniques. Formally, this type of input takes the form '$\mathrm{div}(c, d)$ predates $\mathrm{div}(a, b)$' which is interpreted as, 'for species a, b, c, and d, the divergence of species c and d predates the divergence of species a and b'.

To include both types of input on a single supertree, we extend the concept of a rooted phylogenetic tree. A *ranked phylogenetic tree* $\mathcal{T}$ is a rooted phylogenetic tree in which the interior vertices are assigned a positive integer so that if v_1, v_2 are interior vertices and v_2 is a descendant of v_1, then the integer assigned to v_1 is less than the integer assigned to v_2. Such an assignment of positive integers is a

called a *ranking* of the interior vertices of $\mathcal{T}$. An example of a ranked phylogenetic tree is shown in Fig. 1. Ranking the interior vertices of $\mathcal{T}$ in this way corresponds to an ordering of the speciation events associated to these vertices. Note that two different interior vertices may be assigned the same positive integer, in which case, it is inferred that there is no particular ordering on the associated speciation events.

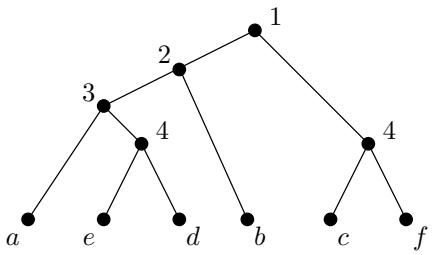

Fig. 1. A ranked phylogenetic tree.

A relative divergence date '$\mathrm{div}(c, d)$ predates $\mathrm{div}(a, b)$' is *preserved* by a ranked phylogenetic tree $\mathcal{T}$ if a, b, c, d are leaf labels of $\mathcal{T}$, and the rank assigned to the interior vertex of $\mathcal{T}$ corresponding to the most recent common ancestor of c and d is less than the rank assigned to the interior vertex of $\mathcal{T}$ corresponding to the most recent common ancestor of a and b. Thus, for example, the ranked phylogenetic tree shown in Fig. 1 preserves the relative divergence date '$\mathrm{div}(e, b)$ predates $\mathrm{div}(c, f)$'. A collection $\mathcal{P}$ of rooted phylogenetic trees and a collection $\mathcal{D}$ of relative divergence dates are *compatible* if there is a ranked phylogenetic tree $\mathcal{T}$ such that the discrete topology of $\mathcal{T}$ displays each of the trees in $\mathcal{P}$ and the ranking of the interior vertices of $\mathcal{T}$ preserves all of the relative divergence dates in $\mathcal{D}$.

The algorithm RankedTree decides whether or not collections of rooted phylogenetic trees and relative divergence dates are compatible. Furthermore, if these collections are compatible, then RankedTree returns a ranked phylogenetic tree that displays each of the rooted phylogenetic trees and preserves each of the relative divergence dates. To illustrate RankedTree, consider the two ranked phylogenetic trees shown in Fig. 2(a) (Janczewski *et al.*, 1995) and (b) (Slattery *et al.*, 1994), each of which is a phylogenetic tree of the cat family. The species labels are the 3-letter abbreviations used in these references. The branch lengths of the source trees have been translated into rankings and added to the interior vertices of these trees. (These branch lengths are also shown.) Observing that species 'LPA', 'PON', and 'CCR' are common to both trees, the ranked phylogenetic tree shown in Fig. 2(c) is the result of applying RankedTree to these two trees. Note that the branch lengths of the resulting ranked phylogenetic tree do not reflect real time.

