Original $O(n^4)$ algorithm

Definitions and general plan

Let n_H and n_P be the number of tips of the host tree and the parasite tree, respectively.
Let $n = n_H + n_P$.

Let the (almost binary) host tree be $G_H = (V_H, E_H)$ and the (almost binary) parasite tree be $G_P = (V_P, E_P)$. For convenience, we will also have a dummy root for the parasite tree as well. These graphs should be rooted and directed. Please assume that as we write v_H, e_H, v_P or e_P probably with subscript in this document, then we refers to $v_H \in V_H$, $e_H \in E_H$, $v_P \in V_P$ and $e_P \in E_P$, respectively. Also, in figures, we will use black for those related to the host, and grey for those related to the parasite.

Let $\text{cost}_{\text{HS}}: E_H \times E_H \to \mathbb{N}$ be a function that compute the cost due to host switching. Furthermore, it needs to have one of the two following running time: $O(1)$ for each computation, or $O(n^4)$ for pre-computation. Let other costs be constants, where $\text{cost}_{\text{COSPEC}}$, cost_{DUP} and $\text{cost}_{\text{LOSS}}$ are the costs for cospeciation, duplication, and loss, respectively. We will also assume that these costs are also integers, but this assumption is not important.

Let T be the set of all possible times, which is probably $\{0, 1, \dots, |n_H|\}$. We require that all elements in T to be integers, the maximum element to be within $O(n)$, and the minimum to be non-negative.

Let $A: E_P \times E_H \times T \to \mathbb{N}$ be the function where $A(e_P, e_H, t)$ calculates the following value:

> the **minimum cost** of placing the **parasite edge** e_P on the **host edge** e_H **right after** time t, including the cost of all its **subtree**. By "**right after**", we mean that no other event occur after time T.

Let $B: E_P \times E_H \times T \to \mathbb{N}$ be the same cost as A but for **right before** time t.

Our plan is to start from the tips of both trees by finding initial values of $A(e_P, e_H, |n_H|)$. Then we will compute $B(e_P, e_H, t-1)$ from $A(e_P, e_H, t)$, and $A(e_P, e_H, t-1)$ from $B(e_P, e_H, t)$, alternatively in the order of decreasing t until we use all edges in the parasite tree, which will give us the cost of the solution. There will be $O(n)$ iterations of computation to be done since $|T| \in O(n)$. We can extend A and B to store not only the cost but also the association (solution). Note that "association" here is a set of placement of vertices in v_P on edges or vertices on v_H.

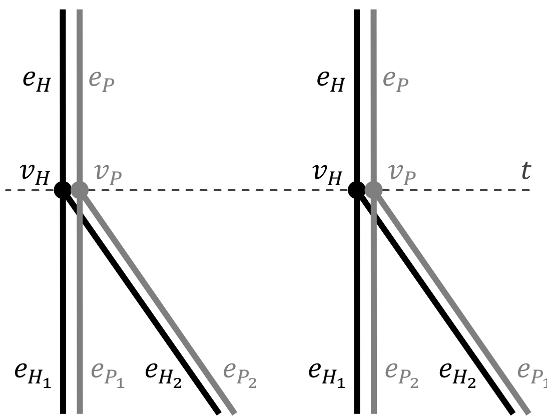
The candidate for final solution (full tree of parasite on the tree of host) will be at any A or B with the edge connected to the dummy root of parasite tree. This suggests that we have already place the entire parasite tree on the host tree already.

For base case, we will start with $B(e_P, e_H, |n_H|)$. Let v_P and v_H be the vertices at the end of e_P and e_H with higher time, respectively. To calculate $B(e_P, e_H, |n_H|)$, we already know the associations of all the tips. Furthermore, we do not include any events at all. Therefore, only possible placement of edges is for those with a tip on one end. That is,

