Comenius Univerisity in Bratislava
Faculty of Mathematics, Physics and
Informatics

Algorithms for Gene Tree Reconciliation

Diploma thesis

2015
Bc. Michal Sabo

Comenius University in Bratislava
Faculty of mathematics, physics and informatics

Algorithms for Gene Tree Reconciliation

Diploma thesis

Study program: Computer Science
Study field: 2508 Computer Science, Informatics
Department: Department of Computer Science
Supervisor: doc. Mgr. Bronislava Brejová, PhD.

Bratislava, 2015

Bc. Michal Sabo

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname:	Bc. Michal Sabo
Study programme:	Computer Science (Single degree study, master II. deg., full time form)
Field of Study:	9.2.1. Computer Science, Informatics
Type of Thesis:	Diploma Thesis
Language of Thesis:	English
Secondary language:	Slovak

Title: Algorithms for Gene Tree Reconciliation

Aim: A phylogenetic tree is a standard representation of evolutionary history of a set of species. A phylogenetic tree can be also built for a set of genes, and it may differ from the species tree due to gene duplications and other phenomena. Reconciliation of a gene tree with a species tree creates a mapping between the two trees and helps to assign evolutionary events to individual nodes of the gene tree. The goal of the thesis is to provide an overview of several existing reconciliation algorithms with the emphasis on the algorithm by Ma et al (PNAS 2008). The description of the algorithm given in this paper appears to contain several mistakes, and thus the goal of the thesis is to provide a corrected version of the algorithm and the proof of its correctness.

Supervisor:	doc. Mgr. Bronislava Brejová, PhD.
Department:	FMFI.KI - Department of Computer Science
Head of department:	doc. RNDr. Daniel Olejár, PhD.
Assigned:	16.10.2014
Approved:	12.12.2014

prof. RNDr. Branislav Rovan, PhD.
Guarantor of Study Programme

.....
Student

.....
Supervisor

19685993

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Michal Sabo
Študijný program: informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: 9.2.1. informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: anglický
Sekundárny jazyk: slovenský

Názov: Algorithms for Gene Tree Reconciliation
Algoritmy pre rekonciliáciu génových stromov

Cieľ: Fylogenetický strom je štandardnou reprezentáciou evolučnej histórie pre sadu biologických druhov. Fylogenetický strom môže byť zostavený aj pre sadu génov a výsledok sa od druhového stromu môže odlišovať kvôli génovým duplikáciám alebo iným javom. Rekonciliácia génového stromu so stromom druhov vytvorí zobrazenie medzi časťami týchto stromov a môže pomôcť priradiť evolučné udalosti jednotlivým vrcholom génového stromu. Cieľom práce je poskytnúť prehľad existujúcich rekonciliačných algoritmov s dôrazom na algoritmus od Ma a kol. (PNAS 2008). Popis algoritmu v tomto článku obsahuje niekoľko chýb a teda cieľom práce je poskytnúť opravenú verziu algoritmu a dôkaz jeho správnosti.

Vedúci: doc. Mgr. Bronislava Brejová, PhD.
Katedra: FMFI.KI - Katedra informatiky
Vedúci katedry: doc. RNDr. Daniel Olejár, PhD.

Dátum zadania: 16.10.2014

Dátum schválenia: 12.12.2014 prof. RNDr. Branislav Rovan, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Acknowledgement

I would like to gratefully and sincerely thank to my supervisor doc. Mgr. Bronislava Brejová, PhD. for her help, valuable advices, patience and time during writing this thesis.

Abstract

Phylogenetics is a scientific discipline that focuses on the evolution of organisms. Evolutionary histories can be represented as phylogenetic trees. A typical kind of phylogenetic tree, called species trees, represent the history of set of species. However, there is also another type of phylogenetic trees, called gene trees, where the subject of interest is evolution of a specific gene within a set of species.

Because species are formed by their genes, gene trees and species trees are closely related. Combing a gene tree with the corresponding species tree in a process called gene tree reconciliation, allows us to determine the functions of internal nodes in the gene tree.

In our work we describe often used approaches of gene tree reconciliation, and we especially focus on a gene tree reconciliation algorithm developed by Ma et al. (PNAS 2008), where we found some inconsistencies and even some simple inputs, on which their algorithm does not work. Therefore we propose a new algorithm that solves the same problem, with its proof of correctness.

Keywords: reconciliation, phylogenetics, gene tree, species tree, speciation, duplication

Abstrakt

Fylogenetika je vedecká disciplína, ktorá sa zaoberá evolúciou organizmov. História evolúcie môže byť reprezentovaná ako fylogenetický strom. Typický druh fylogenetického stromu volajúci sa druhový strom, reprezentuje históriu danej množiny druhov. Avšak existuje taktiež aj iný druh fylogenetického stromu volajúci sa génový strom, kde predmetom skúmania je špecifický gén z danej množiny druhov.

Pretože druhy sú formované pomocou ich génov, génové stromy a druhové stromy sú si veľmi blízke. Prepojenie génového stromu spolu s prislúchajúcim druhovým stromom je proces, ktorý sa volá génová rekonciliácia a umožňuje nám priradiť funkcie vnútorným vrcholom génového stromu.

V našej práci opisujeme často používané metódy génovej rekonciliáciie a špeciálne sa zameriame na algoritmus génovej rekonciliácie, ktorý bol vyvinutý Ma a kol. (PNAS 2008), kde sme objavili isté nezrovnalosti a aj jednoduché vstupy, na ktorých algoritmus nefunguje. Preto sme navrhli nový algoritmus, ktorý rieši rovnaký problém s dôkazom jeho korektnosti.

Kľúčové slová: rekonciliácia, fylogenetika, génový strom, druhový strom, speciácia, duplikácia

Contents

Chapter 1

Introduction

In bioinformatics, evolutionary histories are typically represented as phylogenetic trees. In this thesis, we will discuss two kinds of these trees: species trees, which represent a history of a set off species and gene trees which represent a history of a specific gene. In particular, we will study the gene tree reconciliation problem, where the goal is to map vertices of a gene trees to points on the species tree, which correspond to the same points in the evolutionary history.

In the past, many researches designed and implemented reconciliation algorithms for different version of the problem. Moreover, over the time, they improved their time and space complexity or simplified them. In general, these algorithms can be divided into two groups. Depending on whether the input gene and species trees are rooted or not. Another division could be based on whether the input trees have weighted edges, where the weights correspond to evolutionary distances. We will provide a brief survey of various reconciliation approaches in Chapter 3.

In 2008 a group of researches leaded by David Haussler presented a polynomial-time algorithm for recovering the evolutionary history of a set of genomes under the infinite sites model [ha08]. One part of their complex algorithm is a gene tree reconciliation algorithm for trees with weighted edges. However after studying details of their work, we have found some inconsistencies or even inputs where the algorithm does not work correctly. In this thesis, particularly in Chapter 4, we describe these problems and provide a new algorithm and prove its correctness.

Chapter 2

Background and terminology

In this section we introduce the necessary background information and terminology and describe the reconciliation problem in more details.

