

Supertree Construction with Constraint Programming: recent progress and new challenges

Patrick Prosser

Department of Computing Science, University of Glasgow, Scotland. pat@dcs.gla.ac.uk

1 Introduction

One goal of biology is to build the *Tree of Life* (ToL), a representation of the evolutionary history of every living thing. To date, biologists have catalogued about 1.7 million species, yet estimates of the total number of species ranges from 4 to 100 million. Of the 1.7 million species identified only about 80,000 species have been placed in the ToL [10]. There are applications for the ToL: to help understand how pathogens become more virulent over time, how new diseases emerge, and to recognise species at risk of extinction [10, 7]. One approach to building the ToL is to combine smaller trees into “supertrees”. Phylogenetic trees have been created for relatively small sets of species [14]. These trees are then combined together into supertrees.

In 2003 Ian Gent, Barbara Smith, Christine Wu Wei, and myself reported the first constraint programming model for supertree construction [3]. This was essentially a proof of concept, showing that constraint programming could address this problem in principle although our implementation was somewhat inefficient. This has recently been re-implemented using a faster constraint programming toolkit (JChoco, a java constraint programming tool [5]) and has allowed us to look at larger problems and get a better idea of the limits of this encoding. Furthermore, with this new implementation we are able to demonstrate the flexibility of our model, something that should be expected when using a versatile technology such as constraint programming.

The remainder of this article is organised as follows. First, I reintroduce the problem of supertree construction and briefly present the constraint encoding of [3]. Next, I present a study that attempts to reproduce the results of building a relatively large supertree of sea birds, reported by Kennedy and Page in [6]. I then describe a richer version of the supertree problem, where ancestral dates are included within species trees [12] and show how the constraint model can be modified to address this. Finally, we look at what limits these models, what we might do to break through those limits, and then draw to a conclusion.

2 Previous Work

The problem is to combine leaf labelled species trees, where there is an intersection in the leaf labels of those trees. The trees must be combined whilst respecting all the arboreal relationships in each tree. An example of this is shown in Figure 1, as a rectangular cladogram displayed using Rod Page’s TREEVIEW [9]. The two input trees are A and B from [6]. One of the first techniques for supertree construction is the OneTree algorithm of Ng and Wormald [8] and is based on the build algorithm of [1]. OneTree

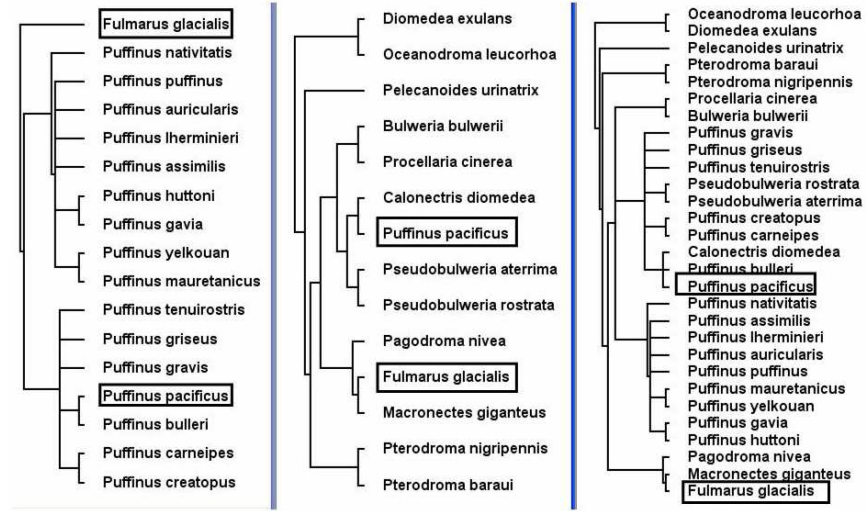


Fig. 1. Two species trees made up of sea birds, and on the right a supertree that combines both. Shared species are highlighted in boxes. The trees correspond to A and B in [6].

