Prior, initial state, and MCMC moves for allopolyploid networks

Graham Jones

2012-09-24

[Incorporates most of 2012-05-19-network-prior-program.]

1 Introduction

This describes the prior distribution used for the network and the MCMC moves used to sample the posterior when using the AlloppNET model. The most complex move is the one which changes the numbers of hybridizations, and this move is related to the choice of prior for reasons described next.

Suppose there are d diploids and m tetraploids. In the language of [2], there are m models $h = 1, h = 2, \dots h = m$, where h is the number of hybridizations. Each model has a different number of parameters. Let W be the network topology and node times. The conditional priors $\pi(W|h)$ must be comparable for different values of h in order to calculate the correct acceptance ratios when jumping between models. Ideally we would like to integrate out the parameters in W analytically, 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. Then we could define an actual density on the space of networks. It would be enough if we could calculate the ratios between such expressions apart from a single unknown constant. However this appears to be hard for any reasonable formula for the prior, so we resort to estimation. If there is considerable information to guide the choice of formula for an unnormalised density, one could estimate the normalization constants using the MCMC algorithm with no data. More commonly, there is little known about what such a formula should look like, so the process of choosing a prior becomes a matter of 'tweaking' a formula until the samples produced by the MCMC process with no data appear reasonable. For example it is possible to adjust things so that each of the models $h = 1, h = 2, \dots h = m$ has approximately equal prior probability.

The situation is similar to that of BPP (Yang and Rannala) where the species tree has different numbers of parameters as the species assignments change. However in that case the prior on the species tree is the distribution for constant rate birth-death process, and this has a known density. In other words the normalization constants are all known to be one.

2 Prior density for times

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 $d + 2h$ 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 ($1 \leq i \leq h$). 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 ($s_1, \dots, s_8$ 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.

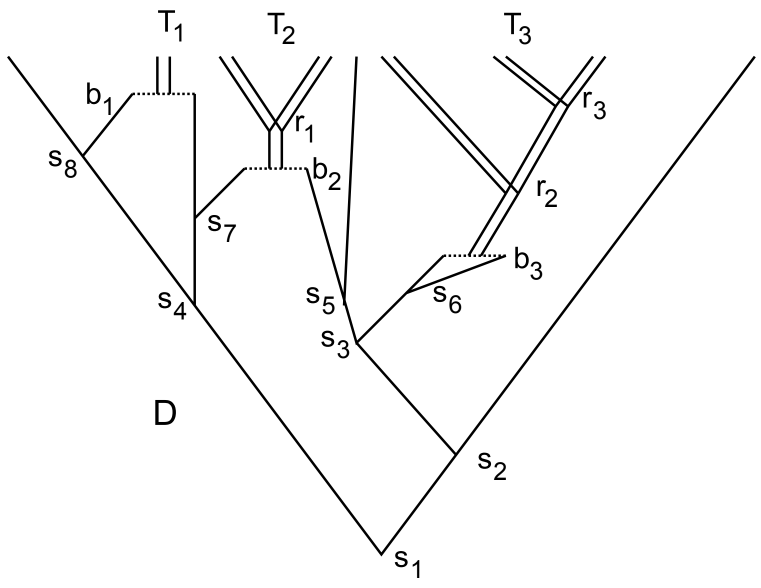

Figure 1: Network notation

The formula I 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' formula would involve them. The rest of the formula might be described as 'Yule-like' but does not represent any evolutionary process. In particular, the probabilities of different topologies is not explicitly specified here. Instead these probabilities must be estimated by sampling from the prior.

The formula above make no mention of population parameters, which also belong to the network, and change in number as the number of hybridizations change. However the population parameters are assumed to be independent of the network topology and node times in the prior, so can be dealt with separately: see subsection 4.5.

3 Initial state

A random initial state for the species network is chosen as follows.

1. The tetraploid species are partitioned into one or more groups using the Chinese restuarant process.

2. Trees from a Yule process are generated for each of the groups of tetraploid species

3. The diploid history is constructed in a manner similar to the Yule process working backwards in time, with some modifications. Each diploid species is a tip and there are two hybridization tips for each tetraploid subtree. Two subtrees are repeatedly chosen from those available and joined into a subtree. When two nodes are selected for joining, the choice is constrained so that the diploids do not all merge while there are still tetraploids left to merge. Also, the height of the root of the new subtree has to be made earlier than either of the nodes chosen for joining (which would not happen automatically, since the hybridization tips have nonzero height).

4 Moves

I have designed five new types of move which are particular to allopolyploid networks:

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
5. Change the assignment of sequences within polyploid individuals

In the implementation, 1,2, and 3 are combined as `AlloppNetworkNodeSlide`. Number 4 is `AlloppChangeNumHybridizations`, and 5 is `AlloppSequenceReassignment`. There is also `AlloppHybPopSizesScale` for population parameters which is similar to the scale operator in BEAST, but written to cope with a changing number of parameters.