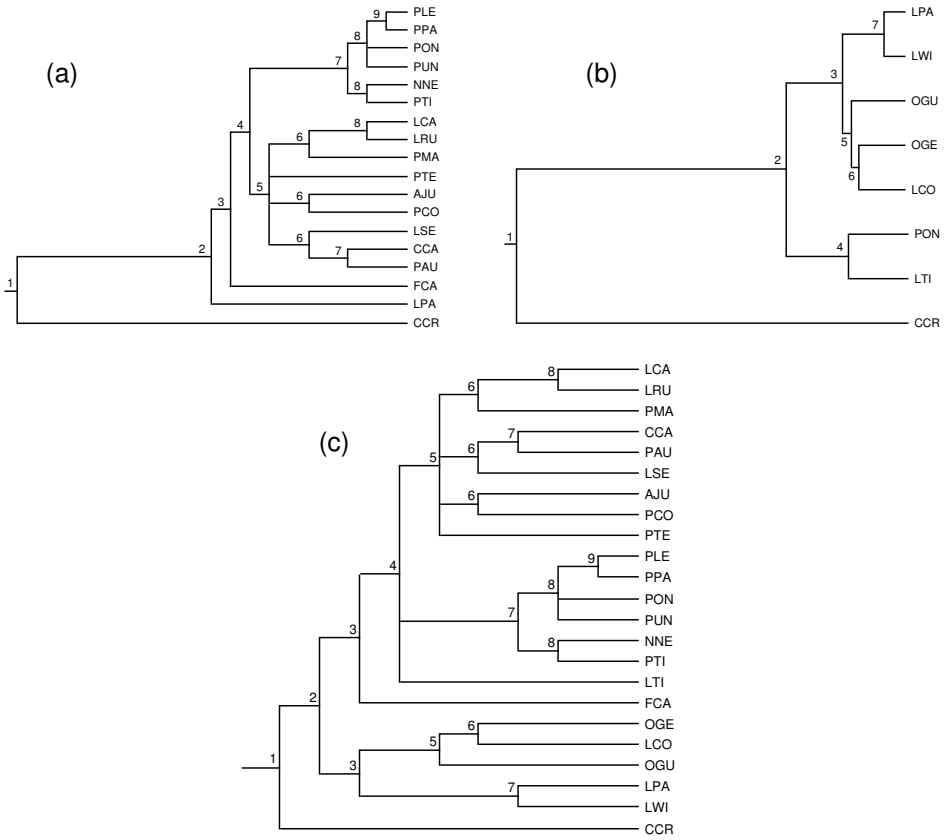

Fig. 2. An application of RankedTree.

An extension of RankedTree allows for time bounds on speciation events as well as rooted phylogenetic trees and relative divergence dates in its input. A *divergence time bound* for species a and b is either an upper or lower bound (or both) on the number of years ago a and b diverged. To include this information as well as the other information provided by the other inputs, we use a *dated phylogenetic tree*. Such a tree is similar to that of a ranked phylogenetic tree in that values are assigned to the interior vertices except that these values now represent the number of years ago the corresponding speciation events occurred. Compatibility for this extended input is defined in the obvious way and RankedTree can be modified to solve this compatibility problem also.

4 Supertree Algorithms for Nested Taxa

For supertree algorithms that take as their input collections of rooted phylogenetic trees and only return rooted trees that are leaf-labelled, it is implicit that, as a whole, the leaves of the trees in the input collection represent non-nested taxa. Thus, for example, *Rattus rattus* and 'mammal' cannot be represented by two distinct leaves in such a collection as the former is nested inside the latter. This is somewhat limiting in the choice of trees for our input collection. In this section, we describe two supertree algorithms for combining rooted trees in which all of the leaves as well as some of the interior vertices are labelled. These trees are called *rooted semi-labelled trees* and the interior labels of such trees represent taxa at a level higher than that of their descendants. Two semi-labelled trees are shown in Fig. 3.