is based on the observation that in a tree any three leaf nodes define a unique relation with respect to their most recent common ancestor ($mrca^1$, such that $mrca(a, b)$ is the interior node furthest from the root that has both leaf nodes a and b as descendants. Given three leaf nodes (labelled a , b , and c) one of four relations must hold²:

- (1) $mrca(a, b) > mrca(a, c) = mrca(b, c)$
- (2) $mrca(a, c) > mrca(a, b) = mrca(c, b)$
- (3) $mrca(b, c) > mrca(b, a) = mrca(c, a)$
- (4) $mrca(a, b) = mrca(a, c) = mrca(b, c)$

This is shown pictorially in Figure 2. Using the terminology from [8] we can say that in (1), (2) and (3) we have the triples $(ab)c$, $(ac)b$, and $(bc)a$ ³ and in (4) we have the fan (abc) . Prior to applying the OneTree algorithm two (or more) species trees are broken up into triples and fans via the BreakUp algorithm [8], and the supertree is then constructed (if possible) using this as input.

¹ Note that $mrca$ is sometimes referred to as lca , for least or lowest common ancestor.

² It is assumed labels a , b , and c are all different. $mrca(a, b)$ delivers an actual interior node in the tree and that if $mrca(a, b)$ is equal to x and $mrca(b, c)$ equals y , $x > y$ if node x exists at a greater depth in the tree than y , and $x = y$ if and only if x and y are the very same node.

Note also that if relation (4) is omitted trees are forced to be binary.

³ ... where $(xy)z$ can be read as “ x is closer to y than z ”

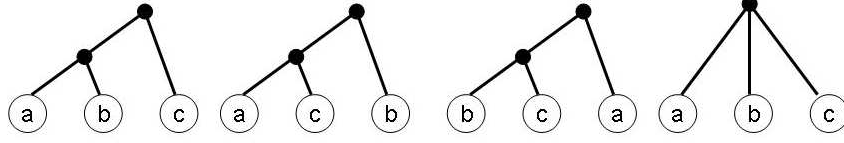


Fig. 2. The four possible relationships between three leaf nodes in a tree: i.e. the three triples $(ab)c$, $(ac)b$, and $(bc)a$, and the fan (abc) .

In [3] we presented the first constraint programming model for this problem. This was based on the rather simple observation that any rooted tree is *ultrametric*⁴. That is, if interior nodes of a tree are labelled with their depth in that tree then any path from the root to a leaf node must be a strictly increasing sequence, and in [4] this is called a min-ultrametric tree. Further, in [4] it is proved that an ultrametric tree has an ultrametric matrix. In an ultrametric matrix M , for any three indices i, j, k where $i \neq j \wedge i \neq k \wedge j \neq k$ one of three relations must hold

$$\begin{aligned} M_{i,j} &> M_{i,k} = M_{j,k} \\ M_{i,k} &> M_{i,j} = M_{j,k} \\ M_{j,k} &> M_{i,j} = M_{i,k} \\ M_{i,j} &= M_{i,k} = M_{j,k} \end{aligned}$$

In an ultrametric tree T , and its corresponding ultrametric matrix M , given two leaf nodes i and j in T , $mrca(i, j)$ will have the same value (and that might possibly be depth in that tree) as $M_{i,j}$. In [4] it is also proved that an ultrametric matrix has a corresponding ultrametric tree, and the proof given is constructive and is therefore an algorithm.

