Program using rjMCMC for exploring allopolyploid network priors

Graham Jones

2012-05-19

1 Introduction

This is an update of 2011-11-23-network-prior-program.pdf.

Suppose there are d diploids and m tetraploids. In the language of [2], there are m models $h = 1, h = 2, \ldots h = m$, where h is the number of hybridizations. Each model has a different number of parameters. The main difficulty with the network prior is that the normalization constants are needed for the different models. Let W be the network topology and node times. The conditional priors $\pi(W|h)$ must be comparable for different values of h. To do this analytically, it is necessary to integrate out the parameters in W, ie calculate for each h the value of

$$\int \pi(W|h)\mathrm{d}W$$

which is a sum over labelled topologies and an integral over node times. That seems to be hard. It is possible to deal with Yule or birth-death priors for trees of different sizes, but I can't find anything analogous for networks.

An alternative is to use some fairly arbitrary prior, and estimate its properties by sampling from it. This note describes a program to do that.

2 Prior density for times

The network topology and node times is denoted by W. There are d diploids, m tetraploids, and $h \in \{1, 2, \ldots, m\}$ hybridizations. The network is divided into h tetraploid trees and a single 'diploid history', which is also a tree. Figure 1 shows an example with $d = 3$, $m = 6$, $h = 3$. Note that the diploid history has $d + 2h = 9$ tips. There are d ordinary diploid tips at time 0, and h pairs of tips at nonzero times (b_1, b_2, b_3 in Figure 1). I will call the latter 'hybridization tips'.

Suppose the ith tetraploid tree has m_i tips ($1 \le i \le h$). Then $m = \sum_{i=1}^{h} m_i$ and there are $\sum_{i=1}^{h}(m_i - 1) = m - h$ internal nodes in the tetraploid trees (r_1, r_2, r_3 in Figure 1). There are h hybridization times, and the diploid history has $d + 2h - 1$ internal nodes ($s_1, \ldots, s_8$ in Figure 1). The total number of parameters (node times and hybridization times) which are operated on by the reversible jumps is thus $d + m + 2h - 1$. If the ratios between different models (different h) is to be the same regardless of the scaling factor λ, then the density must reflect this. (The formula used on 2011-11-25 was wrong, or at least incompatible with the MCMC moves I implemented.)

The formula I now use is

$$f(h; d, m)\lambda^{d+m+2h-1} \exp[-\lambda(t_1 + t_2 + \cdots + t_{d+m+2h-1})] \tag{1}$$

where the t_i are all the node times and hybridization times. The factor $f(h; d, m)$ is subject to experiment and user choice. I have separated the parameters d and m because they are fixed for any particular analysis, so dependence on them only introduces a constant factor. But it seems likely that a 'good'

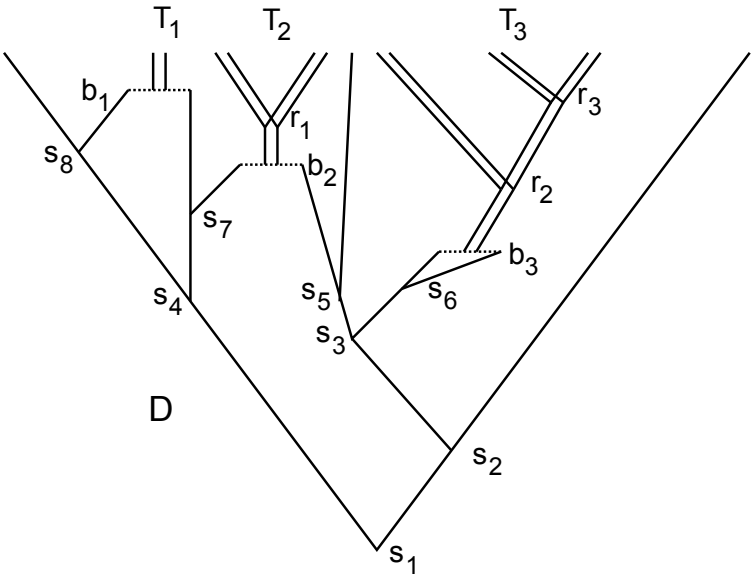

Figure 1: Network notation

formula would involve them. The rest of the formula might be described as 'Yule-like' but does not represent any evolutionary process.