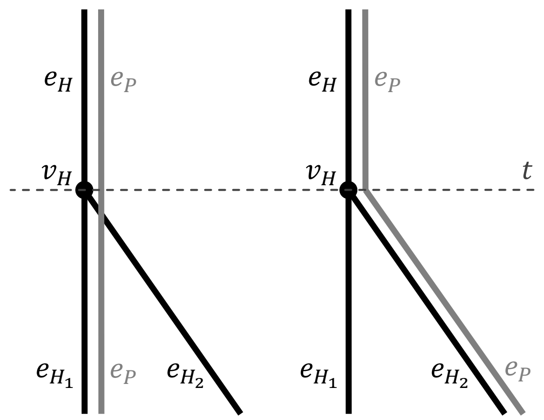
$$B(e_P, e_H, |n_H|) = \begin{cases} 0 & \text{if } v_P \text{ and } v_H \text{ are associated tips} \\ \infty & \text{otherwise} \end{cases}$$

Calculation of $B(e_P, e_H, t)$ from $A(e_P, e_H, t)$

From our definition, this will be a calculation of events happen exactly at time t. Assuming that no events can occur at the same time, the only allowed event at time t will involves exactly one speciation of the host tree. Therefore, there are two possible event types: a cospeciation or a loss. Let the node that speciates in host tree be v_H. Let the edge that connects this vertex to its root be e_H, and the edges that connect to its children are e_{H_1} and e_{H_2}. For other edges in E_H that are not e_H, e_{H_1} and e_{H_2}, there are no events associated with them and therefore value for B will be the same as that in A. We will need to set $B(e_P, e_{H_1}, t)$ and $B(e_P, e_{H_2}, t)$ to ∞ since e_{H_1} and e_{H_2} no longer exists in the time we consider. We will now consider placing e_P on e_H. Let v_P be the vertex at an end of e_P with higher time.



Cospeciation



Loss

In case of cospeciation, v_P must also speciate – let the edges that connect v_P to its children be e_{P_1} and e_{P_2}. Assuming that we do not allow cospeciation with host switching, then e_{P_1} and e_{P_2} must lie on e_{H_1} and e_{H_2} in some sequence. Then, the cost of each case is:

- $A(e_{P_1}, e_{H_1}, t) + A(e_{P_2}, e_{H_2}, t) + \text{cost}_{\text{COSPEC}}$
  The solution will include association of $A(e_{P_1}, e_{H_1}, t)$, $A(e_{P_2}, e_{H_2}, t)$, and v_p on v_H.
- $A(e_{P_1}, e_{H_2}, t) + A(e_{P_2}, e_{H_1}, t) + \text{cost}_{\text{COSPEC}}$
  The solution will include association of $A(e_{P_1}, e_{H_2}, t)$, $A(e_{P_2}, e_{H_1}, t)$, and v_p on v_H.

In case of loss, e_P must come from one of the edge, either e_{H_1} or e_{H_2}. Therefore, in case e_P is not the dummy edge (we do not allow loss on dummy parasite edge), the cost is:

- $A(e_P, e_{H_1}, t) + \text{cost}_{\text{LOSS}}$
  The solution will be the same association as $A(e_P, e_{H_1}, t)$.
- $A(e_P, e_{H_2}, t) + \text{cost}_{\text{LOSS}}$
  The solution will be the same association as $A(e_P, e_{H_2}, t)$.

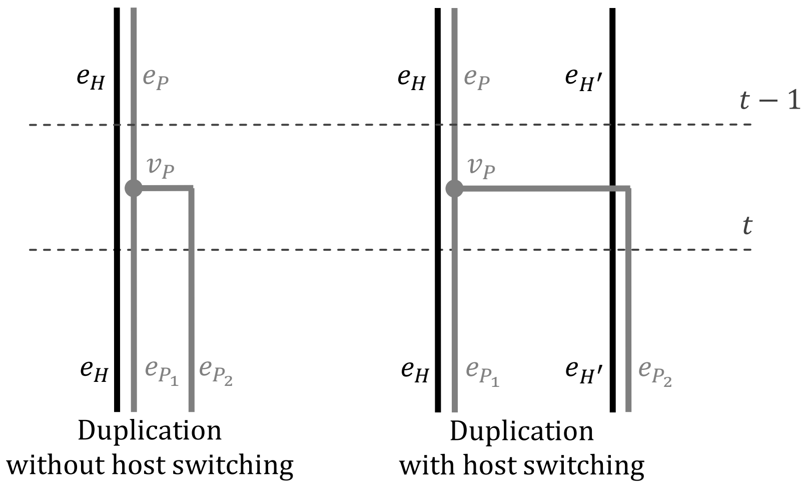
The optimal solution can be obtained simply from the minimum of all 4 cases. The computation in this part takes $O(1)$ for each e_P and e_H.