2.1 Background

Phylogenetics studies the relationship between organisms from the perspective of their evolution. Because we suppose that all organisms are evolved from a common ancestor, we can use appropriate graph structures to presents their evolution. Specifically, the appropriate structure is usually a tree, and in biology and bioinformatics area it is called a phylogenetic tree. A phylogenetic tree can describe not only a topology, but by adding the lengths to the edges, we can describe also the palaeontological time scale of evolution.

One type of a phylogenetic tree is a species tree, and it depicts how species are related to each other. It is sometimes also called a 'Tree of life'. In a species tree, the leaves represent present-day species, and the internal nodes represent speciation events. A speciation is an event in the evolution, where two subpopulations of a species diverge apart and in the course of evolution they form two individual species.

In order to continue, we provide some details about molecular basis of evolution. Each organism contains genetic material encoded in a genome. A genome consists of DNA (deoxyribonucleic acid). DNA is composed of four bases: adenine, thymine, guanine and cytosine, and the order of these bases encodes the genetic information. Usually a DNA is in the form of a two strands DNA also called a double helix. It is a structure, where each DNA strand is connected

with its complementary strand based of the pairing rules, where adenine base is paired with thymine base and guanine base is paired with cytosine base. DNA is packed into structures called chromosomes stored in a cell structure called a nucleus. Parts of DNA encoding the proteins are called genes.

When a cell divides, each daughter cell receives a complete copy of the parental genome created by DNA replication. During this process, two complementary DNA strands are unwounded at some locations and then new complementary strands are added to each original strand. However during this process some bases of DNA can be connected incorrectly, or some other errors can be introduced. This causes mutations in DNA, and as mutations accumulate, new species evolve.

Sometimes during DNA replication, a gene may be duplicated, which means, that instead of one copy of the gene, there are two copies of the same gene in a daughter cell. When this happens, the copies continue to evolve individually. In the contrast to gene duplication, during the DNA replication some genes can disappear from the daughter cell. This event is called a gene loss.

Each species has its own set of genes which differ from genes in different species. However, a gene from one species is related to a gene from another species, if they have been evolved from the same gene in the genome of some common ancestor, and then we call those genes homologues. Homologues are divided into two groups: orthologues and paralogues. Orthologues are genes that diverged by a speciation event. In contrast, paralogues are genes that diverged by a duplication event. A gene can have a paralougue even in the same genome.

The view of species as a set of genes allows us to create a different phylogenetic tree called a gene tree. A gene tree represents the evolution of one gene and its homologous within a set of species. In a gene tree, the leaves represent present-day genes and internal nodes represent speciation or duplication events.

Reconciliation connects a gene tree and a species tree together via a mapping of vertices from a gene tree onto a species tree. The mapping can be from vertices to vertices (preserving speciation) or from vertices to some points of the branches (creating duplication vertices in the reconciled tree) or even to a newly added root of the species tree. The result, a reconciled tree, gives us the information which vertices in the gene trees are duplications and which speciations, and therefore it helps with closer studying of the gene evolution.

Figure 2.1 is an example how an reconciliation helps to distinguish between

a speciation and a duplication event in a gene tree. Figure 2.1 a) is a rooted gene tree with the root r_B and with one internal node x. Without any additional information, one does not know which events both internal nodes represent. One possible evolutionary history would be that both, r_B and x, represent speciation events. Then Figure 2.1 b) would represent a species tree for this evolutionary history. However, if we allow gene losses, then we can form another evolutionary history for the same gene tree. On Figure 2.1 c) we added gene losses that are marked by a dashed lines, and that were lost during evolution. If r_B is a duplication event and x is a speciation event, then Figure 2.1 d) would represent the species tree for this evolutionary history.

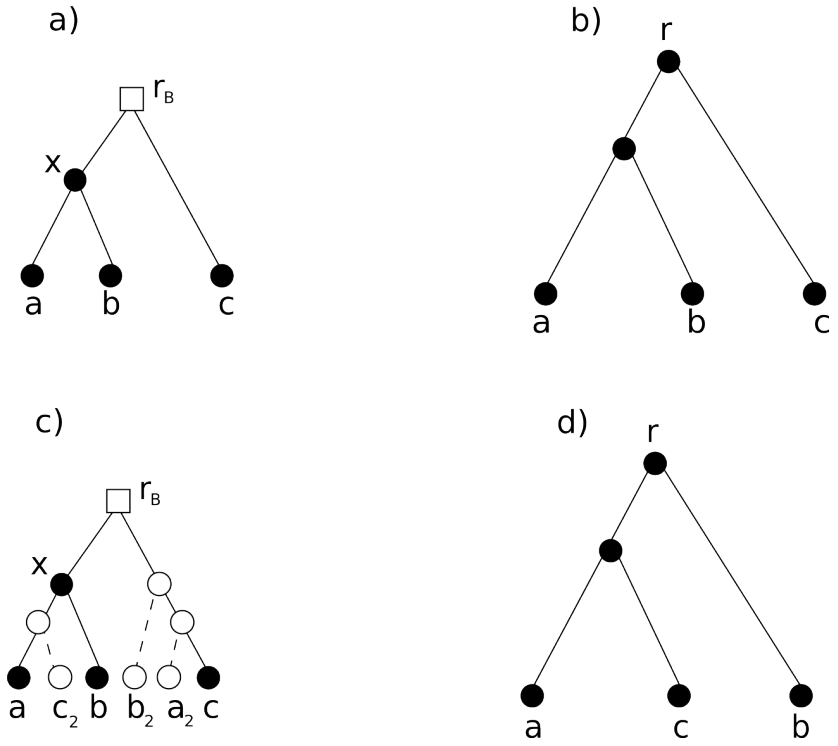

Figure 2.1: Example how an reconciliation helps to distinguish between a speciation and a duplication event in a gene tree a) The input gene tree B b) The species tree for the hypothesis, that both r_B and x in the gene tree are speciations c) The same input gene tree extended by gene loss vertices marked by dashed lines d) The species tree for the hypothesis, that r_B is a duplication event and x in a speciation event

The motivation of gene trees reconciliation is not only in revival of evolutionary histories, but it can be various. One of it is an automation of analysing of

protein sequences, where the there is a need to studying orthologues sequences rather than paralougues sequences [ed01].

2.2 Terminology

In this section, we define basic terms that are widely used in our work.

Definition 1. *Tree is a connected graph without any cycles.*

Definition 2. *Rooted tree is a tree with specification of one node - root. It determines the orientation of the tree.*

Definition 3. *Unrooted tree is a tree without root.*

Definition 4. *Mapping ϕ between two trees B and T is a linear transformation $\phi : B \to T$ in which each node $X_i \in B$ is mapped to the point $Y_i \in T$. Notation $Y_i = \phi(X_i)$ means that the node X_i from B is mapped to the point Y_i from T in mapping ϕ.*

Chapter 3

Different approached to reconciliation

In this chapter, we describe different approaches to tree reconciliation. The problem of gene tree reconciliation is well studied, and various reviews articles have been already written [be11]. However all of them assume that a reader has a certain level of the knowledge about molecular biology. Our aim here is to provide an overview readable also for readers without this knowledge.