4.1 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. Within a single tetraploid subtree the situation is very similar to that in *BEAST. See Figure 2. The steps are:

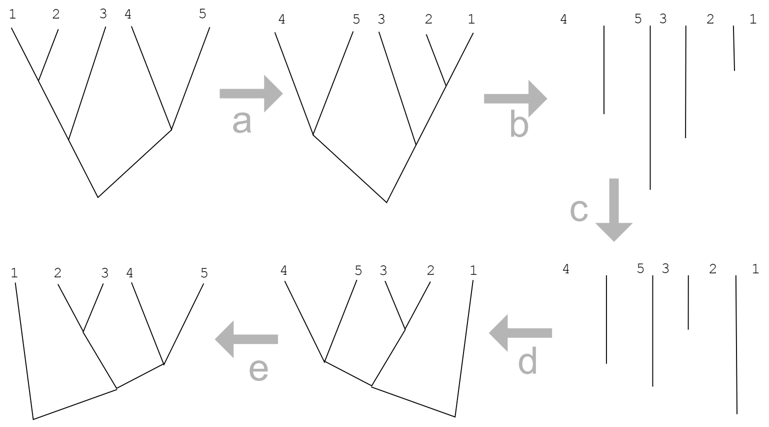

Figure 2: MNL move: a MCMC move based on Mau et al 1999.

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. Choose a node at random and change its height.

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'.

One reason for preferring this move over other types of moves is that the moves can be constrained to keep the species tree compatible with the gene trees. In step (c), the most recent coalescence among genes which would conflict with the chosen node is determined, and the new node height is constrained to be no more ancient than this.

Algorithm: For each gene, find the set of sequences S_{left} which are currently attached to a tip to the left of the chosen node in the oriented species tree, and likewise find S_{right}. This must take into account the assignment of sequences within individuals. The most recent coalescence between a sequence in S_{left} and S_{right} for any gene determines the maximum height for the node.

4.2 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.

4.3 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 and the constraint that the root is a diploid. There are three types of constraint on the new height.

There are constraints from the gene trees as in the tetraploid subtrees. In order to calculate the sets S_{left} and S_{right} for a particular node it is necessary to visit the tetraploid subtrees which are attached to the hybridization tips of the diploid history.

Secondly, there are lower bounds on the new height due to the fact that the hybridization tips have nonzero height. In the oriented tree, this amounts to ensuring that the new height does not become smaller than either of the heights of adjacent nodes. (Nodes adjacent to internal nodes are always tips in the left-right ordering.)

Thirdly there are constraints to 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 node. 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.

4.4 MCMC move for the number of hybridizations

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.

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.

4.4.1 Splitting

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 T become the roots of the two new tetraploid subtrees. The child nodes are not treated symmetrically in the move, so both orderings of the child nodes is treated as candidates. There are thus twice as many candidate splits as tetraploid subtrees.

The steps shown in Fig 3 are:

1. split T into $T1$ and $T2$ and create a new hybridization height for $T1$ between the root height of $T1$ and the root height of T.
2. create a new hybridization height for $T2$ between the root height of $T2$ and the root height of T. Create two ancestor nodes for the hybridization tips, one re-using the root height of T and the other between this time and the minimum of the limits imposed by the gene trees and the height of the node that will become its ancestor node.
3. Join up the topology in the diploid history. Note that they could join the diploid history in many ways but a particular way is always chosen, so that the two new subtrees 'share legs' as shown. Note also that it is necessary to keep track of which of the two 'copies' of a tetraploid subtree is which (ie, the left leg of one must correspond to the left leg of the other).

The most difficult parameter is the new node height when splitting, which is below the root height of the subtree. This can conflict with gene trees, so the sets of (species,sequence) pairs have to be found for the two child nodes of the new node, and the gene trees examined to find the most recent coalescence that conflicts with the new node.

