Pomona College
Department of Computer Science

The Cophylogeny Reconstruction Problem

Benjamin Fish

May 5, 2013

Submitted as part of the senior exercise for the degree of
Bachelor of Arts in Computer Science

Ran Libeskind-Hadas, advisor

Abstract

The event-cost method for the cophylogeny reconstruction problem has been well studied, but current solutions to the problem fail to represent the full scope of possible biological events. One of those possibilities is called a *failure to diverge* event. In this paper we modify a known genetic algorithm called Jane to handle these events, under certain circumstances. Our algorithm solves these problems in polynomial time, and we prove its correctness. Another shortcoming of existing algorithms is that they can only handle host-parasite relationships between two species. In response, we introduce the n-tangled cophylogeny reconstruction problem, which can represent the coevolution of many species, whose relationships do not necessarily need to be that of host-parasite.

Contents

Abstract i
List of Figures v

1 Introduction **1**
1.1 Cophylogeny Reconstruction 2

2 Modifying Jane **7**
2.1 Proof of the correctness of the algorithm 11

3 The *n*-Tangled Cophylogeny Reconstruction Problem **17**

4 Future Work **25**

Bibliography **27**

List of Figures

1.1 Types of events 3
1.2 Example cophylogeny reconstruction instance 3
1.3 Two mappings of the Cootie tree onto the Groody tree 4
1.4 Example of a problem requiring an FTD event 5

2.1 P' is modified to create P'_1 and P'_2 9
2.2 Original parasite subtree showing induced timing 10
2.3 Fixing timing inconsistencies 11
2.4 Example problem 11
2.5 Example problem - locating the FTD event 12
2.6 Example problem - modified parasite trees 12
2.7 Example problem - two matchings 13
2.8 Example problem - final solution with modified timing 13
2.9 Showing that P' must map to H' 14
2.10 Showing all of P' is mapped to H', continued 15

3.1 Cophylogeny reconstruction problem without symmetry . . . 18
3.2 Larger cophylogeny reconstruction problem without symmetry 18
3.3 Example of a 2-tangle 19
3.4 Example of two duplications 19
3.5 Event possible when associating two vertices (Cospeciation) . 21
3.6 Events possible when associating a vertex and an edge 22

Chapter 1

Introduction

An important biological problem is that of determining if two species coevolved, and if they did, how they might have done so. More specifically, of importance is the ability to reconcile two phylogenetic trees, a host tree and a parasite tree. This is called the cophylogeny reconstruction problem. Many attempts to solve this problem use event-cost methods. These methods seek to minimize a cost with respect to some metric, usually with a cost for each event where the parasite tree doesn't naturally match with the host tree. There are many possible different types of events. Most commonly, there are four event types considered: cospeciations, host switches, losses, and duplications.

Various methods have been used to solve this problem. [2] does so optimally, but takes exponential time in the worst case. TreeFitter, Tarzan, CoRe-Pa, Jane, and others use heuristics instead [10, 7, 8, 3, 1, 4]. They differ from each other in several ways. They all take polynomial time, but some don't allow all four different events to occur, such as [4], which only allows duplication and loss events. Others, such as [1], allow solutions that have timing inconsistencies.

In this paper, we improve on the Jane algorithm, which is a genetic algorithm using the event-cost model with all four of the aforementioned event types [3]. We improve the algorithm by describing a modification that results in a polynomial-time algorithm that also includes the ability to use another kind of event, called a failure to diverge event.

We also consider a novel problem that generalizes the cophylogeny reconstruction problem, which we call the n-tangled cophylogeny reconstruction problem. We motivate this new problem by giving an example where the event-cost method results in a mapping that is not symmetric. That is, the minimum-cost mapping of some tree A onto some tree B does not have the same minimum-cost mapping as B onto A. To fix this, we introduce the n-tangled cophylogeny reconstruction problem, which not only has symmetry, but is able to both describe a greater variety of coevolutionary possibilities and describe the coevolution of an arbitrary number of species. We motivate the introduction of this problem by giving an example of a pair of trees such that the optimal solution of the first tree onto the second does not have the same cost as the optimal solution of the second tree onto the first. We also give a polynomial time algorithm for the timed version of the 2-tangled cophylogeny reconstruction problem.

In Section 1, we introduce the cophylogeny reconstruction problem and an approach to solve it called Jane. We also introduce the failure to diverge event. In Section 2, we describe an algorithm for solving the cophylogeny reconstruction problem with failure to diverge events. We then prove its correctness under certain restrictions. In Section 3, we introduce the n-tangled cophylogeny reconstruction problem, motivate its introduction, and give a polynomial time algorithm for the timed version of the 2-tangled cophylogeny reconstruction problem. Finally, in Section 4, we contemplate avenues for future research.

1.1 Cophylogeny Reconstruction

In the cophylogeny reconstruction problem, we are given a host tree, a parasite tree, a mapping between the leaves, or 'tips', of the two trees which represents the relationship between extant taxa, and a cost associated with each possible event type. The four possible event types are the following: Cospeciation is when a speciation event in the parasite tree occurs at the same time as a speciation event in the host tree, so that two non-leaf nodes in the trees are identified (Figure 1.1 (a)). Duplication is when a speciation event in the parasite tree occurs *between* speciation events in the host tree, so that a node of the parasite tree is identified with an edge of the host tree (Figure 1.1 (b)). A host switch is a duplication followed by one of the two descendants of the node in the parasite tree moving to a different edge on the host tree (Figure 1.1 (c)). A loss is when a speciation event in the host tree occurs between speciation events in the parasite tree, so an edge of the parasite tree is identified with a node of the host tree (Figure 1.1 (d)).

Under the event-cost model, the objective of the problem is to find the mapping of the parasite tree to the species tree that minimizes the total cost, where there is a cost associated with each of the four event types. When

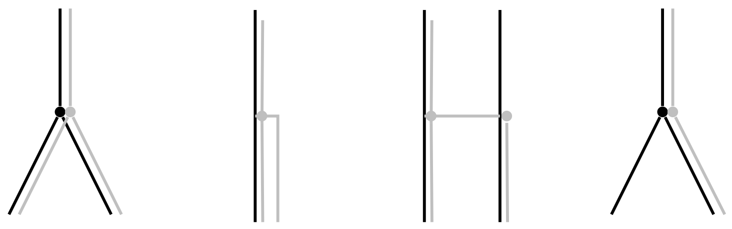

(a) Speciation (b) Duplication (c) Host Switch (d) Loss

Figure 1.1: Types of events

all of these events are allowable, this is an NP-complete problem [6]. For more general information on the cophylogeny reconstruction problem and its variants, see [9, 5].