Gene tree reconciliation requires a gene tree and a species tree on input. In Section 3.1 we briefly describe how these trees are built and then the remainder of the chapter describes known reconciliation algorithms categorized according to the problem they solve.

3.1 Phylogenetic tree reconstruction

In order to understand differences between individual reconciliation methods, one has to firstly understand the evolution model that a specific method works with. The main problem is, that we can only assume how the real evolutionary history looks like, because we know exactly only present-day species and present-day genes. Nonetheless, we can create an evolutionary model with some set of allowed evolutionary operations (evolutionary events). For example in Chapter 2 we introduced three basic evolution events: speciation, gene duplication and gene loss. However besides them, there are more types of evolutionary events for example: gene conversion, horizontal gene transfer, hybridization, linage sorting [sa01]. For example, horizontal gene transfer is a evolution process where a gene

is copied from one genome to a genome of unrelated species, i.e. it was not acquired in the usual way from the parental cells.

Gene tree reconciliation needs to take into account the way how gene and species trees were built. A gene tree is build from DNA sequences of its genes and similarly, a species tree is build from sequences of selected genes from the considered species. Therefore, we need tools, that transform output sequences into a phylogenetic tree. [mi98] provides a useful overview of various methods used for building trees. In general there are three approaches: distance methods, parsimony and probabilistic methods.

The principle of the distance methods is that distances are estimated between all pairs of considered sequences. There are various algorithms for estimating a tree from a distance matrix. Two most famous are a clustering method called UPGMA and the neighbour-joining method. UPGMA method produces a rooted output tree whereas in order to root output tree from the neighbour-joining method, an additional step is needed. This additional step most often consists of adding so called outgroup into the input set of sequences and thus also to the output tree. Outgroup is a species, that is known to belong to a different branch of evolution than all other species in the set.

The second approach is called parsimony, which a well known principle. Most reconciliation algorithms use the parsimony principle as well, as we will see in this chapter. After applying it to building trees, the resulting tree is a the one that requires the smallest number of evolutionary events. One can imagine it as a construction of all possible trees from the set of sequences (where the leaves are sequences and the internal vertices are ancestral sequences in the way that they minimize number of changes) and as the solution is selected the one with the fewest changes. Although this is a NP-hard problem, there are effective heuristics for finding the minimum tree.

The last approach is a probability evolution model. In this model, evolutionary events have added probability that they occur. Two often used method were developed: a maximum likelihood method and Bayesian method.

3.2 Gene tree reconciliation with duplication and loss

The classical approach to gene tree reconciliation uses the parsimony evolution model that minimizes the number of duplication and gene losses. It works with a rooted gene tree and a rooted species tree, which are given on the input, and as the result of the reconciliation we except a reconciled tree that minimizes the number of duplications and gene looses. Although linear-time algorithms are known for this problem [zh97], [ed01] focused on simplifying the algorithm. The algorithm recursively traverses all internal nodes in a gene tree in post-order (because this gene tree is rooted, it is possible) and it checks whether each currently processed internal node represents a duplication event or a speciation event. They also implemented an already known algorithm with the time complexity $O(n)$ and run both of them at the same input. Even if their algorithm has complexity $O(n^2)$, it reached better empirical results than the one with the time complexity $O(n)$. The facts, that the second algorithm is much more complex than their algorithm, and that their algorithm achieves the worst time running only for unbalanced species trees, were given as reasons for this. Because the optimal algorithm that runs in linear time has been found and proved, some researchers focused on analysis of the influence of the quality of data that are given on the input. For example [ha07] emphasized the fact that in reconciliation algorithms there is a need to have correct data, otherwise the result can be biased. Following this, [ma12] developed a polynomial-time heuristic, that removes vertices from the gene tree that can be incorrect.

3.3 Unrooted gene tree reconciliation algorithms

As we mentioned above, building gene trees from sequences does not detect the root directly, instead it needs to be added using outgroup. Although, so far mentioned reconciliation approaches required rooted gene trees on input, indeed, a gene tree can be rooted during the process of reconciliation in a way that this root minimizes the number of given evolutionary operations. [ti13] is an example of a such approach, and moreover they provide an unified framework for reconciliation where a set of unrooted binary gene trees and a rooted binary species tree are given as input. By unified, authors mean, that algorithm gets on

the input also the information in which evolution operations are allowed. The algorithm works in linear time.

3.4 Non-binary gene trees reconciliation algorithms

Next category of gene trees reconciliation algorithms works with non-binary gene trees. [zh14] pointed out some situations, where the reconciliation of non-binary gene trees are needed, for example some currently used phylogenetic programs produce non-binary gene trees as an output. Their another motivation was, that non-binary gene trees allow to construct better heuristic programs for specific reconciliation, because some programs suppose that gene trees are non-binary. They consider the parsimony model with the minimum number of duplication and loss events and they design a linear time algorithm for reconciliation. Before them, within the same evolutionary model [mab12] found an algorithm with quadratic time algorithm.

3.5 Non-binary unrooted gene tree reconciliation where species tree is non-binary and rooted

In this category we assume to reconcile an arbitrary gene tree (thus not necessary rooted and binary) with a corresponding species tree that does not have to be binary, but has to be rooted. [eu11] studied the model and they presented a polynomial-time algorithm that found reconciliation and roots gene trees.

3.6 Arbitrary gene and species trees reconciliation

Previous category naturally required an extension to an unrooted species tree. Thus in this category, a gene tree and a species tree can be arbitrary trees (not necessary rooted and not necessary binary). The motivation for using non-binary gene trees is the same as in the previous category. The motivation for using non-binary species tree is that not all species diverging order is known [zh12] and therefore also species trees can be non-binary. [zh12] studied this category and they proved that this problem is $NP - hard$ even for a parsimony model that

reduced duplication cost, by a reduction to a problem of constructing a species tree from a set of gene trees. However, they proposed a heuristic that solves it. The heuristic firstly processed non-binary species tree in the maximum cubic time, but then it reconciles a gene tree in linear time.

Chapter 4

Isometric reconciliation

This chapter is devoted to reconciliation algorithms that get an unrooted gene tree with exact branch lengths and a species tree also with exact branch lengths on the input. The goal is to find a reconciliation that preserves those distances. We start by defining this problem, which was first studied by Ma at al [ha08] in 2008, more formally. Then we describe their original algorithm. However, we show that it does not always work correctly. Finally, we introduce our algorithm for solving the same problem with the proof of its correctness and the analysis of the time complexity. In order to avoid any confusion, please note that Ma at al. [ha08] use the term 'atom trees' to refer to 'gene trees'. The authors decided to use this terminology because it better corresponds to the infinite sites model they introduced. Nevertheless, we use in our work the traditional name 'gene tree'. Also we often refer to the distance between two vertices in a gene tree or in a species tree. Ma at al. [ha08] used the upper-case letter D for distances in a gene tree and the lower-case letter d for distances in species tree. However we use only the upper-case letter, and thus the symbol $D(u, v)$ represents the distance between the vertices u and v in their respective tree. By T^* we mark the result of reconciliation - a reconciled tree.

