Species Trees and the Ultrametric Constraint

Neil Moore

July 27, 2007

This describes work carried out for my 4th and final year project during my BSc(Hons) degree in Computing Science at the University of Glasgow.

Overview The theory of evolution suggests that all species, living and dead, are related. Biologists typically depict this in tree diagrams. Living species number in the millions and current approaches to combining them into one monolithic tree of life are defeated by size. However, bioinformaticians have recently been combining manageable fragments to obtain increasingly complete approximations to the tree of life. This has a subjective element and current algorithms do not capture all of biologist's requirements in one place. In this project I apply constraint programming techniques to obtain a solution to the supertree problem that is sufficiently fast and flexibile.

Evolution and the tree of life Darwin's theory of evolution by natural selection is an explanation of how complex species could arise from simpler antecedents. The basic theory is that within species individuals have different characteristics (or *adaptations*) due to genetic mutation. Those adaptations that help their possessors to survive and reproduce will be present in a higher proportion in the next generation compared to unhelpful adaptations. This is by virtue of the fact that children inherit their parents' genes and hence their adaptations. Speciation occurs as a result of limited mating whereby adaptations are no longer spread approximately uniformly through the gene pool and hence certain subgroups of the species can develop certain adaptations in greater proportion, perhaps as a result of the formation of mountain barriers or new islands.

Such speciation can be drawn as a rooted bifurcating tree (a *species tree* in the biological parlance), where internal nodes represent ancestral species and leaves represent new species[1]. See Figure 1 for an example of this sort of tree.

[1]This representation is a subject of controversy amongst biologists, as there is evidence to believe that genetic material has jumped from subtree to subtree (lateral gene transfer), leaving a DAG rather than a tree, however the tree paradigm is well established and I will not discuss the controversy any further.

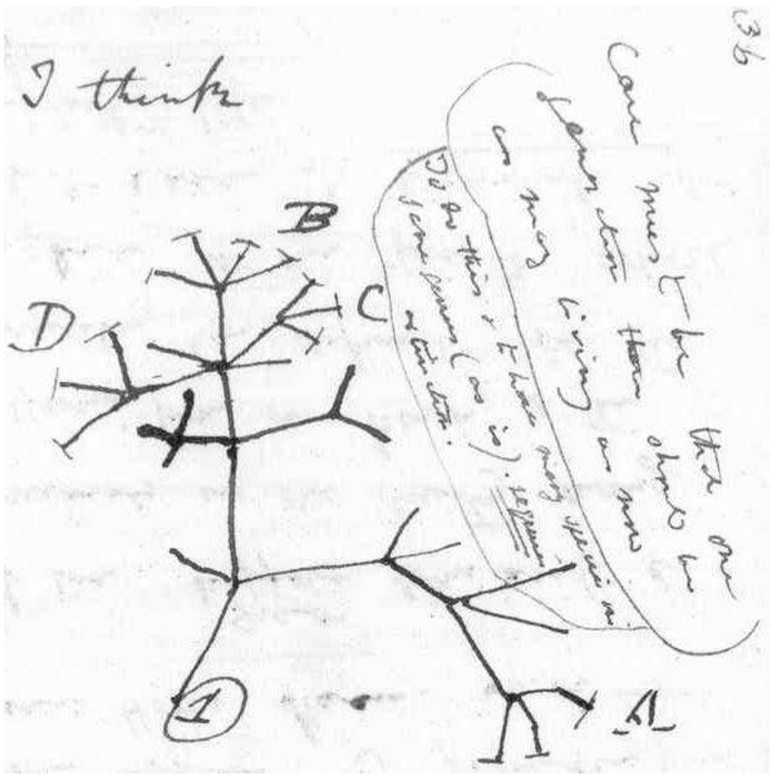
36
I think
Case must be that one generation then should be as many living as now.
To do this & to have many species in same genus (as is). requires extinction.
B
C
D
1
A