Calculation of $A(e_P, e_H, t-1)$ from $B(e_P, e_H, t)$

During the time after $t-1$ until before t, there are no changes in the host tree. Therefore, the events that can occur during that time are only duplications and therefore also host switching.

However, many events may occur. As a result, not only $B(e_P, e_H, t)$ but also the minimum cost of placing e_P on e_H in some intermediate time between $t-1$ and t is needed. Furthermore, we only need information children vertices of v_P to obtain the minimum cost for v_P. In other words, we can fill out the table $A(e_P, e_H, t-1)$ in **post-order** of **edges** e_P, and arbitrary order of e_H that passes through time $t-1$ to t (active edges). Therefore,

- If v_P has no children, then no events can occur below v_P. So, $A(e_P, e_H, t-1) = B(e_P, e_H, t)$ for all e_H.
- Since having no events can be a candidate, then $B(e_P, e_H, t)$ is also a candidate for $A(e_P, e_H, t-1)$ even if it has children.
- If v_P has children and the edges that connect v_P to its children are e_{P_1} and e_{P_2}, and duplication occurs at v_P, then we need $A(e_{P_1}, e_H, t-1)$, $A(e_{P_2}, e_H, t-1)$ for all e_H to calculate $A(e_P, e_H, t-1)$. This is to add an association v_P on the edge e_H.



In case of duplication without host switching, both e_{P_1} and e_{P_2} (defined as in the previous case) need to be on e_H. Therefore, the cost for duplication without host switching is

- $A(e_{P_1}, e_H, t-1) + A(e_{P_2}, e_H, t-1) + \text{cost}_{\text{DUP}}$
  The solution will include association of $A(e_{P_1}, e_H, t-1)$, $A(e_{P_2}, e_H, t-1)$, and v_p on e_H.

Now, in case we also have host switching, then one of the edge e_H will be changed to a different edge $e_{H'}$. As a result, the cost is the minimum, for all active $e_{H'}$, of

- $A(e_{P_1}, e_H, t-1) + A(e_{P_2}, e_{H'}, t-1) + \text{cost}_{\text{DUP}} + \text{cost}_{\text{HS}}(e_H, e_{H'})$
  The solution will include association of $A(e_{P_1}, e_H, t-1)$, $A(e_{P_2}, e_{H'}, t-1)$, and v_p on e_H.
- $A(e_{P_1}, e_{H'}, t-1) + A(e_{P_2}, e_H, t-1) + \text{cost}_{\text{DUP}} + \text{cost}_{\text{HS}}(e_H, e_{H'})$
  The solution will include association of $A(e_{P_1}, e_{H'}, t-1)$, $A(e_{P_2}, e_H, t-1)$, and v_p on e_H.

The optimal solution can be obtained simply from the minimum of all these cases (no events, duplication only, and duplication with host switching). Only the case of duplication with host switching takes $O(n)$ since we need to look for the host to switch to. So the calculation of each cell is $O(n)$. Note that to obtain the active edges also take $O(n)$, but need to be done only once in each iteration of t.

Total running time

We need to fill in all $O(n^3)$ cells (A and B), and the highest running time needed is $O(n)$ in case of duplication with host switching. Therefore, the total running time is $O(n^4)$.

Implementation

The implementation should include the outer loop for time, with two loops inside – one for calculation of B from A and another for A from B. For each loop, go through the edges in E_P first in post-order. Then go through E_H in any order, and calculate the value as described above.

In each calculation of A from B, we can simply set the initial value to ∞, and try to update the value (and association) as we look at each possible case. The same model used in Jane for solution storing should work fine.

In calculation of B from A, however, we may need to construct an array for A to avoid using the speciation on the host tree more than once, and then free B later. This way, the memory needed is $O(n^2)$ cells. We should also consider calculating the list of active host edges at the beginning of the iteration or even keep track of it the whole time instead of looping through the edges every time. We can also pre-compute the post-ordering of the parasite edges for convenience.

cost_{HS}, as long as it takes no more than $O(n^4)$, can be pre-computed without increasing the running time. The candidates for the final solution are A and B with e_P that has dummy root on one end, for any e_H and t.