4.1 Problem definition

Although we devoted the whole chapter to different approaches of reconciliation, we have not explained the specific reconciliation problem that was introduced by Ma at al.[ha08]. They do reconciliation of a rooted species tree and a set of unrooted gene trees where the exact branch lengths of all trees are known. This

kind of reconciliation had not been studied before, and therefore there was a need to name it differently - they decided for the name 'isometric reconciliation'. Informally, an isometric reconciliation is a mapping from a gene tree B to a species tree T that preserves evolution distances (respectively branch lengths) and roots the gene tree. They formally defined it as Definition 5.

Definition 5. *[ha08] Any mapping ϕ from an atom tree B to a species tree T that roots the atom tree is an isometric reconciliation if*

1. *Every leaf of B maps to the leaf of the designated species in T.*
2. *Each internal node of B maps to a speciation node in T or a point on a branch in T.*
3. *The new root r of B maps to a point $\phi(r)$ on a branch in T such that any other node x in B maps to $\phi(x)$ below $\phi(r)$ and $D(\phi(x), \phi(r)) = D(x, r)$.*

However this definition is not strong enough, because it does not enforce preservation of evolution distances between all relevant pairs of nodes, and thus allows to form a reconciliation that does not correspond to any evolution history. Example of this is in Figure 4.1. Although all conditions from Definition 5 are satisfied, this gene tree (Figure 4.1,a)) cannot occur in the evolution history of the species tree (Figure 4.1,b)). One of the reasons why it could not happen is that in the reconciled tree T^* (Figure 4.1, d) we can see that the distance between the leaf node g and the internal node $\phi(x)$ equals 1, however in the gene tree B (Figure 4.1, a) the distance between the leaf g_2 and the node x equals 3. Another problem is that leaves f_1 and g_1 are descendants of node x in the rooted gene tree (Figure 4.1, c)). Therefore also species f, g in the reconciled tree should have been in the sub-tree of $\phi(x)$, but as we can see the species f is not there ($Figure\ 4.1, d$)). Moreover Definition 5 does not allow to map the newly found root r_B to a point $\phi(r_B)$ in T that is above the root of T, which is necessary when the oldest duplication precedes the oldest speciation as in Figure 4.2.

Therefore we modified the original definition of the isometric reconciliation by addressing these problems. Our definition is Definition 6. We added the fourth condition that allows to map vertices only to the correct sub-trees and we modified the third condition to allow mapping of the newly found root r_B of B

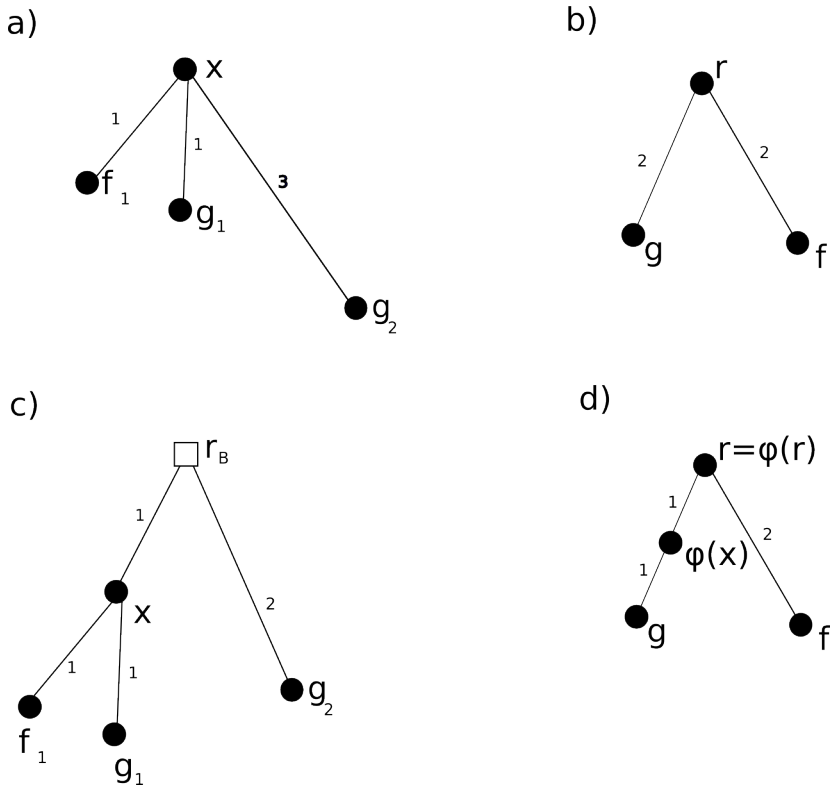

Figure 4.1: Example of an input that satisfies Definition 5 but it does not correspond to any evolution history a) The input unrooted gene tree B b) The input rooted species tree T c) The output rooted gene tree B_r d) The output reconciled tree T^*

to a point $\phi(r_B)$ that is above the root of T. Moreover, we forbid to put the new root r_B of B into an existing vertex of B. Otherwise we would obtain a rooted gene tree in which the root has three children and thus it would not be binary.

Definition 6. *Any mapping ϕ from a gene tree B to a species tree T that roots the atom tree is an isometric reconciliation if*

1. *Every leaf of B maps to the leaf of the designated species in T*

2. *Each internal node of B maps to a speciation node in T or to a point on a branch in T*

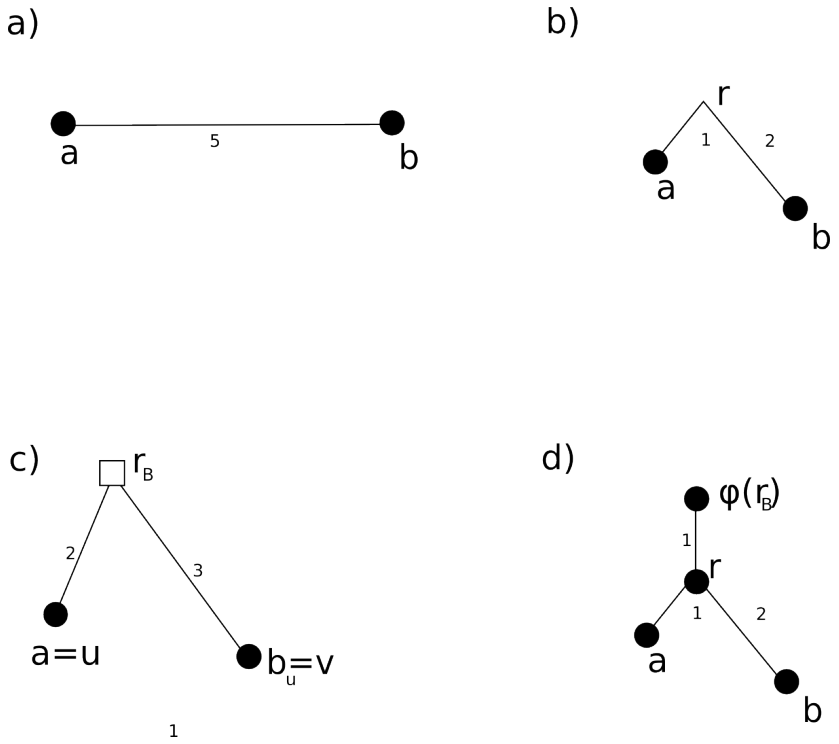

Figure 4.2: Example of an input which Algorithm 4.2 does not process a) The input unrooted gene tree B b) The input rooted species tree T c) The correct output rooted tree B_r d) The correct output reconciled tree T^*; the new vertex $\phi(r_B)$ represents a new root of T^*