Figure 1: Charles Darwin's first sketch of an evolutionary tree from his *First Notebook on Transmutation of Species* (1837)

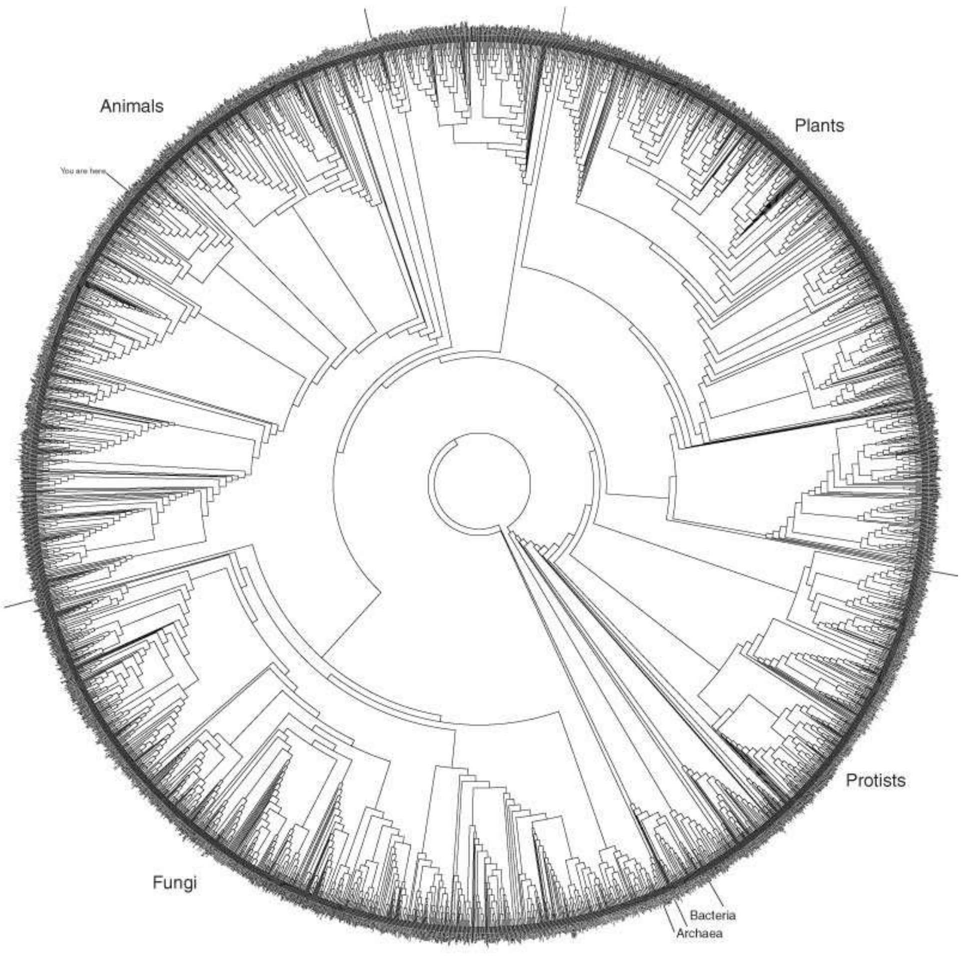

Figure 2: Species tree involving 3000 species and derived from DNA data by David M. Hillis, Derrick Zwickl, and Robin Gutell, University of Texas

The tree of life[Pen03] is the tree that contains all species. The derivation of such a tree could involve data from DNA analyses, morphology, physiology, etc. However the task of deriving the tree of life is computationally and organisationally too hard to be done by a single method[Pag04]. David Hillis has succeeded in creating a tree from the DNA of 3000 species, as shown in Figure 2, but this deals with only a small fraction of the millions of existing species and the prospects of a complete solution seems unlikely on account of the fact that the problem is NP-complete[Gus97].

For this reason we need a hybrid approach of combining fragments of the tree efficiently and in a biologically meaningful way. One such method is using supertrees.