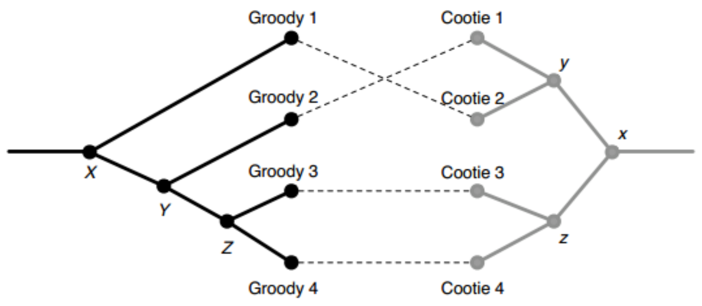


Figure 1.2: Example cophylogeny reconstruction instance

An example is given in Figure 1.2. The dashed lines represents the tip mapping. Since the two trees are not isomorphic, it is necessary to introduce events to map the Cootie tree, the parasite, onto the Groody tree, the host. Figure 1.3 gives two possible mappings, i.e. solutions to the cophylogeny reconstruction problem. Each use a different set of events, leading to different-cost solutions.

1.1.1 Jane

Because the problem is NP-complete in general, Jane uses a genetic algorithm [3]. It maintains a population of explicit timings of the host tree. An explicit timing of the host tree is one where all non-tip vertices are totally ordered, so no two vertices are allowed to be at the same time. This allows

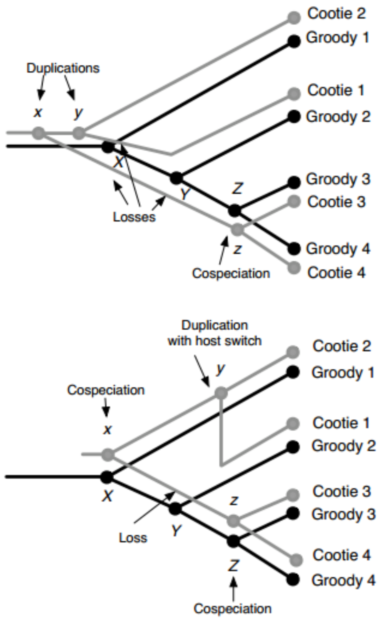


Figure 1.3: Two mappings of the Cootie tree onto the Groody tree from Figure 1.2

Jane to forbid timing inconsistencies to occur. A timing inconsistency is when the placement of host switches imply that some vertex x occurs before another vertex y but also that y occurs before x. Timing inconsistencies may arise in some approaches to this problem, such as [1], but Jane does not allow them. On each of these timed trees, it then employs a dynamic programming solution to map the parasite tree onto the host tree. The cost given by this solution serves as a fitness function for the population of timed host trees. New timings are created with a crossover operator. This crossover randomly changes the timing of a subtree.

1.1.2 Failure to Diverge Events

One goal of this thesis is to modify the Jane algorithm to include another type of event, called a failure to diverge (FTD). This is when a parasite species associated with a host species continues to associate with both of that host's children. That is, a single edge of the parasite tree is identified with two different edges of the host tree.

In Figure 1.4, the b parasite species maps to two host species. Thus an

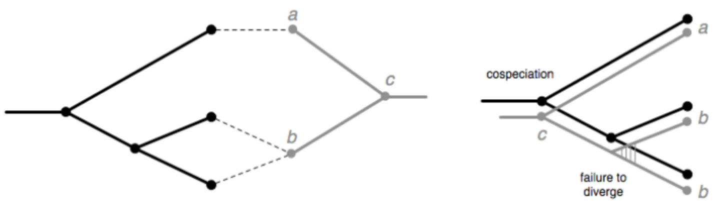


Figure 1.4: Example of a problem requiring an FTD event

FTD event is required to explain the mapping. On the right, an example of a solution is shown.

Chapter 2

Modifying Jane

We now wish to modify the Jane algorithm to include failure to diverge events. Now, the mapping of the parasite tree no longer needs to be isomorphic to the original parasite tree. We propose a model to represent this that is both biologically-reasonable and general: when the parasite speciates, it may speciate from either of the edges representing the ancestors of the tips that failed to diverge, but not both.

The strategy this thesis takes to solve the cophylogeny reconstruction problem with failure to diverge events is to leave the genetic algorithm alone. Only the fitness function is changed. It now consists of a wrapper around the dynamic programming that runs multiple iterations of the dynamic programming algorithm. This approach finds the optimal mapping given a explicitly timed host tree. The optimal mapping may include timing inconsistencies, but the algorithm guarantees a mapping that has the same cost but no timing inconsistencies by changing the timing of the host tree.

Thus, given an explicitly timed host tree, we wish to find a polynomial-time algorithm that finds the minimum cost matching between the host tree and the parasite tree.

In order to do this, we must impose several restrictions on the matching. The most important of these is that we only want to consider host switches when they are absolutely necessary. Thus the cost of a host switch must be sufficiently high to discourage their use. It turns out that the cost of a host switch must be high enough so that some portion of the parasite tree can undergo duplications instead of host switching from and then back to an edge of the host tree. The specific value of this cost is unimportant, so we will refer to this cost as being 'arbitrarily high.'

Furthermore, we assume that the locations of all FTD events are fixed. It is clear that when one parasite tip is mapped to more than one host tip, there must be a FTD event. To simplify the algorithm, we assume that the event must occur at the vertex that is the least common ancestor of those host tips.

Let v be the vertex in a host tree that is the least common ancestor of two tips mapped to the same parasite tip. Let H' be the subtree whose root is immediately above v. Since the host tree is binary, H' in turn has two subtrees. Given H', we call these two subtrees H'_L and H'_R.

Lastly, we restrict the problem so that no subtree of the parasite has tips mapped to both H'_L and H'_R of the host subtree or outside the host subtree. This prevents us from requiring host switches.

In summary, the restrictions are:

1. Host switches have arbitrarily-high cost.
2. FTD events are forced to occur at the least common ancestor of the two host switches mapped to the same parasite tip.
3. Let P' be the smallest subtree of the parasite whose tips are mapped to H', the host subtree rooted immediately above the FTD event. P' has no tips mapped outside of H', and for every subtree T of P' that is rooted at an ancestor of the tip mapped to the two host tips, T' does not have tips mapped to both H'_L and H'_R.