3. *The found root r_B of B lies between two vertices in B and it is mapped to a speciation node in T, or to a point on a branch in T or to a point above the root of T on a newly added branch in T that is connected with the root of T*

4. *For all nodes u, v in the rooted gene tree B: if v is a descendant of u, then $\phi(v)$ is a descendant of $\phi(u)$ in T and $D(u, v) = D(\phi(u), \phi(v))$*

4.2 The original reconciliation algorithm

In this section we describe the original algorithm for isometric reconciliation [ha08]. It solves this problem: For a given set of unrooted genes trees $T_1, ..., T_n$ and a rooted species tree T construct an isometric reconciliation or detect that such an isometric reconciliation does not exists, if all evolution distances are given and are exact. The solution of this was stated in theorem Theorem 1 in the

supplementary material [ha08] and was given in the form of a polynomial-time algorithm and a proof of its correctness.

Theorem 1. *[ha08] Given a set $T_1, ..., T_N$ of unrooted atom trees with designated leaf species and a species tree T, the isometric reconciliation of $T_1, ..., T_N$ with T is unique and there is an efficient procedure to either construct it or detect that no such isometric reconciliation exists.*

Algorithm 1 and Algorithm 4.2 below represent the original algorithm [ha08] with slight modifications. The original version uses the same variables for different purposes, which can be confusing; therefore we use indexes for those duplicate variables (for example d and d_n, λ and λ_n). We also use the abbreviation LCA to stand for 'last common ancestor' and we use only the upper-case D for denoting the distance between two vertices. The rest of algorithm stays unchanged. Algorithm 1 processes all gene trees from the input sequentially. For each gene tree B, it first maps all its leaf nodes to a correct place on the species tree T. Then it iteratively takes an unmapped node x that is connected with at least 2 nodes u and v that have been already processed and calls Algorithm 4.2. The task of Algorithm 4.2 is to map x onto species tree T and to root the processed gene tree B. If x is connected also with the third node z, that has been already processed as well, it means that x is the last unmapped node in B.

In Algorithm 4.2, node x represents an unmapped internal node of the atom tree B with branches to nodes u, v and z. We suppose that nodes u and v have been already processed and mapped to $\phi(u)$ and $\phi(v)$ in the species tree T.

Algorithm 4.2, MapAtomTreeNode(x):

(0) Let $d_1 = D(x, u)$, $d_2 = D(x, v)$, $\lambda = LCA(\phi(u), \phi(v))$ in T (if $\phi(u) == \phi(v)$ then $\lambda = \phi(u) = \phi(v)$), $d'_1 = D(\phi(u), \lambda)$, $d'_2 = D(\phi(v), \lambda)$.

(1) If $d_1 + d_2 < d'_1 + d'_2$, reject and exit.

(2) Let $\epsilon = d_1 + d_2 - d'_1 - d'_2$.

(3) If $d_1 == d'_1 + \epsilon/2$ (and $d_2 == d'_2 + \epsilon/2$) then map x to a point $\phi(x)$ at distance $\epsilon/2$ above λ in the species tree T.

(4) Else if $\epsilon == 0$ then map x to a point $\phi(x)$ at distance d_1 from $\phi(u)$ and d_2 from $\phi(v)$ on the path connecting $\phi(u)$ and $\phi(v)$ in the species tree T.

Algorithm 1 ReconcileTrees($T, \mathbf{T}_{i,i=1,2,..,N}$)

```
1: for all gene trees T_i such that 1 ≤ i ≤ N do
2:   M ⇐ ∅
3:   for all leaf nodes k in T_i do
4:     φ(k) ⇐ g, where g is the genome to which k belongs.
5:     add k to M and set t_k = {k} (mapped subtree for k ∈ M).
6:   end for
7:   while M has more than two tree nodes do
8:     if u, v ∈ M and u, v and z are connected to an unmapped node x in T_i
       then
9:       MapAtomTreeNode(x)
10:      remove u, v and (if necessary) z from M
11:      if M is empty then
12:        set M to the set consisting of just the root node r and set t_r = B
13:      else
14:        add x to M and set t_x = t_u ∪ t_v ∪ {x}
15:      end if
16:    end if
17:  end while
18: end for
```

(5) Else (i.e. if $\epsilon > 0$ and $d_1 \neq d'_1 + \epsilon/2$) if the atom tree B is already rooted then reject and exit.

(6) Else

   (a) Place the root r of B at distance $d'_1 + \epsilon/2$ from u and $d'_2 + \epsilon/2$ from v on the path that connects u and v in B and map r to a point $\phi(r)$ at distance $\epsilon/2$ above λ

   (b) If $d_1 < d'_1 + \epsilon/2$ map x to $\phi(x)$ at distance d_1 from $\phi(u)$, else (i.e. if $d_2 < d'_2 + \epsilon/2$) map x to the point $\phi(x)$ at distance d_2 above $\phi(v)$

(7) If z has been already mapped to the node $\phi(z)$ in T, then

   (a) If the tree is already rooted, reject and exit unless $\phi(z)$ lies below $\phi(x)$ in T or vice-versa and $D(\phi(x), \phi(z)) = D(x, z)$.

   (b) Else

      i. Let $d_n = D(x, z)$, $\lambda_n = LCA(\phi(x), \phi(z))$ in T, $d'_{1n} = D(\phi(x), \lambda_n)$, $d'_{2n} = D(\phi(z), \lambda_n)$, $\epsilon_n = d_n - d'_{1n} - d'_{2n}$

ii. If $\epsilon_n < 0$, reject and exit.

iii. If $\epsilon_n = 0$ and $\phi(x) = \lambda_n$, reject and exit.

iv. Place the root r of B at distance $d'_{1n} + \epsilon/2$ from x and $d'_{2n} + \epsilon_n/2$ from z on the path that connects x and z in B, and map r to $\phi(r)$ at distance $\epsilon_n/2$ above λ_n

4.2.1 A counter-example for the original reconciliation algorithm

Although the task of reconciliation with the given exact branch lengths does not sound complicated, there are some trivial inputs on which the algorithm returns wrong answers. Figure 4.3 is an example of such an input.