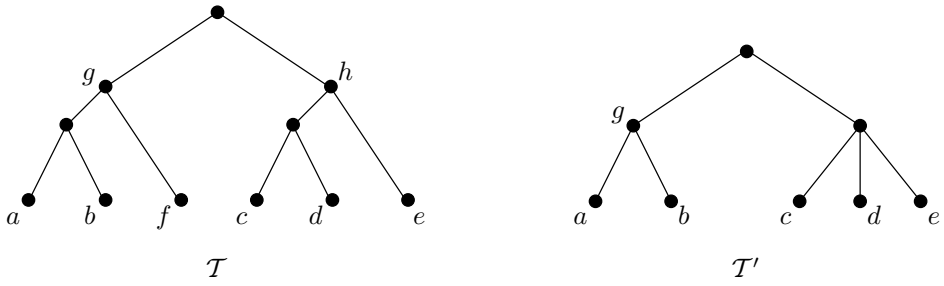

Fig. 3. Two semi-labelled trees.

The two algorithms for combining collections of rooted semi-labelled trees are called Semi-LabelledBuild and AncestralBuild. Both algorithms allow a leaf of one of the input trees to represent a taxa that is represented by an interior label of another tree. The motivation for both algorithms came from a problem posed by Page (2004).

4.1 Semi-LabelledBuild

We say that a rooted semi-labelled tree $\mathcal{T}$ *perfectly displays* a rooted semi-labelled tree $\mathcal{T}'$ if the label set X' of $\mathcal{T}'$ is a subset of the label set of $\mathcal{T}$ and, up to suppressing degree-two vertices, $\mathcal{T}'$ is the minimal rooted subtree of $\mathcal{T}$ that connects the labels in X'. Intuitively, $\mathcal{T}$ perfectly displays $\mathcal{T}'$ if $\mathcal{T}$ preserves all of the ancestral relationships described by $\mathcal{T}'$ exactly. In particular, $\mathcal{T}$ preserves all of the most recent common ancestor relationships described by $\mathcal{T}'$. A collection $\mathcal{P}$ of rooted semi-labelled trees is *perfectly compatible* if there is a rooted semi-labelled tree $\mathcal{T}$ that perfectly displays each of the trees in $\mathcal{P}$.

For a collection $\mathcal{P}$ of rooted semi-labelled trees, Semi-LabelledBuild decides whether or not $\mathcal{P}$ is perfectly compatible. Moreover, if $\mathcal{P}$ is perfectly compatible, then Semi-LabelledBuild returns a rooted semi-labelled tree that perfectly displays each of the trees in $\mathcal{P}$. Figure 4 shows an application of Semi-LabelledBuild. The input consists of the two rooted semi-labelled trees shown in Fig. 4(a) and (b). Both input trees describe the evolution of spiders and were obtained from study S1x6x97c14c42c30 in TreeBASE. There are taxa common to both trees and it is of particular interest to note that the taxon *Araneoclada* labels a leaf in the tree in (a), but an interior vertex in the tree in (b). The rooted semi-labelled tree resulting from applying Semi-LabelledBuild to the two input trees is shown in Fig. 4(c).