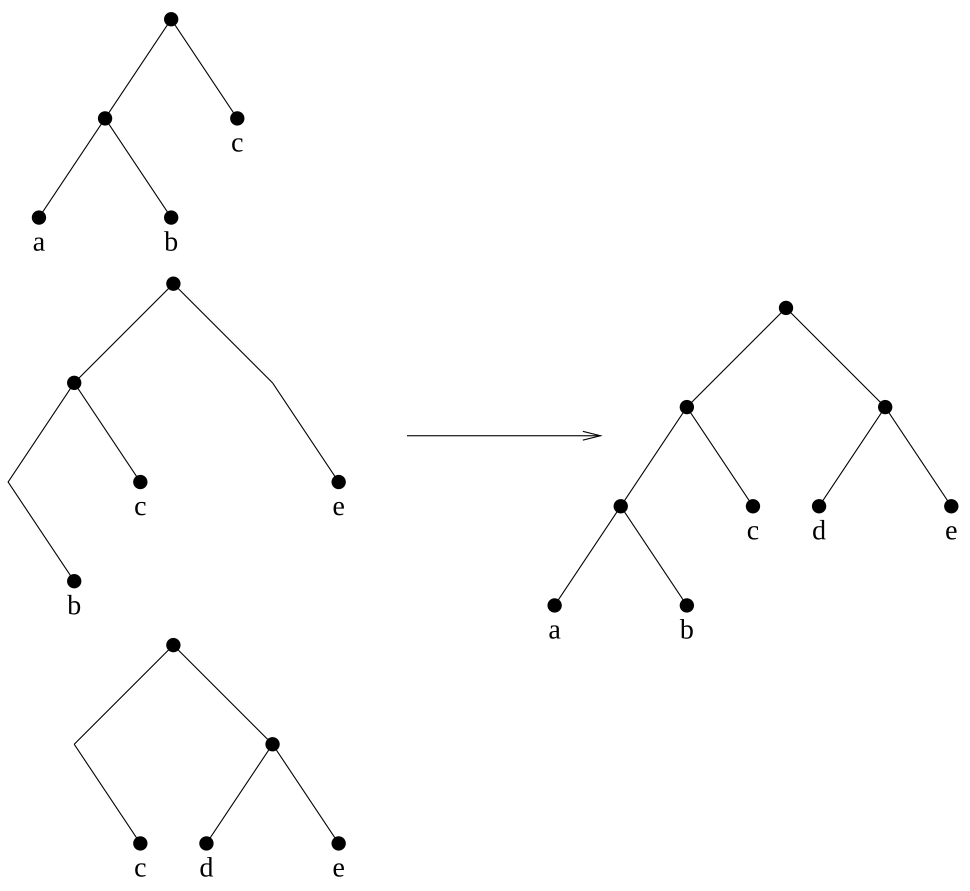

Figure 3: Instance (left) and solution (right) to supertree problem

The supertree problem A supertree[Pag04] for a set of input trees is a tree that contains all the species of the input and also preserves all ancestral relationships, e.g., the fact that mice and rats are more closely related to each other than either is to pigs, daffodils or e-coli bacteria. An example instance and solution to the supertree problem is shown in Figure 3. On the left a collection of input trees are drawn and on the right is a solution. All input species are present in the solution, and, in the case of the 3rd input, species d and e remain more closely related to each other than to c.

The first solution to this problem was published in [ASSU81] in 1981 in the context of database technologies. Since then related algorithms have been published in the bioinformatics literature to solve the same[NW96] and related problems[SS00, SDH$^+$04, Dan03]. These algorithms run in polynomial time and have been successfully applied in practical situations to derive supertrees (e.g., [KP02]).