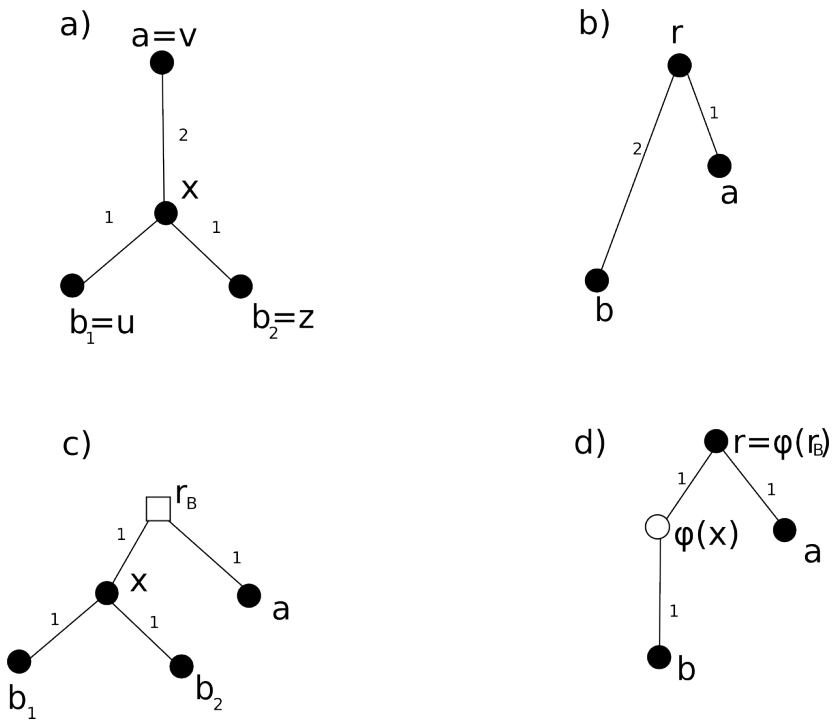

Figure 4.3: Example of an input on which Algorithm 4.2 does not work correctly a) The input unrooted gene tree B b) The input rooted species tree T c) The correct output rooted tree B_r d) The correct output reconciled tree T^*; the new vertex $\phi(x)$ represents a duplication

If we run Algorithm 1 on this input, it firstly maps the leaves and then calls Algorithm 4.2 to map the last unmapped vertex x. Let us assume that node a

has label v, b_1 has label u and b_2 has label z. The following simulation shows working of Algorithm 4.2 in this situation.

(0) Let $d_1 = D(x, u) = 1$
$d_2 = D(x, v) = 2$
$\lambda = LCA(\phi(u), \phi(v)) = r$ in T
$d'_1 = D(\phi(u), \lambda) = 2$
$d'_2 = D(\phi(v), \lambda) = 1$ - some initializations

(1) If $3 < 3$, reject and exit - this step is skipped

(2) $\epsilon = 0$.

(3) If $1 == 2$ (and $2 == 1$) then map x to a point $\phi(x)$ at distance $\epsilon/2$ above λ in the species tree T this step is skipped

(4) Else if $0 == 0$ then map x to a point $\phi(x)$ at distance 1 from $\phi(u)$ and 2 from $\phi(v)$ on the path connecting $\phi(u)$ and $\phi(v)$ in the species tree T this step happens and x is correctly mapped

(5) Else ... this step is skipped

(6) Else ... this step is skipped

(7) If z has been already mapped to the node $\phi(z)$ in T, then

(a) If the tree is already rooted, reject and exit unless $\phi(z)$ lies below $\phi(x)$ in T or vice-versa and $d(\phi(x), \phi(z)) = D(x, z)$ - this step is skipped

(b) Else

i. $d_n = D(x, z) = 1$
$\lambda_n = LCA(\phi(x), \phi(z)) = \phi(x)$ in T
$d'_{1n} = d(\phi(x), \lambda_n) = 0$
$d'_{2n} = d(\phi(z), \lambda_n) = 1$
$\epsilon_n = d_n - d'_{1n} - d'_{2n} = 0$ - another initializations

ii. If $0 < 0$, reject and exit - this step is skipped

iii. If $0 = 0$ and $\phi(x) = \lambda_n$, reject and exit - the algorithm rejects the input at this step

Although there exists an reconciliation for this input, the Algorithm 4.2 rejected it. We can notice that the algorithm failed to root the tree of the branch between x and v. Another example is in Figure 4.2, which depicts a gene tree B with only two leaves. Algorithm 1 maps them correctly, but the Algorithm 4.2 is never called, and therefore B stayed unrooted, even if the root exists.

These two examples show that although Algorithm 4.2 is introduced with the proof of the correctness in [ha08], it does not always work correctly. After closer examination, we found out that the finding the root in gene trees causes the problems. In Algorithm 4.2 a gene tree can be rooted only at two places - $(6)(a)$ and $(7)(b)(iv)$. However these places do not cover all possibilities that can occur, as our example demonstrates.

4.3 Our reconciliation algorithm

Based on the observation above, we have modified Algorithm 4.2 to work correctly according to our modified stronger definition of the isometric reconciliation Definition 6. Our reconciliation algorithm consists of three steps:

- Mapping vertices from the gene tree into the species tree
- Rooting the gene tree
- Checking if the output represents a correct isometric reconciliation

Note that we have decoupled the mapping and rooting steps so that the algorithm need to consider fewer special cases.

4.3.1 Modified algorithm

Algorithm 2 represents the modified algorithm Algorithm 1, and Algorithm 3 represents the modified algorithm Algorithm 4.2. The Algorithm 2 calls three other algorithms Algorithm 3, Algorithm 4 and Algorithm 5 during processing of one gene tree B. Algorithm 3 is called for each unmapped vertex x, and its task is to map it correctly to the species tree. Once all vertices are mapped, Algorithm 4 finds the root of B and maps it to the correct point in the species trees. Finally, Algorithm 5 checks if the output fulfils the definition of an isometric reconciliation.

Algorithm 2 ReconcileTrees($T, \mathbf{T}_{i,i=1,2,..,N}$)

for all gene trees T_i such that $1 \leq i \leq N$ **do**
  $M \Leftarrow \emptyset$
  for all leaf nodes k in T_i **do**
    map k to a leaf $\phi(k)$, such that $\phi(k)$ is the genome to which k belongs
    add k to M
  end for
  while M has at least three tree nodes **do**
    if $u, v \in M$ and u, v are connected to an unmapped node x in T_i **then**
      MapAtomTreeNode(x,u,v)
      add x to M
      remove u, v from M
    end if
  end while
  RootGeneTree(B)
  CheckTheResults(T^*, T_i)
end for

Algorithm 3 MapGeneTreeNode(x, u, v)

Require: mapping of $\phi(u)$ and $\phi(v)$ have already been determined
$d_1 = D(x, u)$
$d_2 = D(x, v)$
$\lambda = LCA(\phi(u), \phi(v))$ in T
$d'_1 = d(\phi(u), \lambda)$
$d'_2 = d(\phi(v), \lambda)$
$\epsilon = d_1 + d_2 - d'_1 - d'_2$
if $d'_2 \geq d_2$ **then**
  map x to $\phi(x)$ at distance d_2 above $\phi(v)$
else
  map x to $\phi(x)$ at distance d_1 above $\phi(u)$
end if

