A new recursive algorithm to count the number of combinations of two phylogenetic trees

Elisa Kasbohm - March 25, 2015
Joint Work with Ina Maria Deutschmann

Allan Wilson Centre & Biomathematics Research Centre, University of Canterbury
SUPERVISORS: Mike Steel, Charles Semple, Magnus Bordewich

KEYWORDS: *phylogenetic tree, supertree, combinatorial algorithm*

1 MOTIVATION

Phylogeny is a field of evolutionary biology that is particularly focused on analysing the ancestry of species and reconstructing the branching processes that led to the formation of new taxa (*speciation*). Most commonly, analyses are based on sequence data for a distinct set of genes or proteins derived from sequencing DNA or RNA and protein samples, respectively. Similarities between the sequences indicate a closer relationship, hence a hypothesis about the relationships among the taxa can be deduced from the sequence data. Finally, the findings are summarised and illustrated by a *phylogenetic tree*. However, as the genetic make-up of species in general and additionally the protein expression in distinct tissues as well as under changing test conditions differs enormously it can be necessary to combine the results of several analyses that also may include different sets of taxa. This leads to the problem of combining phylogenetic trees.

For merging a set of phylogenetic (source) trees with overlapping taxa into a single, bigger parent tree, *supertree methods* are the methods of choice. A *supertree* denotes here the resulting parent tree of the merged source trees. One big advantage of these methods is that the source trees need not come from the same type of data or the same analysis. The role of supertree methods in evolutionary biology and the variety of ways in which these methods are used are highlighted in [1]. Nevertheless, assuming that the source trees can be combined to a supertree without causing contradictions to one or more of the source trees, there will be in general more than one possible configuration for a supertree. The supertree should ideally display the true evolution of the taxa, so if the number of possible supertrees is very large, then the given set of source trees contains only little information about the true relationships between the species. Further on, it is shown in [2] that counting the number of supertrees for a set of arbitrary source trees is *#P-complete*, which means that it is very unlikely that there exists an algorithm to compute the number of possible supertrees in polynomial time. However, this is not the case when considering only two source trees, for example [3] and [4] give two different algorithms that return all possible supertrees in polynomial time. These algorithms generate all possible configurations of supertrees, that's why they are not very efficient and take relatively long computing time if one is only interested in the number of supertrees and not in the supertrees themselves. Thus, the aim of this project was to investigate a new recursive approach for determining the number of supertrees of two rooted binary trees in polynomial time without returning the structures of all the possible supertrees. This new algorithm facilitates the computations in significantly less time than before and enables further investigations on the characteristics of the number of supertrees.

2 A SHORT SUMMMARY OF THE RECURSIVE ALGORITHM

For two rooted binary trees on arbitrary leaf sets first a backbone tree is built on all common leaves. All remaining subtrees will be attached to the edges of this base frame. The topologies of the two given source trees already define to which edge of the backbone tree the residual subtrees have to be attached, respectively, but the relative positions of the subtrees of the two given trees to each other are variable. So for the recursion let the position of the first i subtrees of the frist tree for a distinct edge of the backbone tree and the position of the first j subtrees of the second tree for the same edge be fixed. Then the number of combinations of the remaining subtrees has to be calculated which means to determine the number of supertrees for two rooted binary trees on disjoint leaf sets. This will be repeated for every i and j.

For counting the number of supertrees of two phylogenetic trees with disjoint leaf sets the recursion splits the given source trees on the root into a right and a left subtree. Now either the two trees are combined by defining a new root and inserting the given source trees as right and left subtree or the first subtree is merged with the right or the left part of the second tree and vice versa or the two subtrees of the first tree are merged independently with the two subtrees of the second tree. When considering the subtrees this will invoke a new recursive call that will now split the subtrees on internal nodes unless at least one of the two 'trees' is a singleton.

This recursion can even be more simplified for caterpillar trees and balanced trees by considering only the number of leaves, respectively, and using the given structure. Further improvements by means of dynamic programming can even refrain completey from recursive calls for these special cases by memorising only the required intermediate steps.

The recursive algorithm for counting the supertrees of two phylogenetic trees on disjoint leaf sets was implemented in C++. Additionally, to improve the user-friendliness and to simplify the interpretation of the results, a graphical user interface was created in R by means of the package 'shiny' and added to the programme. In fact, our algorithm for this special case calculated the number of supertrees in neglectable time whereas the SUPERB algorithm [3] needed several hours.

3 FURTHER FINDINGS

When comparing the calculated numbers of supertrees for different tree shapes and sizes of leaf sets we recognized that the result actually depends on the shape of the tree. We observed that two trees of the same structure always give more combinations than two trees of different structures. Also, the highest number of combinations will be achieved for two balanced trees whereas two caterpillar trees will always give less supertrees. Moreover, when generating trees randomly according to the Yule-Harding model the spread of the calculated numbers of supertrees increases stronger than the number of leaves in the supertree.

4 DISCUSSION

For further improvement of the algorithm, it would be helpful to memorise some of the recursive calls that are needed repeatedly. This will save many recursive calls and speed up the runtime of the programme. Moreover, a special class or library for enabling arbitrary-precision arithmetic should be included, as so far numbers bigger than $9 \cdot 10^{19}$

can't be handled. These modifications can be included with little effort and will improve the perfomance of the algorithm significantly. The next step would be to implement the algorithm for rooted binary trees on non-disjoint leaf sets.

5 Acknowledgements

I am very grateful to my supervisors Mike Steel, Charles Semple and Magnus Bordewich for their extensive support during my internship at the University of Canterbury. Additionally, I wish to thank the Allan Wilson Centre for providing funding for this project.

6 References

[1] O. R. P. Bininda-Emonds. *Phylogenetic supertrees: combining information to reveal the tree of life*. Kluwer, Dordrecht, 2004.

[2] M. Bordewich, C. Semple, and J. Talbot. Counting consistent phylogenetic trees is #P-complete. *Advances in Applied Mathematics*, 33(2):416–430, 2004.

[3] M. Constantinescu and D. Sankoff. An efficient algorithm for supertrees. *Journal of Classification*, 12(1):101–112, 1995.

[4] M. P. Ng and N. C. Wormald. Reconstruction of rooted trees from subtrees. *Discrete Applied Mathematics*, 69(1):19–31, 1996.