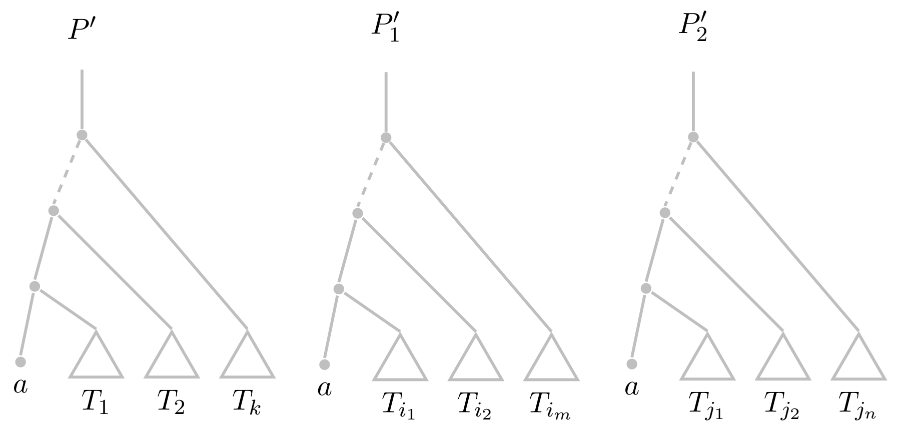
We will refer to these as restrictions 1., 2., and 3.

Assuming these restrictions hold, we now present an algorithm to find the least cost mapping of parasite tree to a timed host tree. It is a modification of the dynamic programming in Jane.

Algorithm 1.

1. Find the least common ancestor for every pair of host tips mapped to the same parasite tip. These vertices are the locations of the FTD events. We will run the DP algorithm of Jane on each subtree of the host rooted immediately above these vertices. Call this subtree of the host H'. For each H', we consider the smallest subtree of the parasite whose tips map only to this subtree of the host. Call this subtree P'.
2. We now modify P' by 'splitting' it into two trees: Let a be the tip of the parasite that is mapped to two host tips. Let $T_1, \ldots T_k$ be the subtrees of the parasite tree rooted at the vertices of the ancestors of a. This tree is split in two, called P'_1 and P'_2: The new trees both have an edge for a, and subtrees coming off of that edge. If T_i has tips in

H'_L, then it is one of the subtrees coming off the a edge in P'_1, and similarly if the tips map to H'_R, then it is one of the subtrees coming off the a edge in P'_2. This is shown in Figure 2.1. Note that every T_i is assigned to either P'_1 or P'_2. All of the tips of P'_1 map to H'_L, and similarly P'_2 to H'_R. Then the DP algorithm is run on the H'_L subtree of the host with P'_1 and the H'_R subtree of the host with P'_2.



(a) Original parasite subtree

(b) New parasite subtree whose tips map to H'_L

(c) New parasite subtree whose tips map to H'_R

Figure 2.1: P' is modified to create P'_1 and P'_2

3. This matching may create a timing inconsistency. We fix this by moving the vertices of the host subtree. The original parasite subtree induces a timing given by the order of the subtrees off of the a edge. This is shown in Figure 2.2. x_1 must occur after x_2, which must occur after x_3, etc. But this timing is not necessarily respected in the mapping, as shown in Figure 2.3 (a). For example, if x_{j_1} must happen before x_{i_1}. To fix this, we change the timing of the host tree. Specifically, we change the timing of the host tree vertices mapped to each x_i. The timing is changed to respect the ordering of the x_i's given by the parasite tree. For example, in Figure 2.3 (b), x_{i_2} occurs after x_{j_1}, so the timing is changed to reflect that. The timings of the T_i's are left the same.

   This is done for each host subtree induced by an FTD event, starting with the host subtree closest to the tips.

4. After, the DP is run on the entire host tree (with the modified timings)

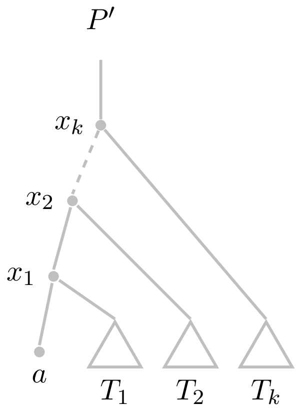


Figure 2.2: Original parasite subtree showing induced timing

and parasite tree, modified in the following manner: for every edge and event involved in an above mapping, the DP only considers mapping the edge in the manner given by the above mappings. For other edges, the DP runs as it would in the original DP algorithm.

Example 2. Now we give a small example of how the algorithm runs. The example problem is shown in Figure 2.4.

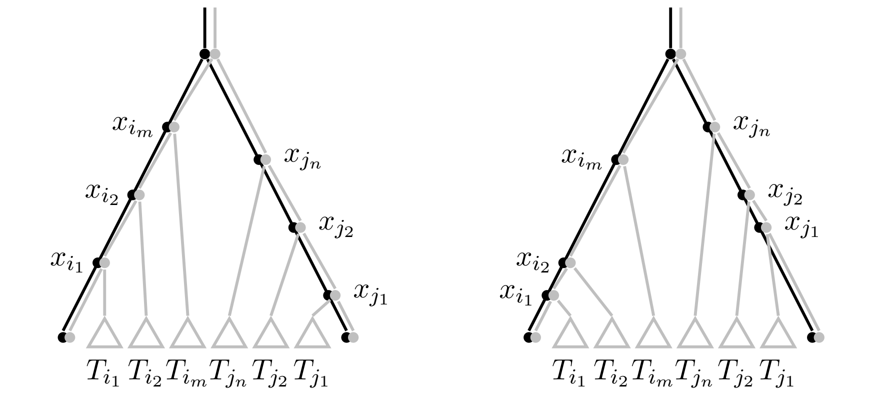
1. Since there is only one parasite tip mapped to two host tips, there is only one FTD event, located at the least common ancestor of the two host tips. This is shown in Figure 2.5.

2. The FTD event occurs at the top of P, so $P = P'$. P is then modified into two trees P_1 and P_2. This is shown in Figure 2.6.

   Then we use the DP to give the lowest cost solution. The solution that has the smallest cost depends on the costs associated with each event types. Two solutions are given in Figure 2.7. For the sake of the example, suppose the one on the right is minimum-cost.