Note that since we create dummy edges, there is a bijection between edges and nodes. In implementation, it should be easier to refer to edges by its endpoint with higher time (the child).

Further optimization with extra assumptions

Let us consider making the following assumptions:

1. There is no limit in host-switch distance.
2. The host-switch cost is uniform.

Then, notice that the cost for host-switch does not depend on the location of the hosts. In the process of computing the A table, consider the candidate of $A(e_P, e_H, t)$ for host-switch where e_{P_2} switches form e_H to $e_{H'}$, which is

$$\min_{\text{active } e_{H'}}\left\{A\left(e_{P_1}, e_H, t\right) + A\left(e_{P_2}, e_{H'}, t\right) + \text{cost}_{\text{DUP}} + \text{cost}_{\text{HS}}(e_H, e_{H'})\right\}.$$

Since cost_{HS} does not depend on the take-off and landing sites, the formula becomes

$$\min_{\text{active } e_{H'}}\left\{A\left(e_{P_1}, e_H, t\right) + A\left(e_{P_2}, e_{H'}, t\right) + \text{cost}_{\text{DUP}} + \text{cost}_{\text{HS}}\right\}.$$

Since we fill the dynamic-programming table in post-order (reversed topological order) of parasite edges, we will have filled out the values for $A\left(e_{P_1}, e_H, t\right)$ and $A\left(e_{P_2}, e_H, t\right)$ for all e_H when we compute $A(e_P, e_H, t)$. So, $A\left(e_{P_1}, e_H, t\right)$ can be obtained in $O(1)$ running time, and is a constant when we compute $A(e_P, e_H, t)$. As a result, we obtain the candidate

$$A\left(e_{P_1}, e_H, t\right) + \text{cost}_{\text{DUP}} + \text{cost}_{\text{HS}} + \min_{\text{active } e_{H'}} A\left(e_{P_2}, e_{H'}, t\right).$$

So, in order to minimize this candidate, we need to find the minimum value of $A\left(e_{P_2}, e_{H'}, t\right)$ over all active $e_{H'}$. Instead of looping through all active $e_{H'}$, we can keep track of the minimum value of $A\left(e_{P_2}, e_{H'}, t\right)$ that we have found so far as we compute them. Therefore, we will construct an additional table $D(e_P, t)$ as following:

- The original value of each cell of $D(e_P, t)$ is ∞.
- As we compute $A(e_P, e_H, t)$ for any e_H, we update the cost of $D(e_P, t)$ to $A(e_P, e_H, t)$ if the new cost is lower.

With this, we can compute the optimal host-switch cost where e_{P_2} switches to some active host in $O(1)$ via the formula

$$A\left(e_{P_1}, e_H, t\right) + \text{cost}_{\text{DUP}} + \text{cost}_{\text{HS}} + D\left(e_{p_2}, t\right).$$

In the same fashion, the optimal host-switch cost where e_{P_1} switches to some active host is

$$A\left(e_{P_2}, e_H, t\right) + \text{cost}_{\text{DUP}} + \text{cost}_{\text{HS}} + D\left(e_{p_1}, t\right).$$

Occasionally, the optimal cost may be obtained when the take-off and landing sites are the same. If cost_{HS} is non-negative, then duplication will give a better cost. However, in case we allow cost_{HS} to be negative, then we will need to keep track of the best **two** values of $D(e_P, t)$, and also the landing site of the best one. This will allow us to check and prevent host-switch from and to the same edge within constant running time.

As a result, the computation of each value in A table can be done in $O(1)$ running time. Therefore, the total running time of the algorithm becomes $O(n^3)$.

Optimizations while allowing Jane's features

Please read the timing incompatibilities write-up before reading this section. We will reuse the notations from that write-up. Also, that write-up is on a more simple case of this version.