Our constraint encoding starts by producing an $n \times n$ matrix M of constrained integer variables, each with a domain 1 to $n - 1$. Amongst the trees to be combined there are exactly n species and each species is mapped to an integer. The array M is symmetric such that $M_{i,j}$ is the same constrained integer variable as $M_{j,i}$ and all diagonal elements $M_{i,i}$ are preset to zero. An ultrametric constraint is blanketed across the array. By that I mean that for all i, j, k where $1 \leq i < j < k \leq n$ the following

⁴ A metric on a set of objects is given by the assignment of a real number $d(x, y)$ to every pair of objects x and y such that $d(x, y)$ has the following properties:

- $d(x, y) > 0$ for $x \neq y$
- $d(x, y) = 0$ for $x = y$
- $\forall x, y [d(x, y) = d(y, x)]$
- $\forall x, y, z [d(x, y) \leq d(x, z) + d(y, z)]$ (triangular inequality)

To be ultrametric we have the additional property: $\forall x, y, z [d(x, y) \leq \max(d(x, z), d(y, z))]$.

constraint is posted:

$$\begin{aligned}
M_{i,j} &> M_{i,k} = M_{j,k} \vee \\
M_{i,k} &> M_{i,j} = M_{j,k} \vee \\
M_{j,k} &> M_{i,j} = M_{i,k} \vee \\
M_{i,j} &= M_{i,k} = M_{j,k}
\end{aligned}$$

The species trees are then broken up, using the BreakUp algorithm [8], into triples and fans. If in a tree we have the fan (s_0, s_1, \dots, s_m) , i.e. species s_0 to s_m have the same most recent common ancestor, then the set of mC_3 3-fans $\{(s_0, s_1, s_2), (s_0, s_1, s_3), \dots, (s_0, s_1, s_m), \dots, (s_{m-2}, s_{m-1}, s_m)\}$ are produced by our BreakUp algorithm. These triples and 3-fans are then used to break disjunctions in the above constraint. The M variables are then the decision variables, and a solution is found. Assuming a solution exists, the resultant supertree is constructed from the ultrametric matrix using the algorithm in chapter 17 of [4].

3 A Supertree of Sea birds

One obvious limiting factor of our constraint model is its sheer size. It generates $O(n^2)$ constrained integer variables and $O({}^nC_3)$ ultrametric constraints as above. A study was performed to determine just how far this model could be pushed. The model was re-coded in JChoco, a free java constraint solver [5], and can be downloaded from [13]. An attempt was made to reconstruct the supertree produced by Kennedy and Page [6]. The data set is seven species trees of sea birds, identified as A through to G. This is shown in Table 1. A table entry gives the number of species involved in a pair of trees, and along the diagonal the number of species in an individual tree. For example, combining tree A (17 species) with tree B (14 species) results in a supertree with 29 species. Therefore A and B have 2 species in common. A table entry in closed round brackets shows that the two trees are incompatible. In particular, trees A and C are incompatible, B and C are incompatible, C and G are incompatible, as are D and G. An entry of a dash (-) means that the data set was too large to model.

	A	B	C	D	E	F	G
A	17	29	(32)	47	-	31	46
B	29	14	(29)	42	-	30	40
C	(32)	(29)	20	50	-	34	(44)
D	47	42	50	30	-	44	(56)
E	-	-	-	-	90	-	-
F	31	30	34	44	-	16	(38)
G	46	40	(44)	(56)	-	(38)	30

Table 1. Size of species trees and supertrees for the 7 trees in [6]. The diagonal gives the size of individual trees. Off the diagonal is given the size of the supertree, in brackets if incompatible, and a dash if too large to model.

In the Auk paper [6] Kennedy and Page went back to the underlying evidence to successfully combine all of these trees. I cannot do that, and from Table 1 it can be seen

that the best that can be done is to combine trees A, B, D and F⁵. The trees can be constructed in a number of ways, i.e. by adding A to B to get supertree AB, adding D to F to get supertree DF, and then combining supertrees AB and DF. This was in fact done, however there is a risk associated with this. Supertree AB is not unique, and neither is DF. Furthermore AB and DF may be incompatible! This was demonstrated by Sander-son, Purvis, and Henze in [11]. The input program was modified such that it takes as input a file containing names of trees to be combined, i.e. the encoding takes as input a forest. This forest is then broken up into triples and 3-fans before solving. The resultant supertree is shown in Figure 4 at the end of this paper. This took about 20 seconds to produce on a 3 GHz processor. The supertree has 69 species.