Algorithm 4 RootGeneTree(B)

for all edges(u,v) **do**
  $d = D(u, v)$
  $\lambda = LCA(\phi(u), \phi(v))$
  $d'_1 = D(\phi(u), \lambda)$
  $d'_2 = D(\phi(v), \lambda)$
  $\epsilon = d - d'_1 - d'_2$
  if $\epsilon < 0$ or $d'_1 + \epsilon/2 > d$ **then**
    reject and exit
  else if $d > d'_1 + d'_2$ or $(\lambda \neq \phi(u)$ and $(\lambda \neq \phi(v))$ **then**
    place the root of B to a point r_B at distance $d'_1 + \epsilon/2$ from u at the edge(u,v) and map it to a point $\phi(r_B)$ at distance $\epsilon/2$ above λ
  end if
end for
if zero or multiple roots found or if root is placed to a node **then**
  reject and exit
end if

Algorithm 5 CheckTheResults(T^*, B^*)

for all edges(u,v) $\in B$ **do**
  check if the edge(u,v) satisfies the condition of isometric reconciliation
  if $(D(u, v) \neq D(\phi(u), \phi(v))$ **then**
    reject and exit
  end if
  $\gamma = LCA(u, v)$
  $\lambda = LCA(\phi(u), \phi(v))$
  if $u == \gamma$ **then**
    if $\phi(u) \neq \lambda$ **then**
      reject and exit
    end if
  else if $v == \gamma$ **then**
    if $\phi(v) \neq \lambda$ **then**
      reject and exit
    end if
  end if
end for

4.3.2 Algorithm proof

For the sake of simplicity, we consider that on the input we have only one unrooted gene tree B and a species tree T (with exact branch lengths). In case of more gene trees on the input, Algorithm 2 processes them sequentially one by one, and therefore this simplification does not influence the correctness of the proof. In order to show that our modified algorithm works correctly, we need to prove two things. Firstly, if an isometric reconciliation exists for this input, then the algorithm finds it. Secondly, if the algorithm finishes without rejection and roots the gene tree, the result fulfils the definition of isometric reconciliation.

Claim 1. *If an isometric reconciliation for the gene tree B and the species tree T exists, then the first part of Algorithm 2 and Algorithm 3 correctly map all vertices of B.*

Proof 1.

Claim 1 assumes that the isometric reconciliation between the gene tree B and the species tree T exits. Algorithm 2 starts with processing all leaves from B and mapping them into correct place on T. This step is correct, because all nodes are mapped to genomes they belong to. Also these already processed vertices are added to the set M. M represents mapped vertices that are connected to candidate vertices for the next mapping. If the initial size of M equals 1, it means that the B is a tree with one vertex and if the initial size of M equals 2, it means that the B has only two vertices, both leaves. However, in both cases there is no unmapped vertex. If the size of M is at least 3, then Algorithm 3 is called repeatedly until the number of vertices in M reaches 2. Inside this loop, the algorithm selects vertices u a v from M that are connected with an unmapped vertex x. After Algorithm 3 maps x, x is added into M and u, v are removed from M, because they are not connected with any unmapped vertex anymore. If the size of M reaches 2, it means that the remaining two vertices in M have to be connected each other and therefore there no other unmapped vertex in B and the loop terminates. Now we focus on calling Algorithm 3. We prove by induction the invariant that after i calls of Algorithm 3, all already mapped vertices have been mapped correctly and so far unmapped vertices form connected sub-graph of B. As the basis for this induction, we use the 0th step, thus $i = 0$, the algorithm mapped all its leaves correctly as we showed above. Because B is a tree structure,

after processing its leaves, all unprocessed nodes are all internal nodes of B and therefore are connected each other and form a sub-graph of B. Therefore the invariant holds for the basis. Now we want to prove that if the variant holds after $k-1$ calls of Algorithm 3, it holds also after k calls. In order to prove it, we firstly need to show that if u and v have been already mapped correctly to $\phi(u)$ and $\phi(v)$, and x is an unmapped vertex that is connected to u and v, then Algorithm 3 maps it correctly. We prove it by the consideration of all possibilities where the true root r_B is in B and then where x maps to T in the true isometric reconciliation. In the total , there are 5 possibilities, where the root r_B of B can be:

- **r_B is in already processed sub-tree of u (Figure 4.4 a), b)):**
  In such a case, $\phi(u)$ has to be an ancestor of $\phi(x)$, and $\phi(x)$ has to be an ancestor of $\phi(v)$. Therefore the $\lambda = LCA(\phi(u), \phi(v)) = \phi(u)$, and $\phi(x)$ has to be mapped somewhere on the path between $\phi(u)$ and $\phi(v)$. Because the distances has to be preserved, we know that $D(\phi(v), \phi(x))$ has to be equal to $D(v, x)$. Therefore we can certainly map $\phi(x)$ at the distance $D(v, x)$ above $\phi(v)$. Algorithm 3 will have $d_2' = D(\phi(v), \lambda) = D(\phi(v), \phi(u)) = D(u, v) = d_1 + d_2$, thus $d_2' > d_2$ and therefore it will map x correctly in line 8.

- **r_B is in already processed sub-tree of v (Figure 4.4 c), d)):**
  It is a symmetric case to the case above, $\phi(v)$ has to be an ancestor of $\phi(x)$, and $\phi(x)$ has to be an ancestor of $\phi(u)$. Therefore the $\lambda = LCA(\phi(u), \phi(v)) = \phi(v)$, and $\phi(x)$ has to be mapped somewhere on the path between $\phi(u)$ and $\phi(v)$. Because the distances has to be preserved, we know that $D(\phi(u), \phi(x))$ has to be equal to $D(u, x)$. Therefore we can certainly map $\phi(x)$ at the distance $D(u, x)$ above $\phi(u)$. Algorithm 3 will have $d_2' = D(\phi(v), \lambda) = D(\phi(v), \phi(v)) = 0$, thus $d_2' < d_2$ and therefore it will map x correctly in line 10.

- **r_B is on edge between u and x (Figure 4.5 a))**:
  It means that $\phi(x)$ is an ancestor of $\phi(v)$, and we know the position of $LCA(\phi(u), \phi(v)) = \lambda$. Then $\phi(x)$ has to be mapped at one out of two places:

  a) on the path from $\phi(v)$ to λ including endpoints (Figure 4.5 b))

b) on the path from λ to $\phi(r_b)$ including endpoints (Figure 4.5 c))

In both cases, $\phi(x)$ has to be on the path between $\phi(v)$ and $\phi(r_B)$ and also the distances have to be preserved, i.e. we know that $D(\phi(v), \phi(x))$ has to be equal to $D(v, x)$. Therefore we can again certainly map $\phi(x)$ at the distance $D(v, x)$ above $\phi(v)$. If case a) happens, Algorithm **??**alg11) will have $d_2' = D(\phi(v), \lambda) \geq D(\phi(v), \phi(x)) = D(x, v) = d_2$ and therefore, $d_2' \geq d_2$, thus it will map x correctly in line 8. If case b) happens, Algorithm 3 will have $d_2' = D(\phi(v), \lambda) < D(\phi(v), \phi(x)) = D(x, v) = d_2$ and therefore, $d_2' < d_2$ and thus it will map x correctly in line 10.