3 Moves

I have implemented four types of move:

- 1. Change a tetraploid subtree, tipwards of the hybridization
- 2. Change an hybridization time
- 3. Change the diploid history, rootwards of hybridizations
- 4. Change the number of tetraploid subtrees

3.1 MCMC moves in tetraploid subtrees

Many MCMC moves for trees have been developed. *BEAST uses a MCMC move for the species tree based on the ideas of [1], and this move also also seems appropriate here. See Figure 2. The steps are:

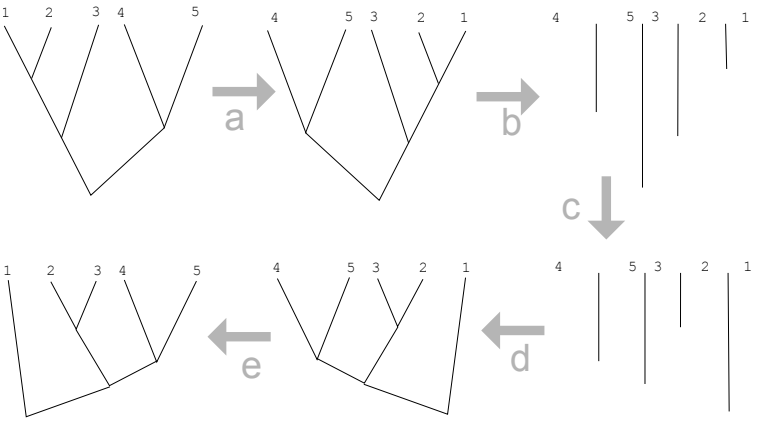

Figure 2: MNL move: a MCMC move based on Mau et al 1999.

a. Randomly orient the tree, giving each branch at each internal node a left/right label.

b. Convert to a point process representation (a 'drawing'), with every node getting an (x,y) position.

c. Change one or more of the node heights.

d. Then, forgetting the identity of the internal nodes, but keeping the left-right order of the tips, construct a new tree by joining up clades in order of node height.

e. (Not necessary, but it makes the move more symmetric.) Forget the left-right ordering.

This will change the topology if two nodes which were originally parent and child get heights which reverses the order. Step (c) should be reversible, so that the probability of going from one set of heights A to another set of heights B is the same as going from B back to A. I call this type of move an 'MNL move'.

3.2 MCMC move for the hybridization time

This is straightforward. In Figure 1, it means moving b_1 in $[0, s_1]$, b_2 in $[r_2, s_2]$, or b_3 in $[r_3, s_3]$. There is no change of topology.

3.3 MCMC move in the diploid history

The MNL move described above can be adapted to deal with the diploid history. The diploid history is an ordinary tree with some tips having nonzero times. So in step (c) the heights must be changed in such a way that no node gets a height smaller than the tip height to its immediate left or right.

I also keep the root as a diploid. If node to change height is the root, and the second highest node is to left or right of all diploids, then the root must stay the root: there is a lower limit which is the height of second highest. If node to slide is not the root, and is to left or right of all diploids, then it must not become the root: there is an upper limit which is the root height.

In the implementation in BEAST, there will also be upper limits from the gene trees.

To keep the height changes simple, I change only one height at a time, and use a uniform kernel.

3.4 MCMC move for the number of hybridizations

NOTE: this describes what the program does (on 2012-05-19), but it is not a rigorous proof that this is what it should do. I think it is correct, and in any case its behaviour must be examined from simulations.