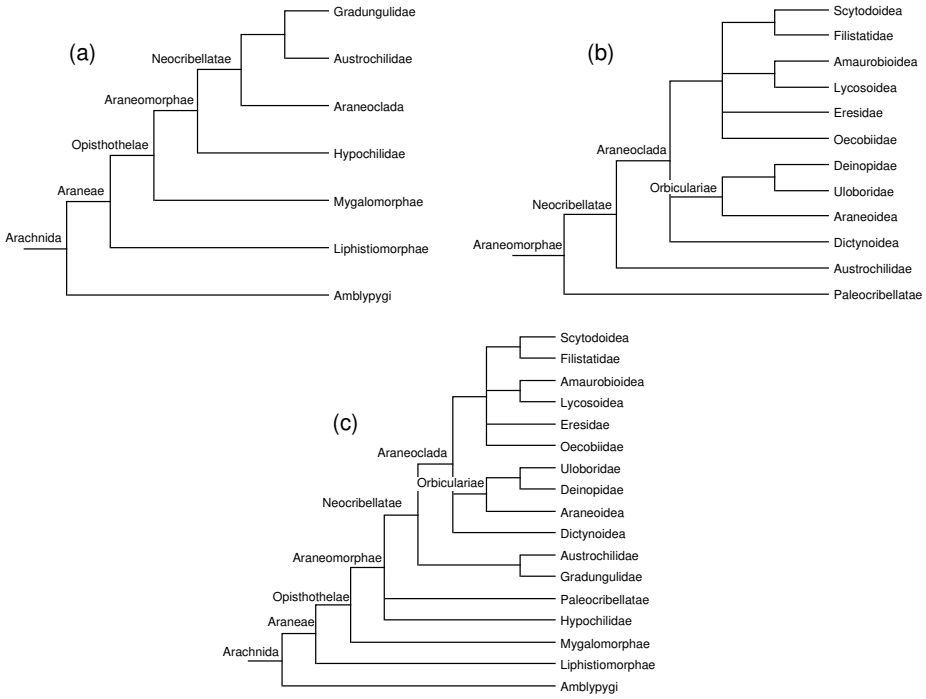

Fig. 4. An application of Semi-LabelledBuild.

4.2 AncestralBuild

The criteria of perfectly displays is very strong as a collection $\mathcal{P}$ of rooted semi-labelled trees is perfectly compatible precisely if there is a rooted semi-labelled

tree $\mathcal{T}$ that preserves all of the most recent common ancestor relationships described by the collection. Thus Semi-LabelledBuild does not allow for the resolution of polytomies. The compatibility criteria and the associated algorithm we describe next relaxes this criteria and allows for the resolution of polytomies.

A rooted semi-labelled tree $\mathcal{T}$ *ancestrally displays* a rooted semi-labelled tree $\mathcal{T}'$ if the following properties hold:

(i) the label set X' of $\mathcal{T}'$ is a subset of the label set of $\mathcal{T}$;
(ii) up to suppressing degree-two vertices, $\mathcal{T}'$ is a refinement of the minimal rooted subtree of $\mathcal{T}$ that connects the labels in X'; and
(iii) for all labels in X',
  (a) if a is a proper ancestor of b in $\mathcal{T}'$, then a is a proper ancestor of b in $\mathcal{T}$, and
  (b) if a is neither an ancestor nor a descendant of b in $\mathcal{T}'$, then a is neither an ancestor nor a descendant of b in $\mathcal{T}$.

To illustrate ancestrally displays and compare it with perfectly displays, in Fig. 3, $\mathcal{T}$ ancestrally displays $\mathcal{T}'$, but $\mathcal{T}$ does not perfectly display $\mathcal{T}'$.

One can think of ancestrally displays as preserving all of the ancestor-descendant relationships. However, observe that it does not preserve the most recent common ancestor relationships. For example, if the most recent common ancestor of a and b is c in $\mathcal{T}'$, then, although c is an ancestor of both a and b in $\mathcal{T}$, and neither a nor b is an ancestor of each other in $\mathcal{T}$, c need not be the most recent common ancestor of a and b in $\mathcal{T}$. A collection $\mathcal{P}$ of rooted semi-labelled trees is *ancestrally compatible* if there is a rooted semi-labelled tree $\mathcal{T}$ that ancestrally displays each of the trees in $\mathcal{P}$. The algorithm AncestralBuild determines the ancestral compatibility of a collection of rooted semi-labelled trees, in which case, it outputs a rooted semi-labelled tree that ancestrally displays each of the trees in this collection.

5 Conclusion

Supertree methods have attracted much interest recently, particularly in the light of well-funded 'Tree of Life' initiatives, and studies that have combined large numbers of trees to construct phylogenies on hundreds, or even thousands of species. This has led to some vigorous argument both for and against the use of supertree (versus supermatrix) approaches for phylogeny reconstruction, as well as the emergence of some new techniques as alternatives to the standard MRP (matrix recoding with parsimony) approach. In this paper, we have described some further new methods, that allow for additional types of information to

be incorporated—information that would be difficult to include directly using a traditional MRP analysis.