3. Unfortunately, this solution has a timing inconsistency. Recall from Figure 2.4 that x_b follows x_c. But in the solution x_c follows x_b. To fix this, we switch the order of the host vertices mapped to x_b and x_c. This results in the mapping seen in Figure 2.8.

4. Since this is the entire tree, we are done, and the final solution is Figure 2.8.



x_{i_m}
x_{j_n}
x_{i_2}
x_{j_2}
x_{i_1}
x_{j_1}
T_{i_1} T_{i_2} T_{i_m} T_{j_n} T_{j_2} T_{j_1}
x_{j_n}
x_{i_m}
x_{j_2}
x_{j_1}
x_{i_2}
x_{i_1}
T_{i_1} T_{i_2} T_{i_m} T_{j_n} T_{j_2} T_{j_1}

(a) Mapping with timing inconsistency (b) Mapping without timing inconsistency

Figure 2.3: Fixing timing inconsistencies

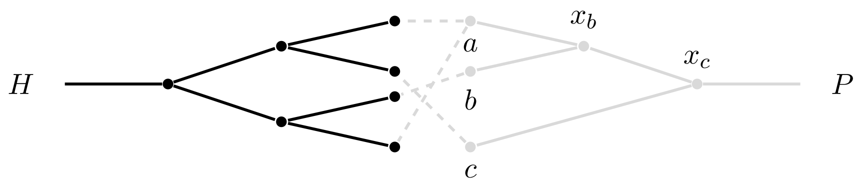


Figure 2.4: Example problem

2.1 Proof of the correctness of the algorithm

To prove that this algorithm is correct, we need to show several claims. First, we show that for a given cophylogeny reconstruction problem with timed host tree, the algorithm finds the optimal solution (but it may have timing inconsistencies). Second, we show that any timing inconsistency can be removed without changing the cost of the solution. And finally, we show that this algorithm takes polynomial time.

We start by giving an outline of the proof of the first of these, that for a given cophylogeny reconstruction problem with timed host tree, the algorithm finds the optimal solution. In order to do this, we prove that the optimal solution is a solution where the modified subtree of the parasite given by the algorithm is mapped to the subtree rooted at the FTD event. This is sufficient, as the dynamic programming solution in Jane is known to be correct. This is because the subproblem of mapping the modified

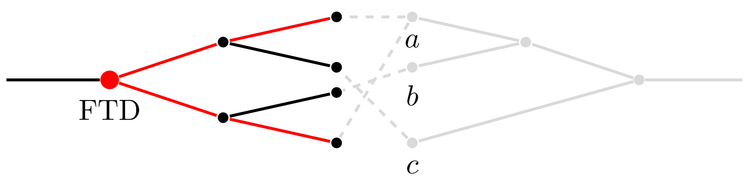


Figure 2.5: Example problem - locating the FTD event

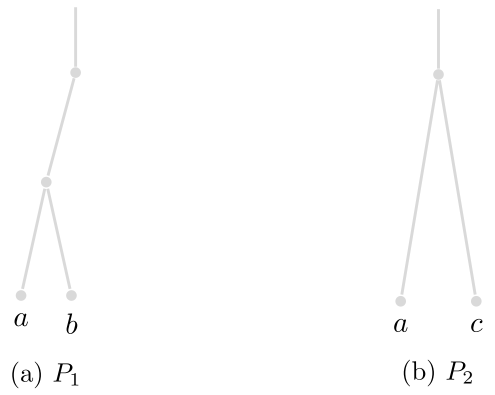


Figure 2.6: Example problem - modified parasite trees

subtree of the parasite onto the subtree of the host rooted at the FTD event is solved by the dynamic programming algorithm. Furthermore, the correctness of the final solution for the problem is correct given the two claims above, that each FTD subproblem is solved correctly, and that the minimum-cost solution includes each of these mappings.

Thus we must prove that the optimal solution is a solution where the modified subtree of the parasite given by the algorithm is mapped to the subtree rooted at the FTD event. There are two possible ways in which this could not occur. The first is that the optimal solution maps at least part of the parasite tree to any other part of the host tree besides the subtree rooted at the FTD. The second is that while the parasite tree is mapped to the correct subtree of the host, it is not the modification of the parasite tree given by the algorithm. We start with the first:

Proposition 3. *Let H' be the subtree of the host tree rooted immediately above the FTD event. Let P' be the subtree of the parasite whose tips are mapped entirely to H'. Then the optimal solution maps P' to H'.*

Proof. Suppose P' is mapped somewhere else besides H' or an ancestor

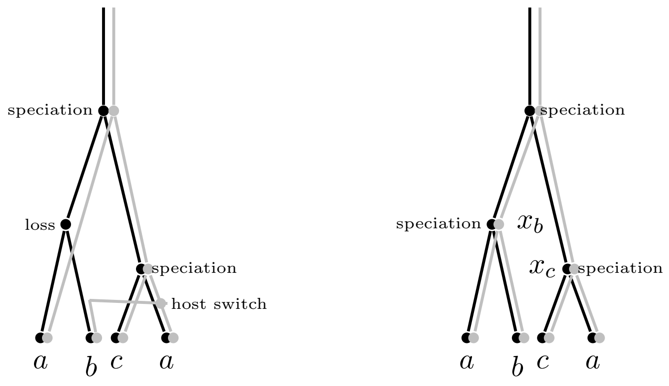


Figure 2.7: Example problem - two matchings

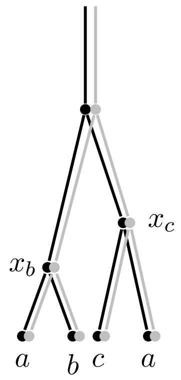
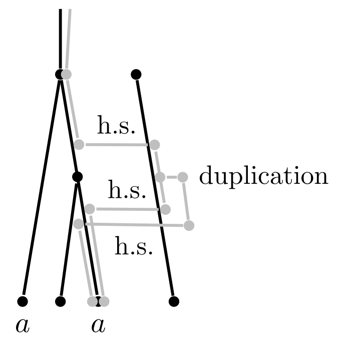


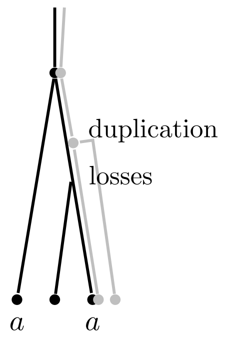
Figure 2.8: Example problem - final solution with modified timing

of H'. From restriction 2., the parasite subtree is forced to be rooted at H' or an ancestor of a H'. From restriction 3., there are no tips mapped to anywhere else besides H', so if part of P' is mapped anywhere else, it must eventually be mapped back to H' afterwards. This requires at least two host switches. Since from 1., host switches have arbitrarily-high cost, a lower cost-solution would be to remove both of these host switches and to have never left H' (or an ancestor of H'). Note this may involve using additional events, like duplications and losses, but it will still have lower cost. Figure 2.9 shows an example of this. Note in general, there may not be a single duplication but an entire mapping elsewhere in the tree.