Sampling all values of h can be done by repeatedly changing h to $h-1$ or $h+1$, and that can be done by splitting one tetraploid subtree into two and merging two into one. The difficult part is making the moves reversible, so that the probability of a move going from one network state A to another B is balanced by a reverse move.

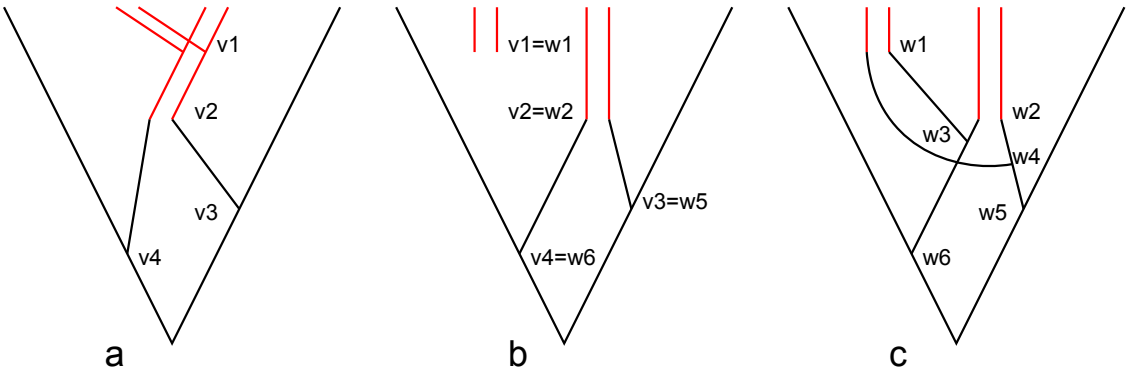

Figure 3: MCMC move to change number of tetraploid subtrees.

When the MCMC move for changing h is chosen, a split or a merge is chosen with equal probabilty. If a split is chosen, but no splits are possible, no move is made; the same network state is sampled again. Likewise, if a merge is chosen, but no merges are possible, the same network state is sampled again.

Splitting (going left to right in Fig 3). Any tetraploid subtree with more than one tip can be split. One, T, is chosen at random. The two child nodes of the root of the tetraploid subtree become the roots of the two new tetraploid subtrees T_1, T_2. In Fig 3, the root of T is at v_1, and the two new roots are also tips. The hybridization time v_2 becomes the hybridization time of one of the new tetraploid subtrees (w_2 in T_2), and the root time v_1 becomes the hybridization time of the other, T_1. This produces the situation labelled (b).

Then two new tips with time w_1 are added to the diploid history. They are joined into the two external branches of the diploid history which led to the tips at w_2. One new node is created in each of these, at w_3 and w_4. The new nodes could be chosen anywhere in the diploid history earlier than w_1. The restricted choice is to make things simpler.

Merging (going right to left in Fig 3) must be the reverse of splitting. So two tetraploid subtrees can only merge if they have a configuration like that labelled (c) in the figure. It is required that the hybridization time of one tree is earlier than the root time of the other tree, and later than the hybridization time of the other ($0 \leq w_1 \leq w_2$ in the figure). It is also required that there are two nodes in the diploid history (the ones at w_3 and w_4 in the figure) which each have one child as a hybridization tip for T_1, and the other child as a hybridization tip for T_2. It is not necessary that the ancestors of these two nodes at w_3 and w_4 be different: the nodes at w_5 and w_6 could be identical.

A list of possible pairs of tetraploid subtrees is made, and if there any suitable pairs, one pair is chosen at random, and the merge is carried out. The most recent hybridization time (w_1) becomes the root time of the merged tetraploid subtree T. The tree (T_2) with the earlier hybridization time provides the hybridization tips for T, and the hybridization tips of T_1 are removed.

There are various Hastings ratios to be worked out. I will describe what I have implemented on 2012-05-19. I am using [2] as my main reference. In particular, equations (7) and (8) are applicable.