4 Ancestral Divergence Dates

In [12] ancestral divergence dates are added to the interior nodes of trees, where dates may be relative or explicit. The RANKEDTREE algorithm (proposed in [2]) takes as input two species trees where interior nodes are assigned integer values such that if the divergence of species A and B predates the divergence of species X and Y then the most recent ancestor of A and B will be assigned a value less than the most recent common ancestor of species X and Y.

This is trivial to incorporate into the constraint model. If we assume that trees have already been ranked, and some or all interior nodes have been labelled, then for each pair of species (X,Y) in the leaf set of a tree we get the value of $mrca(i, j)$ where i and j are the integer indices corresponding to species X and Y respectively. The constraint integer variable $M_{i,j}$ is then set to the value of $mrca(i, j)$ if and only if $mrca(i, j)$ is labelled. The tree is then broken up into its triples and 3-fans and these constraints are then used as disjunction breakers. In [13] this has been implemented as the RBuild (Rank Build) method.

In fact, we can go one step further. We associate a decision variable $D_{i,j,k}$ with each ultrametric constraint and post the following constraints:

$$\begin{aligned} D_{i,j,k} = 1 &\leftrightarrow M_{i,j} > M_{i,k} = M_{j,k} \\ D_{i,j,k} = 2 &\leftrightarrow M_{i,k} > M_{i,j} = M_{j,k} \\ D_{i,j,k} = 3 &\leftrightarrow M_{j,k} > M_{i,j} = M_{i,k} \\ D_{i,j,k} = 4 &\leftrightarrow M_{i,j} = M_{i,k} = M_{j,k} \end{aligned}$$

Therefore, rather than instantiate the variables in M we instantiate the disjunction breaking decision variables D . As a consequence of this, in a solution variables in M may have ranges of values. This is demonstrated in Figure 3. On the left and right we have two ranked species trees of cats taken from [12] (where cat names were given three-letter abbreviations). On the right we have one of the 17 possible resultant supertrees. Note that the most recent common ancestor of PTE and LTI is labelled with

⁵ Table 1 can be considered as an adjacency matrix A of a graph where an entry $A_{i,j}$ not in closed round brackets means that there is an edge (i, j) signifying that tree i is compatible with tree j . In the corresponding graph of A the largest clique has 4 vertices and those vertices are A, B, D and F.

the range [6..9]. In total 7 of the 17 solutions contain interior nodes with ranges. Without this 30 solutions are produced. This addresses one of the issues raised in [12], i.e. to enumerate all supertrees compactly. Our constraint model has been further modi-

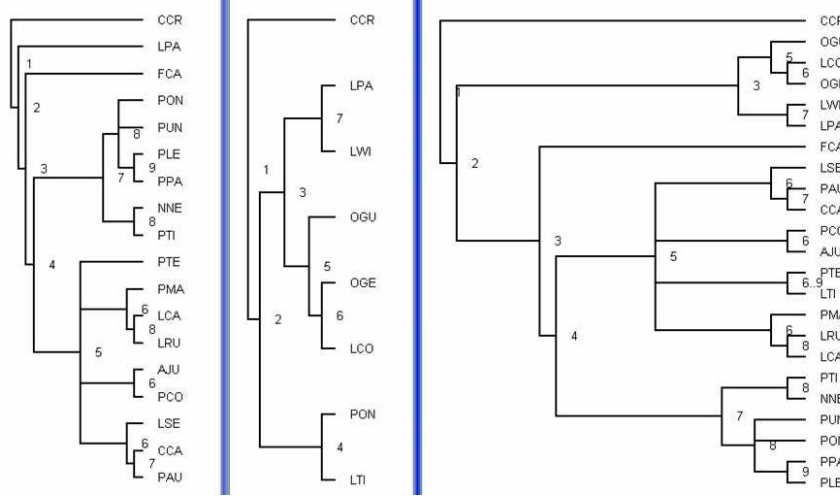


Fig. 3. Two ranked trees of cats taken from [12] and on the right one of the 17 possible supertrees, this one with the most recent common ancestor of PTE and LTI having a range of values.