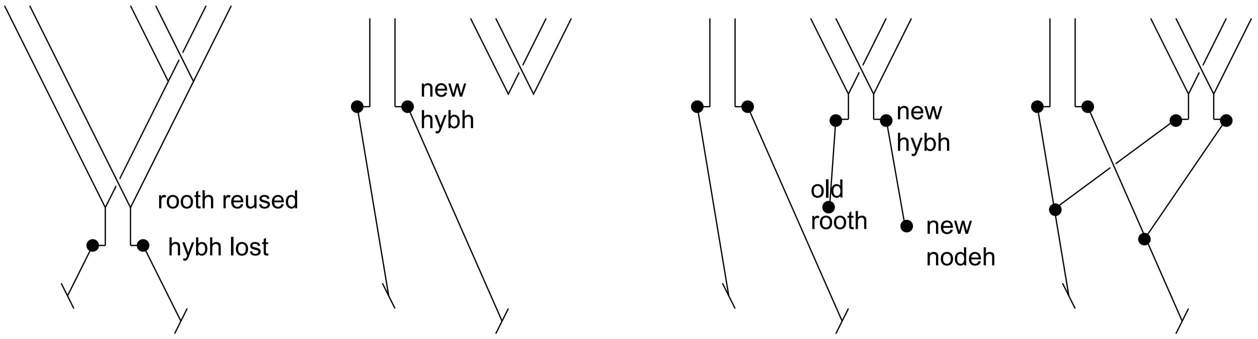

Figure 3: MCMC move to increase number of tetraploid subtrees. ‘nodeh’ is short for ‘node height’, ‘rooth’ for ‘root height’, ‘hybh’ for ‘hybridization height’.

4.4.2 Merging

Merging (going right to left in Fig 4) must be the reverse of splitting. So two tetraploid subtrees can only merge if they have a configuration like that in the figure, ie ‘sharing legs’. It is not necessary that the two nodes at the bottom of the figure be different.

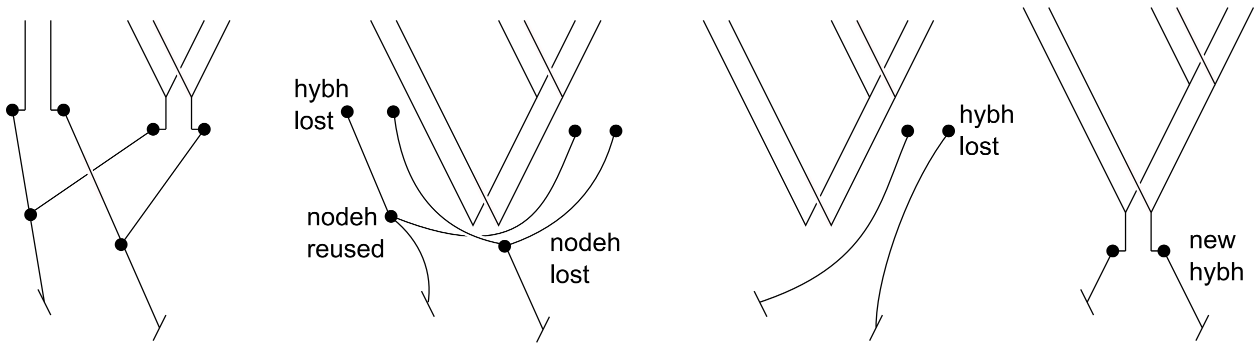

Figure 4: MCMC move to decrease number of tetraploid subtrees. ‘nodeh’ is short for ‘node height’, ‘rooth’ for ‘root height’, ‘hybh’ for ‘hybridization height’.

A list of possible pairs of tetraploid subtrees is made, and if there any suitable pairs, one pair $T1, T2$, is chosen at random, and the merge is carried out. The steps shown in Fig 4 are:

1. Merge $T1$ and $T2$ into T. The root height of T becomes the height of the most recent ancestor to a hybridization tip.
2. Remove the hybridization tips for $T1$ from the diploid history. This loses a hybridization height and a node height. The latter requires finding the limit from gene trees for Hastings ratio.
3. Give the hybridization tips for $T1$ new heights below the root height of T, and join up.

Note that the limit from gene trees must be calculated for the lost node height (as when splitting), in order to calculate the Hastings ratio.

4.4.3 Hastings ratios

There are various Hastings ratios to be worked out. I will describe what I have implemented on 2012-09-12. I am using [2] as my main reference. In particular, equation (7) is applicable. This provides the acceptance ratio for a splitting move

$$\min\left\{1, \frac{p(S,\theta^{(S)}|y)j(S,\theta^{(S)})q_S(u^{(S)})}{p(M,\theta^{(M)}|y)j(M,\theta^{(M)})q_M(u^{(M)})}\left|\frac{\partial(\theta^{(S)},u^{(S)})}{\partial(\theta^{(M)},u^{(M)})}\right|\right\} \tag{2}$$

where S (for 'split') and M (for 'merged') replace Green's 1 and 2. Here y is the data, and $p(S,\theta^{(S)}|y)$ and $p(M,\theta^{(M)}|y)$ the usual Bayesian posteriors. The term $j(S,\theta^{(S)})$ gives the probability of choosing the splitting move and $j(M,\theta^{(M)})$ of choosing the reverse merging move. The term $\theta^{(S)}$ is a vector of node and hybridization times for the network with the split case, and $\theta^{(M)}$ is a vector of node and hybridization times for the network with the merged case. The vectors $u^{(S)}$ of length 3 and $u^{(M)}$ of length 1 provide the extra parameters created when doing the jump. The function q_S is the density of the distribution from which $u^{(S)}$ is sampled; likewise q_M.