It is important to note that all of the algorithms described in this paper are 'all-or-nothing' algorithms. Each algorithm either returns a supertree with certain desirable properties relative to the input or returns a statement indicating that there is no such supertree. In practice, this limits their use. However, such algorithms are important first steps in developing supertree algorithms that always return a supertree and whose input includes information that goes beyond leaf-labelled phylogenetic trees. Indeed, for each of the algorithms described in this paper, we are currently developing MinCutSupertree type algorithms (Page, 2002; Semple and Steel, 2000) that overcomes this limitation.

Lastly, if a supertree is returned by one of the algorithms described in this paper, two natural questions arise: (i) how many such supertrees are there and (ii) what common information is carried by all of these supertrees? In the case of (i), if there are too many, one may want to refine the original data to reduce this number. However, it is an immediate consequence of the main result by Bordewich *et al.* (in press) that determining this number exactly is #P-complete for all of the algorithms described in this paper. For (ii), Daniel (2004) has investigated this problem in the case where the input consists of just rooted phylogenetic trees. An approach similar to that taken by Daniel could be used for the algorithms described in this paper.

Acknowledgements. The second author was supported by the New Zealand Institute of Mathematics and its Applications funded programme *Phylogenetic Genomics* and the first and last authors were supported by the New Zealand Marsden Fund.

References

1. Aho, A. V., Sagiv, Y., Szymanski, T. G., and Ullman, J. D. (1981). Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions, *SIAM Journal on Computing*, **10**, 405-421.
2. Bininda-Emonds, O. R. P., ed. *Phylogenetic supertrees: combining information to reveal the Tree of Life*, in press.
3. Bordewich, M., Semple, C., and Talbot, J. Counting consistent phylogenetic trees is #P-complete, *Advances in Applied Mathematics*, in press.
4. Bryant, D., Semple, C., and Steel, M. Combining evolutionary trees with ancestral divergence dates. In O. Bininda-Emonds (ed.), *Phylogenetic supertrees: combining information to reveal the Tree of Life*, Computational Biology Series, Kluwer, in press.
5. Daniel P. (2004). *Supertree Methods: Some New Approaches*, MSc thesis, University of Canterbury.
6. Daniel, P. and Semple, C. Supertree algorithms for nested taxa. In O. Bininda-Emonds (ed.), *Phylogenetic supertrees: combining information to reveal the Tree of Life*, Computational Biology Series, Kluwer, in press.

7. Janczewski, D. N., Modi, W. S., Stephens, J. C., and O'Brien, S. J. (1995). Molecular Evolution of Mitochondrial 12S RNA and Cytochrome b Sequences in the Pantherine Lineage of Felidae, *Molecular Biology and Evolution*, **12**, 690-707.
8. Page, R. D. M. (2002). Modified mincut supertrees. In R. Guigó and D. Gusfield (eds.), Proceedings of the Second International Workshop on *Algorithms in Bioinformatics* (*WABI* 2002), pp.537-552, Springer.
9. Page, R. D. M. Taxonomy, Supertrees, and the Tree of Life. In O. Bininda-Emonds (ed.), *Phylogenetic supertrees: combining information to reveal the Tree of Life*, Computational Biology Series, Kluwer, in press.
10. Semple, C. and Steel, M. (2000). A supertree method for rooted trees, *Discrete Applied Mathematics*, **105**, 147-158.
11. Semple, C. and Steel, M. (2003). *Phylogenetics*, Oxford University Press.
12. Slattery, J. P., Johnson, W. E., Goldman, D., O'Brien, S. J. (1994). Phylogenetic Reconstruction of South American Felids Defined by Protein Electrophoresis, *Journal of Molecular Evolution*, **39**, 296-305.