In this version, we will still support the following features:

- Dividing the host nodes/edges into regions, and allowing additional host-switch costs between (not necessarily distinct) regions
- Limited host-switch distance

We will create the C and D tables in the same way we used in the case where timing incompatibilities are allowed. However, to allow the two mentioned features, we will add more dimensions to the tables. So, C and D will have five parameters (e_P, e_H, t, d, r) where

- $e_P \in E_P$, the set of parasite edges
- $e_H \in E_H$, the set of host edges
- $t \in \{0, \dots, n_H\}$, the set of all possible time slices
- $d \in \{0, \dots, k\}$, the set of all possible host-switch distance, where k is the largest host-switch distance allowed but no higher than $2h - 1$ where h is the height of the tree
- $r \in R$, the set of regions

that determines the set of candidates $A(e_P, e_{H'}, t)$.

$C(e_P, e_H, t, d, r)$ is the minimum value of $A(e_P, e_{H'}, t)$ for all $e_{H'}$ such that $e_{H'}$

- is e_H or its descendant
- is alive right after time t
- is exactly at distance d from e_H
- is in region r

This can be written recursively as follows:

- If $d = 0$, e_H is alive right after time t, and is in region r, then $C(e_P, e_H, t, d, r) = A(e_P, e_H, t)$.
- Otherwise, if $d \geq 0$ and v_H is not a tip, then $C(e_P, e_H, t, d, r) = \min\{C(e_P, e_{H_1}, t, d-1, r), C(e_P, e_{H_2}, t, d-1, r)\}$.
- Otherwise, $C(e_P, e_H, t, d, r) = \infty$.

Therefore, we can compute each cell of $C(e_P, e_H, t, d, r)$, for each e_P after the table A is computed for this e_P, in post-order of e_H, any order of t, d and r, in $O(1)$. So, for each e_P, the computation takes $O(n^2 k|R|)$, which will contribute to the running time of $O(n^3 k|R|)$ in total.

$D(e_P, e_H, t, d, r)$ is the minimum value of $A(e_P, e_{H'}, t)$ for all $e_{H'}$ such that $e_{H'}$

- is not e_H, its descendant, or its ancestor; i.e., without constraints on distance, e_P is allowed to switch from e_H to $e_{H'}$
- is alive right after time t
- is exactly at distance d from e_H
- is in region r

This can be written recursively as follows:

- If e_H is the dummy root edge, then $D(e_P, e_H, t, d, r) = \infty$.
- Otherwise, $D(e_P, e_H, t, d, r) = \min\left\{C\left(e_P, e_{H_s}, t, d-1, r\right), D\left(e_P, e_{H_p}, t, d-1, r\right)\right\}$.

Therefore, we can compute each cell of $D(e_P, e_H, t, d, r)$, for each e_P after the table A is computed for this e_P, in pre-order of e_H, any order of t, d and r, in $O(1)$. Again, for each e_P, the computation takes $O(n^2k|R|)$, and the running time is $O(n^3k|R|)$ in total.

With D, we can compute the candidate for host-switch at $A(e_P, e_H, t)$ as follows:

$$\mathrm{HS}(e_P, e_H, t) = \min_{d,r}\left\{\begin{matrix} A\left(e_{P_1}, e_H, t\right) + D\left(e_{P_2}, e_H, t, d, r\right) + \mathrm{cost}_{\mathrm{HS}}(R(e_H), r), \\ A\left(e_{P_2}, e_H, t\right) + D\left(e_{P_1}, e_H, t, d, r\right) + \mathrm{cost}_{\mathrm{HS}}(R(e_H), r) \end{matrix}\right\},$$

where $R(e_H)$ is the region of e_H. So, each cell $A(e_P, e_H, t)$ can be computed in $O(k|R|)$, and still results in $O(n^3k|R|)$ in total.

The overall algorithm can be summarized as shown below:

```
For each time $t$ in decreasing order
    For each parasite edge $e_P$ in post-order
        For each host edge $e_H$ in any order
            Compute $B(e_P, e_H, t)$
        For each host edge $e_H$ in any order
            Compute $A(e_P, e_H, t-1)$
        For each host edge $e_H$ in post-order, for each $t$ and $r$ in any order
            Compute $C(e_P, e_H, t-1, d, r)$
        For each host edge $e_H$ in post-order, for each $t$ and $r$ in any order
            Compute $D(e_P, e_H, t-1, d, r)$
```