In my case $u^{(S)}$ and $u^{(M)}$ can be generated by independent sampling from the uniform distribution on $[0,1]$ for each dimension. In this case $q_S(u^{(S)})$ and $q_M(u^{(M)})$ are both 1 and can be omitted from the formula. The new parameters are then derived from these values, as functions of the other old parameters. In my case, all these function are linear functions mapping $[0,1]$ to a suitable range, the range being some function of the other parameters $\theta^{(S)}$ or $\theta^{(M)}$. This means that the contribution to the Hastings ratio consists of the lengths of these ranges (or the reciprocals of these lengths).

This leaves the probabilities of the moves being chosen, namely $j(\mathrm{M},\theta^{(M)})$ and $j(\mathrm{S},\theta^{(S)})$. Let the number of splitting moves from a particular network state $\theta^{(M)}$ in model M be N_s. The probability that one splitting move is chosen is $1/N_s$, since the move is chosen uniformly from the possibilities. Thus $j(M,\theta^{(M)}) = 1/N_s$. For merging, the number of ordered pairs N_m of tetraploid subtrees which are suitable for merging is found. 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.

4.4.4 Problem with old move

In 2012-05-19-network-prior-program.pdf I have a move which seemed OK, but I now think that it is only OK in the absence of constraints from gene trees. When splitting it can make a network incompatible with the gene trees, when merging not. The situation is complicated by the fact that Hastings ratios were calculated for the unconstrained interval. So splitting has a too large interval, so a correspondingly low density, but fails to move at all if the new time is incompatible with the gene trees. These two things cancel and the result is like an inefficient version of one that chose compatible times in a smaller interval. However, when merging, the big interval is used for the Hastings ratio, and nothing cancels that.

The old splitting move invented two times which could cause incompatibility, and fixing it looked very complicated, so I use a new move which only invents one. It does at least appear to mix better.

4.5 Population parameters

When the number of hybridizations changes, the number of population parameters also changes. It increases by one for each hybridization. So in a split a new population parameter is added, and in a merge one is removed. When one is added, it is sampled from the prior for the population. The Hastings ratio is calculated from the value of the density of the population prior at the new parameter value when splitting, or at the lost value when merging.

4.6 Change the assignment of sequences within polyploid individuals

This is quite straightforward. A uniform prior on the possible assignments is used, so the Hastings ratios are all 1. The reassignment moves need to visit all possible 'flips' for each gene in each tetraploid individual. This is easy to arrange in a mathematical sense. The only difficulty is choosing combinations of flips which have good mixing properties. As usual with MCMC algorithms, this requires experimentation.

As of 2012-10-08, I have three types of move. The first flips the assignment of a single gene in a single individual. The second works within a single gene tree, and chooses a random node within it, and roughly speaking, flips a clade of individuals. The details are in the code.

The third type of move was introduced to avoid the MCMC chain getting stuck in a particular situation (see below). The merging move which reduces the number of hybridizations by one, requires the left legs to be 'shared', and the right legs to be shared; left leg shared with right leg will not do. However the network can get into a situation where the legs are 'reversed' *and* the sequence assignments of the genes at the tips of the two candidate are opposite to one another. This makes a state which may be very difficult to leave. So I have added a third move which flips sequence assignments of all genes of all individuals of all species in a tetraploid subtree, *and* switches the legs around, swapping left and right. This takes the network to an equivalent state with the same likelihood, but one from which the merging move can operate. This may well not be the best way to tackle the problem, but it seems to work so far.

```
Diploid history      height ttree leg
+                     0.00811
|-+                   0.00266
| |-                 4.63e-04   1    R
| --+                6.64e-04
|   |-               4.38e-04   0    L
|   --b                  zero
--+                   0.00354
  |-+                7.29e-04
  | |-               4.38e-04   0    R
  | --               4.63e-04   1    L
  --a                    zero

MUL-tree                height
+                     0.00811
|-+                   0.00266
| |-z1                   zero
| --+                6.64e-04
|   |-y0                 zero
|   --b                  zero
--+                   0.00354
  |-+                7.29e-04
  | |-y1                 zero
  | --z0                 zero
  --a                    zero

Gene tree 0
+                     0.00849
|-+                   0.00506
| |-+                 0.00111
| | |-01zA               zero
| | --01yA               zero
| --01aA                 zero
--+                   0.00575
  |-01zB                 zero
  --+                 0.00254
    |-01yB               zero
    --01bA               zero
Sequence assignments
01zA:0  01zB:1
01yA:1  01yB:0
```

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