It remains to show that P' is not mapped to an ancestor of H'. Again from restriction 2., the only events that would allow P' to be mapped above



(a) P' leaves H' using host switches

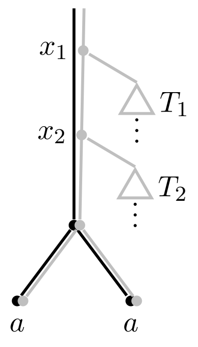


(b) Host switches are removed, introducing other events

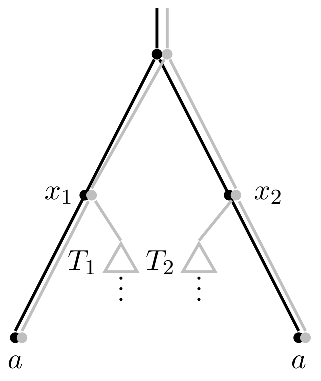
Figure 2.9: Showing that P' must map to H' - removing the host switches introduces losses

the FTD event are duplications. Suppose some subset of P' is mapped above the FTD event with duplications. This causes the solution to have a loss at the FTD event for every duplication that occurs. But this solution can be improved, removing those loss events: For every subtree T of P' rooted at an ancestor of the parasite tip mapped to two host tips, the corresponding node on the ancestor that lies above T may duplicate above the FTD event. That duplication may be moved below the FTD event, since by restriction 3., all of its tips must map to either H'_L or H'_R of H'. This is depicted in Figure 2.10. In Figure 2.10 (a), each of the edges from T_1 and T_2 may undergo loss events at the vertex where the failure to diverge event occurs. By moving them down, as shown in Figure 2.10 (b), none of the loss events happen. This cannot increase the overall cost, so an optimal solution must include this modified mapping. $\square$

Now we show that not only is the parasite subtree tree mapped to the correct subtree of the host, it is the modification of the parasite tree given by the algorithm. Why does the parasite tree need to be modified at all? This is a consequence of how we model FTD events. When the parasite speciates, it may speciate from either of the edges representing the ancestors of the tips that failed to diverge, but not both. This leads to the kind of 'split' given in the algorithm: Let a be the tip of P' that is mapped to two host tips. Let $T_1, \dots T_k$ be the subtrees of the parasite tree rooted at the vertices of the ancestors of a. This tree is split in two trees, called P'_1 and P'_2: The new trees both have an edge for a, and subtrees coming off of that edge.



(a) Part of P' is mapped above H'



(b) T_1 and T_2 is moved below the failure to diverge event

Figure 2.10: Showing all of P' is mapped to H' - Moving P' to below the root of H'

The subtrees are $T_1, \ldots T_k$. However, there are exponentially (in k) ways to assign $T_1, \ldots T_k$ to P'_1 and P'_2. It remains to show that we only need to consider the 'split' given by the algorithm.

Proposition 4. *The minimum-cost solution includes mappings between H'_L and P'_1, and H'_R and P'_2.*

We have already shown that P' maps to H', and we force the FTD event to occur at the root node of H', so it remains to show that other 'splits' beside P'_1 and P'_2 must not yield lower-cost solutions.

This is due to the fact that any other 'split' of P' must give a solution with host switches. Let A and B be such a pair of trees based on the original parasite. Without loss of generality, there must be a tip of A that is mapped to a tip on the other host subtree, H'_R. In order for it to be a valid solution, the edge must at some point host switch to H'_R, as the two trees are mapped to H'_L and H'_R below the first speciation of the host subtree. Since there are no host switches allowed, there is no need to consider this pair of trees.

Now we have shown that the optimal solution must map each modified subtree of the parasite to H'. Thus, the algorithm finds the optimal solution, but it may have timing inconsistencies. We now show that any timing inconsistency can be resolved.

Proposition 5. *Any timing inconsistency of the mapping induced from the timing of the vertices given by the parasite tree can be resolved.*

We now outline the proof.

It is important to note that no other kinds of timing inconsistencies are possible either. Timing inconsistencies induced by incompatible host switches cannot be possible because we use the DP from Jane, which insures that these cannot happen.

First of all, there are no host switches between H'_L and H'_R of P', the smallest subtree of the parasite whose tips are mapped to H', the host subtree rooted immediately above the FTD event. This is shown above.

Now we resolve the timing inconsistencies. We do so by changing the timing of the host tree. As in the description of the algorithm, let $T_1, \dots T_k$ be the subtrees of the parasite tree rooted at the vertices of the ancestors of a. Any mapping of the parasite subtree to the host subtree must map the roots of each T_i to either of the two edges that represent the ancestors of a. Mapped to each of these is either a vertex of the host tree or an edge. However, the parasite tree induces a timing on each T_i such that T_{i-1} must come after T_i. But in the matching, if T_{i-1} is matched to one edge that represents a and T_i is matched to the other, this will not necessarily be the case. To fix this, we change the timing of the host tree to match the timing induced by the parasite tree. That is, we change the timing of the vertices and edges that are the ancestors of the tips mapped to a to the timings given by the timing of the T_i induced by the parasite tree. Changing timings can potentially invalidate a mapping. Specifically, a host switch from edge x to edge y may be invalidated if y occurs strictly before x. However, since there are no host switches between H'_L and H'_R, this cannot happen between the two subtrees. There may be host switches between two edges contained entirely in H'_L or H'_R, but the changed timing never changes the order between T_i's rooted on H'_L or H'_R. Thus the timings of the T_i's may be left alone. Hence timing inconsistencies can be resolved.

We now turn to the last thing that needs to be shown: that the algorithm takes polynomial time. There are a polynomial number of FTD events, as there are only a polynomial number of nodes in the tree. It is not difficult to find each one in polynomial time, as they are located at the least common ancestor. Creating the modified parasite tree takes polynomial time, as it just requires finding the smallest subtree whose tips map to the given host subtree. Running the DP takes polynomial time. Finally, fixing the timings takes polynomial time, as the required timing is induced by the parasite tree and hence fixing the timing just requires copying that timing. Thus the algorithm takes polynomial time.