fied such that a penalty is taken when a decision variable $D_{i,j,k}$ takes a value 4, i.e. when a fan is selected. The penalties are then minimised to produce the supertree that contains the least number of fans. This might not be biologically sound but it has been implemented to demonstrate the versatility of the model. Again, this is available at [13].

5 Limitations, Future Work, Conclusion

Clearly the model is self limiting by its cubic size. There are $O(n^3)$ ternary constraints and the same number of variables when we address the optimisation problem (minimising fans). The largest trees we have built have about 70 species. One obvious next step is to make this model more compact, and this might be done by implementing a specialised ultrametric constraint that involves three variables. This constraint might propagate more efficiently than as at present (using toolkit primitives) and each of the constraints might take less space. However, we still have $O(n^3)$ of these constraints. Therefore the step after that might be to design an n-ary ultrametric constraint that takes as arguments the $n \times n$ array M .

Our model is now available, being re-implemented in java using JChoco [13]. We have been able to demonstrate the versatility of the constraint programming technology, by taking a model that essentially does the same as OneTree, modified it to take a forest as input, dealt with ancestral divergence dates, been able to produce all solutions

compactly, and address an optimisation problem (although this might not be biologically sound). However, the model is limited in what it can do by its sheer size, and this should be addressed soon.

Where to next? In [12] the authors pose the question “what common information is carried in all these supertrees?” I believe that constraint programming will be the technology to address this next challenge.

References

1. 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.
2. O.R.P Bininda-Emonds. *Phylogenetic supertrees: Combining information to reveal the Tree of Life*. Springer, 2004.
3. Ian P. Gent, Patrick Prosser, Barbara M. Smith, and Christine Wu Wei. Supertree Construction with Constraint Programming. In *CP2003 (LNCS 2833)*, 2003.
4. Dan Gusfield. *Algorithms on Strings, Trees, and Sequences*. Cambridge University Press, 1997.
5. JChoco. <http://choco.sourceforge.net/> A java library for constraint satisfaction problems, constraint programming, and explanation-based constraint solving, 2006.
6. M. Kennedy and R.D.M. Page. Seabird supertrees: Combining partial estimates of procellariiform phylogeny. *The Auk, A Quarterly Journal of Ornithology*, 119:88–108, 2002.
7. Gorgina M. Mace, John L. Gittleman, and Andy Purvis. Preserving the Tree of Life. *Science*, 300:1707–1709, 2003.
8. Meei Pyng Ng and Nicholas C. Wormald. Reconstruction of rooted trees from subtrees. *Discrete Applied Mathematics*, 69:19–31, 1996.
9. R.D.M. Page. TREEVIEW: An application to display phylogenetic trees on personal computers. *Computer Applications in the Biosciences*, 12:357–358, 1996.
10. Elizabeth Pennisi. Modernizing the Tree of Life. *Science*, 300:1692–1697, 2003.
11. M.J Sanderson, A. Purvis, and C. Henze. Phylogenetic supertrees: assembling the tree of life. *TREE*, 13:105–109, 1998.
12. C. Semple, P. Daniel, W. Hordijk, R.D.M. Page, and M. Steel. Supertree algorithms for ancestral divergence dates and nested taxa. *Bioinformatics*, 20(15):2355–2360, 2004.
13. CP Supertrees. <http://www.dcs.gla.ac.uk/~pat/supertrees> Patrick Prosser’s JChoco Supertree Builders.
14. TreeBASE. <http://www.treebase.org/> TreeBASE: a database of phylogenetic knowledge, 2003.

Acknowledgements: Thanks to Pierre Flener for his help and encouragement.