Constraint solution to the supertree problem A drawback of these algorithms is that, despite their computational efficiency, they are purpose built to solve one particular problem. The task of deriving them and proving them correct is difficult and only a small number of people are able to do it (some of their names appear multiple times in my bibliography!). Constraint programming is an alternative strategy for solving the problem that was first tried in [GPSW03]. In constraint programming, problems are described in a relatively high-level, declarative way (a constraint program) and from this description a generic solver finds solutions to particular instances. This is done by choosing a set of variables and a set of constraints that constrain the values that the variables can simultaneously take. Provided that the constraints describe all the essential properties of a solution, every solution the solver finds will also be a solution to the problem. Constraint programming technologies offer a number of advantages in this application:

- Related problems can be modelled by the addition of side constraints where new imperative solutions would require a complete rethink. In addition, either one or all solutions can be generated from the same model.
- It is easier to verify the correctness of constraint programs due to their declarative nature and intrinsic readability.
- Models benefit from advances in CP technology. For example, *explanations*[JB00] allow users to discover the reason why no solution exists, as opposed to the imperative solution which merely informs them that this is the case! This flexibility is due to the decoupling of the electronic specification of the problem and the means of solving.

However, CP's flexibility comes at a cost in efficiency and this is the backdrop to my main contribution, which was to reduce the time and space complexity of the CP model of [GPSW03] so that the constraint model now runs in polynomial time and an asymptotic improvement in memory usage has been achieved. The algorithm of [ASSU81] finds a solution in $O(n^2)$, but the CP model of [GPSW03] had a worst case complexity of $O(n^{n^2})$ and at times it exhibited this exponential worse case behaviour. The memory requirements of the model was $O(n^3)$ and this prevented larger instances from being loaded into the memory of a desktop computer. As it turned out, the solution to both of these problems was to create a new design of propagator for one of the constraints that was involved in the model.

Advances in constraint solution Most modern solvers for constraint programs use two techniques: search and propagation. During search the solver works its way systematically through every possible assignment of a values to each variable, in this way it ensures that no solutions are missed

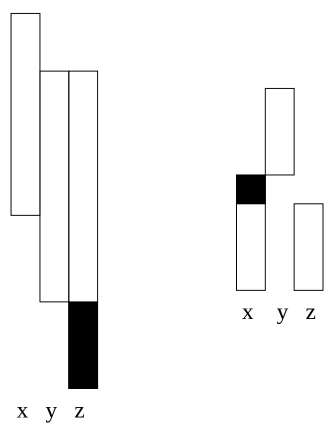

Figure 4: Two examples of maximum ultrametric propagation

or repeated. During search, whenever a variable is assigned a value in a potential solution, the solver will also do propagation. What happens is that values that are incompatible with a part of the current partial solution are removed from further consideration. This avoids a solver trying out values that must inevitably lead to a non-solution and thus wasting its time on dead ends.

A particular constraint, called the *ultrametric* constraint, is used extensively in the constraint model for the supertree problem. For a choice of 3 variables, it constrains two of them to be the same and the other to be at least as large. For example, $x = 1$, $y = 2$ and $z = 1$ is a solution but $x = 1$, $y = 2$, $z = 3$ is not. This constraint is used in the model so that for each choice of 3 species i, j and k their pairwise most recent common ancestor's tree depth is conformant to one possible evolutionary relationship. Letting M_{ij} be the depth of the m.r.c.a. of species i and j, it must be the case that one pair of species, say i and j, are most recently diverged, i.e., $M_{ij} > M_{ik} = M_{ik}$ or that all 3 diverged at the same time, i.e., $M_{ij} = M_{ik} = M_{ik}$.

A propagator for the ultrametric constraint must take as input the current domains of the three variables it is posted over, and return the same three domains but with all impossible values trimmed out. Figure 4 shows two examples of the maximum possible propagation that can be done to 3 variables x, y and z constrained to be ultrametric. In the first, the shaded part of z beneath the other two cannot be part of a solution, because there are no matching values in either of the other domains. In the second, the shaded part of x cannot be tied for least, because no equal value exists, and it cannot be the largest because no lesser values in the other domains tie.