This concludes the proof that the algorithm is correct and runs in polynomial time.

Chapter 3

The n-Tangled Cophylogeny Reconstruction Problem

While our formulation of the cophylogeny reconstruction problem is a powerful way of looking at cophylogeny, it is not all-encompassing. The five events considered in the previous chapter do not cover all biological possibilities. Two coevolving species may evolve in a mutually beneficial relationship rather than in a host-parasite relationship. In addition, it is possible for more than two species to coevolve. It is desirable to be able to model these relationships, which the current formulation does not allow us to do. This is due to the fact that in the current formulation, the host cannot undergo events such as host switches. Thus we introduce a new problem, called the n-tangled cophylogeny reconstruction problem. It is similar to the old formulation, except instead of mapping the parasite onto the host, now all trees are mapped onto the others, and there may be more than two of them. We call this mapping a tangle, as now both trees can undergo events.

This removes the asymmetry inherent in the old problem: The minimum-cost mapping of some tree B onto A is not necessarily equal to the minimum-cost mapping of A onto B. This kind of symmetry is highly desirable when the two species do not display a host-parasite relationship.

We show that the old problem does not have asymmetry with an example, given in Figure 3.1.

When the cost of cospeciation is 1, loss is 4, duplication is 1, and switch is 3, the minimal cost solution of A onto B does not have the same cost as the minimal cost solution of B onto A. This is the smallest known example of an assymetric mapping. However, there are many larger examples that can be created using this small one as the basis. For example, Figure 3.2

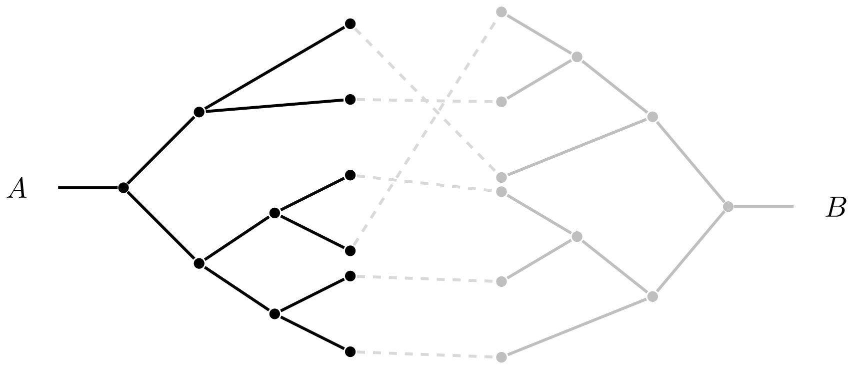


Figure 3.1: Cophylogeny reconstruction problem without symmetry

shows a way of creating larger examples. If T is an unbalanced subtree on both A and B where tip x on T on A maps to tip x on T on B, then Figure 3.2 is a cophylogeny reconstruction problem without symmetry. This shows that there are arbitrarily large examples of asymmetrical cophylogeny reconstruction problems.

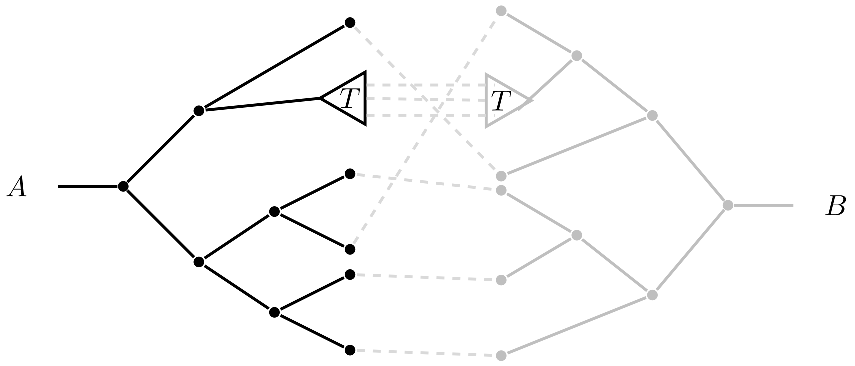


Figure 3.2: Larger cophylogeny reconstruction problem without symmetry

This motivates the n-tangled cophylogeny reconstruction problem. More formally, in the n-tangled cophylogeny reconstruction problem, we are given n trees, a mapping between the leaves, or 'tips', of the trees which represents the relationship between extant taxa, and a cost associated with each possible event. The five possible events, for all trees, are speciation, duplication, loss, host switch, and failure to diverge. The map between tips is given as a equivalence relation over tips, so that any number of taxa may be associated with each other.

Under the event-cost method for the problem, the objective of the problem is to find a mapping of the n trees together that minimize the total cost, where there is a cost associated with each of the five events.

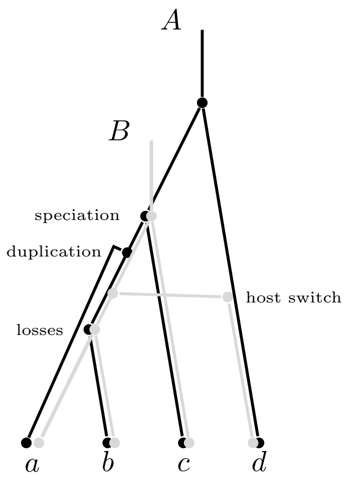


Figure 3.3: Example of a 2-tangle

Figure 3.3 is an example of a solution for a two-tangled cophylogeny reconstruction problem. It is important to note that this mapping can not be represented by the traditional cophylogeny reconstruction problem. For example, if we consider B to be the host, then how do we represent the host switch depicted? It is not clear. Other problems can arise while trying to do this kind of translation, such as host switching to a species that has duplicated. This demonstrates that this problem formulation has considerable more representative power.

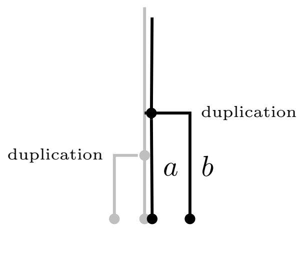


Figure 3.4: Example of two duplications

However, it is important to note that the costs of solutions to this problem cannot directly be compared to the costs of solutions to the original problem. Figure 3.4 gives a small example of a mapping with two duplications where all four tips are associated with each other. The cost of this mapping is the cost of not two, but three duplications: the second duplication is associated with both edge a and edge b, and hence must pay a duplication cost for each edge associated with it. While biologically reasonable, this makes costs of solutions difficult to compare with solutions given by the original problem.