When new node times (w_3 and w_4) are created, this is implemented by sampling two values u_1, u_2 from the uniform distribution on $[0, 1]$ and then using $w_3 = w_2 + u_1(w_6 - w_2)$ and $w_4 = w_2 + u_2(w_5 - w_2)$. All other node heights are set equal to existing values (they just acquire new interpretations). This means the determinant of the Jacobian when splitting is

$$\left| \frac{\partial(w)}{\partial(v, u)} \right| = (w_6 - w_2)(w_5 - w_2) \tag{2}$$

Here $w = \theta^{(S)}$ is a vector of node and hybridization times for the network with the split case, $v = \theta^{(M)}$ is a vector of node and hybridization times for the network with the merged case, and $u = (u_1, u_2)$. The reciprocal of equation (2) is used when merging. Since the density of u_1 and u_2 are both 1, the 'q' terms in equations (7) and (8) of [2] can be omitted.

This leaves the probabilities of the moves being chosen, namely $j(\mathrm{M}, \theta^{(M)})$ and $j(\mathrm{S}, \theta^{(S)})$ in Green's notation, to be accounted for, where M and S stand for 'merged' and 'split', which are Green's cases 1 and 2.

Let the number of splitting moves from a particular network state $\theta^{(M)}$ in model M be N_s. N_s is twice the number of tetraploid subtrees with more than one tip, since either child of the root can play the role of T_1, the tree that gets new hybridization tips. The probability that one splitting move is chosen is $1/N_s$, since the move is chosen uniformly from the possibilities. Thus $j(\mathrm{M}, \theta^{(M)}) = 1/N_s$. For merging, the number of ordered pairs N_m of tetraploid subtrees which are suitable for merging is found. The condition on the hybridization times means only one ordering of each pair is possible, and there are no further choices in how to merge two trees, so there are N_m merges to choose from. Thus $j(\mathrm{S}, \theta^{(S)}) = 1/N_m$. This results in a ratio $(1/N_m)/(1/N_s) = N_s/N_m$ when splitting, that is, moving from model M to model S, and N_m/N_s when merging. The merging case in pseudo-code:

```
doMergeMove() {
HastingsRatio = 1;
X = findCandidateMerges()
```

```
if (|X| > 0) {
  choose a random merge x from X
  HastingsRatio *= |X|
  HastingsRatio *= doMerge(x)
  Y = findCandidateSplits()
  HastingsRatio /= |Y|
  }
return HastingsRatio;
}
```

Here `doMerge(x)` carries out the move from S to M, and returns the Jacobian.

4 Results

Some preliminary results. 10 million generations sampled every 100, after 100,000 states of burnin.

$d = 2$, $m = 10$, $f(h; d, m) = 1$, $\lambda = 1.0$. With $f(h; d, m) = 1$ the results give large h a high probability, and $h < 5$ very low probability, which is not realistic. The opposite would be usually be expected, since speciations of tetraploid species are more common than hybridizations for most groups of species.

h	count
1	0
2	0
3	3
4	56
5	144
6	1011
7	3361
8	12319
9	33325
10	49781

$d = 2$, $m = 10$, $f(h; d, m) = 4^{-h}$. Fairly close to uniform. Still not suitable for most analyses, but a better starting point.

h	count
1	8932
2	9262
3	10040
4	10301
5	11274
6	12289
7	13583
8	12438
9	8567
10	3314

References

[1] Bob Mau, Michael A. Newton, Bret Larget, *Bayesian Phylogenetic Inference via Markov Chain Monte Carlo Methods*, Biometrics, Vol. 55, No. 1 (Mar., 1999), pp. 1–12

[2] Peter J Green, *Reversible jump Markov chain Monte Carlo computation and Bayesian model determination*, Biometrika (1995), Vol. 82, 4, No. 1 pp. 711–32