I implemented my design of propagator in Java and then proceeded to prove it correct using a series of rigorous mathematical proofs. One part of this proof turned out to be especially interesting in the context of the overall problem. This was the theorem that I discovered that after the propagator has completed, it is guaranteed that the least remaining value for each of the variables together form a correct instantiation. This property is problem specific and is not be true of all constraints. The practical repercussion of this is that the search stage of constraint solving is now redundant and the entire problem can be solved by first propagating and then taking the lower bound of each variable as the solution. Since propagation is a polynomial time problem, the CP supertree problem has been pushed down from exponential time to tractable polynomial time, $O(n^4)$ to be precise. In his review article on supertrees[Pag04], Page says that there is only one polynomial time solution to the supertree problem; this is no longer correct. Furthermore this breakthrough removes a very valid criticism of the old model that it may be too slow to use.

The other aim of this work was to reduce the memory requirements of the constraint solution. The crux of the problem is that the ultrametric constraint must be posted $C(n,3) = O(n^3)$ times, for an instance involving n species. In most constraint solvers this amounts to $O(n^3)$ space which dominates the space requirements of all the other constraints and variables. By replacing the $O(n^3)$ extensional representation of the set of ultrametric constraints by a $O(1)$ code representation I was able to reduce the overall space usage to $O(n^2)$. In practical instances this amounts to a 290 times space reduction, more for larger instances. Now all foreseeable instances can be loaded into the memory of a desktop computer.

Since this work was completed, I have been tackling a series of variants on the supertree problem which have become accessible now that a polynomial time solution to the original problem has been achieved. This proves that the model is sufficiently versatile that it can be adapted to solve similar challenges without starting from scratch. Furthermore, I believe that we have now progressed to the stage where constraint programming should be considered seriously for any problem in supertrees. I was able to retain the strengths of a CP solution (as described above), but also dispense with the weaknesses in large part.

References

[ASSU81] A.V. Aho, Y. Sagiv, T.G. Szymanski, and J.D. Ullman. Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions. *SIAM J. Comput*, 10(3):405–421, August 1981.

[Dan03] Philip Daniel. Supertree methods: Some new approaches. Master's thesis, Department of Mathematics and Statistics, University of Canterbury, 2003.

[GPSW03] Ian P. Gent, Patrick Prosser, Barbara M. Smith, and Wu Wei. *Supertree Construction with Constraint Programming*, pages 837–841. Principle and Practice of Constraint Programming. Springer, 2003.

[Gus97] Dan Gusfield. *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology.* Cambridge University Press, 1997.

[JB00] Narendra Jussien and Vincent Barichard. The PaLM system: explanation-based constraint programming. In *Proceedings of TRICS: Techniques foR Implementing Constraint programming Systems, a post-conference workshop of CP 2000*, pages 118–133, Singapore, September 2000.

[KP02] M. Kennedy and R.D.M. Page. Seabird supertrees: Combining partial estimates of procellariiform phylogeny. *The Auk*, 69:88–108, 2002.

[NW96] Meei Pyng Ng and Nicholas C. Wormald. Reconstruction of rooted trees from subtrees. *Discrete Appl. Math.*, 69(1-2):19–31, 1996.

[Pag04] Roderic D.M. Page. Taxonomy, supertrees, and the tree of life. In Olaf Bininda-Emonds, editor, *Phylogenetic Supertrees: Combining information to reveal the tree of life*, pages 247–265. Computational Biology Series Kluwer, 2004.

[Pen03] Elizabeth Pennisi. Modernizing the tree of life. *Science*, 300:1692–1697, June 2003.

[SDH$^+$04] Charles Semple, Philip Daniel, Wim Hordijk, Roderic D.M. Page, and Mike Steel. Supertree algorithms for ancestral divergence dates and nested taxa. *Bioinformatics*, 20(15):2355–2360, 2004.

[SS00] Charles Semple and Mike Steel. A supertree method for rooted trees. *Discrete Appl. Math.*, 105(1-3):147–158, 2000.