The decision problem version of the n-tangled cophylogeny reconstruction problem is clearly in NP. Given a target cost, it is easy to check whether a tangle has that cost.

The n-tangled cophylogeny reconstruction problem also has a timed version, a parallel to the timed version of the original cophylogeny reconstruction problem that Jane solves. In the n-tangled timed cophylogeny reconstruction problem, each of the trees has an explicit timing, i.e. a total order on the internal nodes of the tree such that no two internal nodes can be at the same time.

We now present an algorithm to solve the 2-tangled timed cophylogeny reconstruction problem in polynomial time, as a first step toward exploring solutions to the more general problem. We restrict the type of events to cospeciations, duplications, host switches, and losses.

Here, we largely follow the notation used in [11]. As such, let $\textsc{cost}_{\text{co}}$, $\textsc{cost}_{\text{dup}}$, $\textsc{cost}_{\text{switch}}$, and $\textsc{cost}_{\text{loss}}$ denote the non-negative costs associated with cospeciation, duplication, host switch, and loss, respectively. Let A and B be the two trees, and a and b nodes or edges of the tree. Let $C(a, b)$ denote the cost of the optimal solution for the subproblem comprising of the two subtrees of A and B rooted at a and b respectively. The solution to the problem is then

$$\min_{a \in A_V, b \in B_V} \{C(a_r, b), C(a, b_r)\}$$

where a_r and b_r are the root nodes of A and B, and A_V and B_V are the nodes of A and B.

Following the strategy used in Jane, we treat C as a dynamic programming table. It then suffices to give a recursive algorithm to calculate $C(a, b)$ and an algorithm for computing the dynamic programming (DP) table.

Algorithm 6. The DP calculates C by considering edges and nodes in both trees bottom up. Since only nodes are given an ordering, not all edges are ordered, so that an edge exists through a range of times determined by its child and parent nodes. Thus there can be multiple edges that exist at a given time. It does not matter in the DP table which is calculated first. It

only matters that $C(a,b)$ is calculated after C is calculated for all nodes and edges that must be strictly after a and b. This suffices, as the algorithm we will describe always recurses on a smaller subtree than the tree rooted at a or a smaller subtree than the tree rooted at b.

Now we give a recursive algorithm for $C(a,b)$. First suppose that a and b are both nodes in A and B respectively. If they are both tips, then

$$C(a,b) = \begin{cases} 0 & \text{if } \phi(a) = b \\ \infty & \text{otherwise} \end{cases}$$

where ϕ is the tip matching function.

Let a_{e_1} and a_{e_2} be the child edges of a, and similarly let b_{e_1} and b_{e_2} be the child edges of b. Let a_{v_1} and a_{v_2} be the corresponding child nodes of a, and similarly let b_{v_1} and b_{v_2} be the corresponding child nodes of b. Now suppose a and b are both non-tip nodes. The only possible type of event is a cospeciation, shown in Figure 3.5.

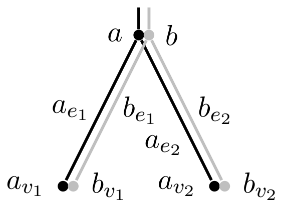


Figure 3.5: Event possible when associating two vertices (Cospeciation)

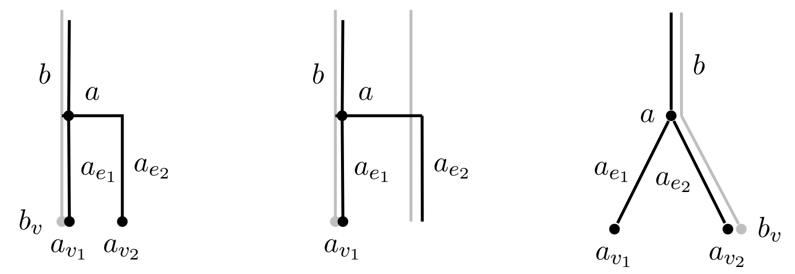
The cospeciation term given below considers the following possibilities: There are two ways to map the subtrees together: either a_{e_1} is associated with b_{e_1} or a_{e_1} is associated with b_{e_2}. For each of these possibilities, each of the two subproblems yield three possibilities for how they are associated: The child node of b is associated with a, a's child edge, or somewhere below a's child edge. In the last case, it then must be the case that the child node of a must be associated with the child edge of b. This yields 18 possibilities. Thus when both a and b are non-tip nodes, but they're not both tips,

$$C(a,b) = \begin{cases} \textsc{Co}(a,b) & \text{if } a \text{ and } b \text{ are not tips} \\ \infty & \text{otherwise} \end{cases}$$

where

$$\textsc{Co}(a,b) = \textsc{cost}_{\text{co}} + \min_{(i,j)\in\{(1,2),(2,1)\}} \left\{ \begin{array}{l} C(a_{v_1}, b_{v_i}) + C(a_{v_2}, b_{v_j}), \\ C(a_{v_1}, b_{e_i}) + C(a_{v_2}, b_{v_j}), \\ C(a_{e_1}, b_{v_i}) + C(a_{v_2}, b_{v_j}), \\ C(a_{v_1}, b_{v_i}) + C(a_{v_2}, b_{e_j}), \\ C(a_{v_1}, b_{e_i}) + C(a_{v_2}, b_{e_j}), \\ C(a_{e_1}, b_{v_i}) + C(a_{v_2}, b_{e_j}), \\ C(a_{v_1}, b_{v_i}) + C(a_{e_2}, b_{v_j}), \\ C(a_{v_1}, b_{e_i}) + C(a_{e_2}, b_{v_j}), \\ C(a_{e_1}, b_{v_i}) + C(a_{e_2}, b_{v_j}) \end{array} \right\}$$

Else one of a or b is a node and the other an edge. Without loss of generality, suppose a is the node and b the edge. Let b_v now denote the child node of b. In this case, there are three possible types of events: duplication, host switch, and loss. These possibilities are shown in Figure 3.6.



(a) Duplication (b) Host Switch (c) Loss

Figure 3.6: Events possible when associating a vertex and an edge

The duplication term must consider similar possibilities as the cospeciation term does. However, the duplication event associates both child edges of a to b, and because A has a total ordering on the vertices, the child nodes of a cannot occur at the same time. This results in 8 possibilities, given below.

The loss term considers 6 possibilities: There are two child edges b may be associated with, and for each of those, the child node of b is associated with a's child edge, a's child node, or somewhere below a's child node.

The switch term must consider a non-constant number of possibilities: Each child edge of a may be the edge that undergoes a host switch event.

The child edge of a that host switches may move to any edge in B at the same time as b. In regards to the other child edge of a, the child node of b is associated with that child edge of a, that child node of a, or somewhere below that node.

Hence when a is a node and b is an edge,

$$C(a,b) = \begin{cases} \min \begin{Bmatrix} \textsc{Dup}(a,b), \\ \textsc{Loss}(a,b), \\ \textsc{Switch}(a,b) \end{Bmatrix} & \text{if } a \text{ and } b \text{ aren't tips} \\ \infty & \text{otherwise} \end{cases}$$

where

$$\textsc{Dup}(a,b) = \text{COST}_{\text{dup}} + \min \begin{Bmatrix} C(a_{e_1}, b_v) + C(a_{e_2}, b_v), \\ C(a_{v_1}, b_v) + C(a_{e_2}, b_v), \\ C(a_{v_1}, b) + C(a_{e_2}, b_v), \\ C(a_{e_1}, b_v) + C(a_{v_2}, b_v), \\ C(a_{v_1}, b) + C(a_{v_2}, b_v), \\ C(a_{e_1}, b_v) + C(a_{v_2}, b), \\ C(a_{v_1}, b_v) + C(a_{v_2}, b), \\ C(a_{v_1}, b) + C(a_{v_2}, b) \end{Bmatrix},$$

$$\textsc{Loss}(a,b) = \text{COST}_{\text{loss}} + \min_{i \in \{1,2\}} \begin{Bmatrix} C(a_{e_i}, b_v), \\ C(a_{v_i}, b_v), \\ C(a_{v_i}, b) \end{Bmatrix},$$

and

$$\textsc{Switch}(a,b) = \text{COST}_{\text{switch}} + \min_{(i,j) \in \{(1,2),(2,1)\}} \left\{ \min \begin{Bmatrix} C(a_{e_i}, b_v), \\ C(a_{v_i}, b_v), \\ C(a_{v_i}, b) \end{Bmatrix} + \min_{b' \in B_E{}'} \{C(a_{v_j}, b')\} \right\}$$

where $B_E{}'$ is the set of edges in B that exist at the same time as b. Note C is undefined for two edges. This is done to simplify the algorithm, but because the above two cases (for two vertices and for a vertex and an edge) do not recursively compute C for two edges, we are able to maintain correctness.

Let the size of the larger tree be n. Then this algorithm has a worst-case running time of $O(n^3)$: There are $O(n)$ vertices and edges in each of A and B, so the DP table is size $O(n^2)$. Calculating each of $\textsc{Co}(a,b)$, $\textsc{Dup}(a,b)$, and $\textsc{Loss}(a,b)$ takes a constant number of look-ups, and $\textsc{Switch}(a,b)$ takes (worst-case) $O(n)$ look-ups. Hence the worst-case running time is $O(n^3)$, which is certainly polynomial time.

Chapter 4

Future Work

While we have made significant progress in the cophylogeny reconstruction problem, there is much more work to do. First, we suspect that the cophylogeny reconstruction problem with timed host trees and failure to diverge events can be solved in polynomial time without the three restrictions we impose. However, it appears that such an algorithm would have to take a different tack than the one employed in this thesis.

Second, there is much more work to be done on the tangled cophylogeny reconstruction problem. We believe it is NP-complete, but that has yet to be shown. As with the cophylogeny reconstruction problem, it also appears likely that a genetic algorithm like Jane may give efficient results for this problem. We have started to work toward this by giving an algorithm for the 2-tangled timed cophylogeny reconstruction problem.

This new formulation of the cophylogeny reconstruction problem may yield additional insight into the coevolution of species not defined by a host-parasite relationship. This thesis starts that process while continuing to expand our ability to solve the traditional cophylogeny reconstruction problem.

Bibliography

[1] Mukul S. Bansal, Eric J. Alm, and Manolis Kellis. Efficient algorithms for the reconciliation problem with gene duplication, horizontal transfer and loss. *Bioinformatics*, 28(12):i283–i291, 2012.

[2] MA Charleston. Jungles: a new solution to the host/parasite phylogeny reconciliation problem. *Mathematical biosciences*, 149(2):191–223, 1998.

[3] Chris Conow, Daniel Fielder, Yaniv Ovadia, and Ran Libeskind-Hadas. Jane: a new tool for the cophylogeny reconstruction problem. *Algorithms for Molecular Biology*, 5(1):16, 2010.

[4] Dannie Durand, Bjarni Halldrsson, and Benjamin Vernot. A hybrid micro-macroevolutionary approach to gene tree reconstruction. In Satoru Miyano, Jill Mesirov, Simon Kasif, Sorin Istrail, Pavel Pevzner, and Michael Waterman, editors, *Research in Computational Molecular Biology*, volume 3500 of *Lecture Notes in Computer Science*, pages 993–993. Springer Berlin / Heidelberg, 2005.

[5] R. Guerra, D.B. Allison, and D. Goldstein. *Meta-analysis and Combining Information in Genetics and Genomics*. Chapman & Hall/CRC mathematical and computational biology series. Taylor & Francis, 2009.

[6] R. Libeskind-Hadas and M.A. Charleston. On the computational complexity of the reticulate cophylogeny reconstruction problem. *Journal of Computational Biology*, 16(1):105–117, 2009.

[7] D. Merkle and M. Middendorf. Reconstruction of the cophylogenetic history of related phylogenetic trees with divergence timing information. *Theory in Biosciences*, 123(4):277–299, 2005.

[8] D. Merkle, M. Middendorf, and N. Wieseke. A parameter-adaptive dynamic programming approach for inferring cophylogenies. *BMC bioinformatics*, 11(Suppl 1):S60, 2010.

[9] P. Pevzner and R. Shamir. *Bioinformatics for Biologists.* Cambridge University Press, 2011.

[10] F. Ronquist. Three-dimensional cost-matrix optimization and maximum cospeciation. *Cladistics*, 14(2):167–172, 1998.

[11] A. Yodpinyanee, B. Cousins, J. Peebles, T. Schramm, and R. Libeskind-Hadas. HMC CS Technical Report CS-2011-1: Faster dynamic programming algorithms for the cophylogeny reconstruction problem. Technical report, Harvey Mudd College, 2011.