- r_B **is on edge between** v **and** x **(Figure 4.6 a))**:
  It is a symmetric case to the case above, it means that $\phi(x)$ is an ancestor of $\phi(u)$, and we know the position of $LCA(\phi(u), \phi(v)) = \lambda$. Then $\phi(x)$ has to be mapped at one out of two places:

  a) on the path from $\phi(u)$ to λ including endpoints (Figure 4.6 b))

  b) on the path from λ to $\phi(r_b)$ including endpoints (Figure 4.6 c))

  In both cases, $\phi(x)$ has to be on the path between $\phi(u)$ and $\phi(r_B)$ and also the distances have to be preserved, i.e. we know that $D(\phi(u), \phi(x))$ has to be equal to $D(u, x)$. Therefore we can again certainly map $\phi(x)$ at the distance $D(u, x)$ above $\phi(u)$. Moreover $D(\phi(v), \phi(x)) > D(\phi(v), \lambda)$, therefore in both cases, Algorithm 3 will have $d_2' = D(\phi(v), \lambda) < D(\phi(v), \phi(x)) = D(x, v) = d_2$ and therefore, $d_2' < d_2$, thus it will map x correctly in line 8.

- r_B **is in somewhere else(Figure 4.7 a), b)):**
  It means that $\phi(x)$ is an ancestor of both $\phi(u)$ and $\phi(v)$ and because the distances has to be preserved, $\phi(x)$ has to be mapped at the distance $D(x, u)$ above $\phi(u)$ and also at the distance $D(x, v)$ above $\phi(v)$. In both cases the results is the same. Because $D(\phi(v), \phi(x)) \geq D(\phi(v), \lambda)$, Algorithm 3 will have $d_2' = D(\phi(v), \lambda) < D(\phi(v), \phi(x)) = D(x, v) = d_2$ and therefore, $d_2' < d_2$, thus it will map x correctly in line 8.

Because we covered all branches where the root r_B of B can be, we can claim that $\phi(x)$ is mapped or at the distance $D(u, x)$ above $\phi(u)$ or at the distance $D(v, x)$ above $\phi(v)$.

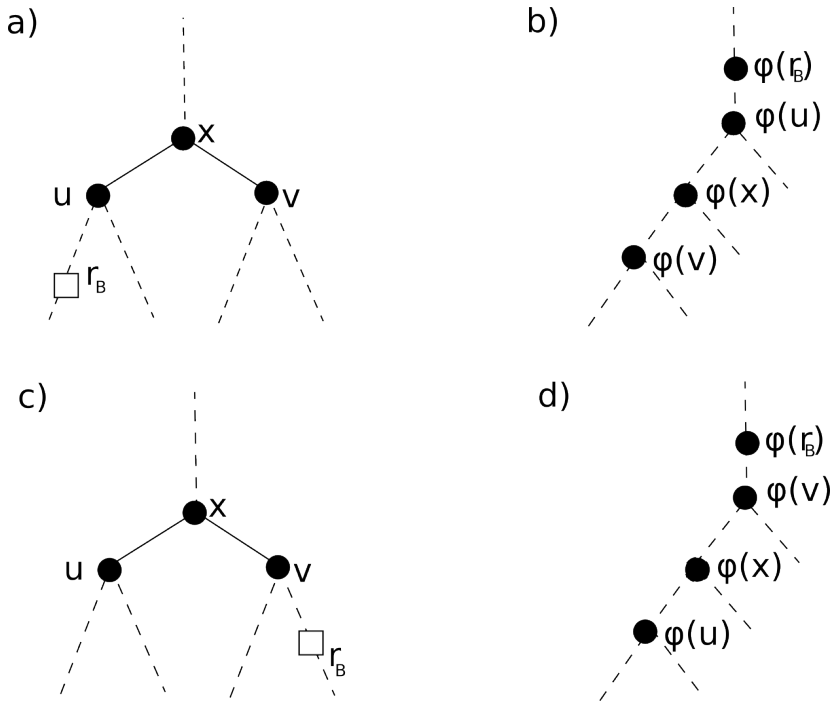

Figure 4.4: Example where root r_B of B can be a) The input gene tree B and r_B is sub-tree of u b) The input rooted species tree T c) The input gene tree B and r_B is sub-tree of v d) The input rooted species tree T

By this, we have just proved that the x is mapped correctly. The second part of the invariant, that after mapping x, the rest of unmapped vertices form the connected sub-graph of B is easy to prove. Because B is a binary tree, x has to be connected with 3 another vertices - with u,v and z. Because u,v have been already processed, there are only two possibilities a) z has been processed as well or b) z is still unmapped. If z has been processed, then thanks to induction we know that x was the last unmapped vertex and therefore there are not any another unmapped vertex, thus the invariant holds. If z has not been processed yet, it has to be connected with other two vertices and because (thanks to induction) before processing x, all unmapped vertices form connected sub-graph of B, after mapping x this condition is still valid (because x was connected only with one unmapped vertex. Thus we proved Claim 1.

Next part that is needed to prove is that we root an gene tree correctly. It is stayed as Claim 2.

Claim 2. *If an isometric reconciliation exists, Algorithm 4 roots the gene tree*

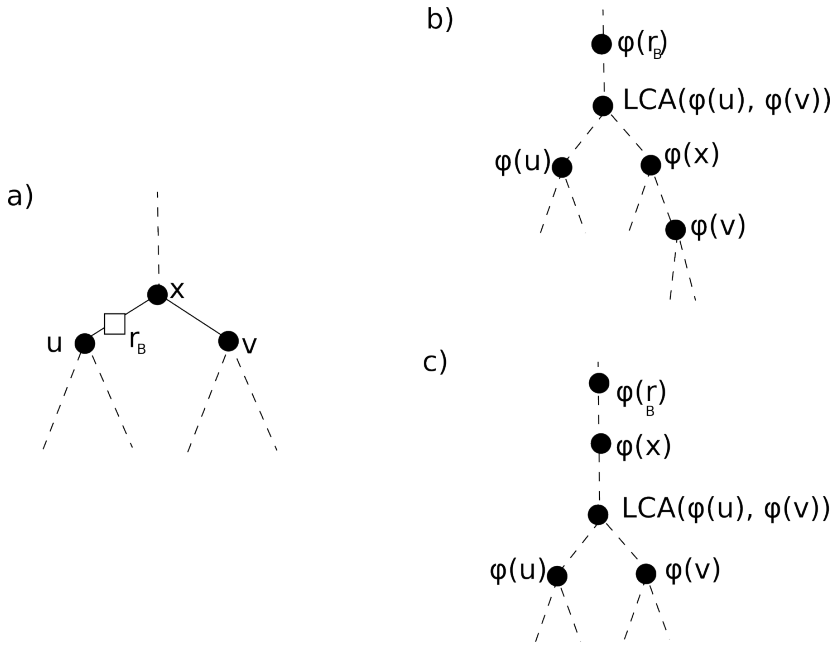

Figure 4.5: Example where root r_B of B can be a) The input unrooted gene tree B b) The input rooted species tree T c) The input rooted species tree T

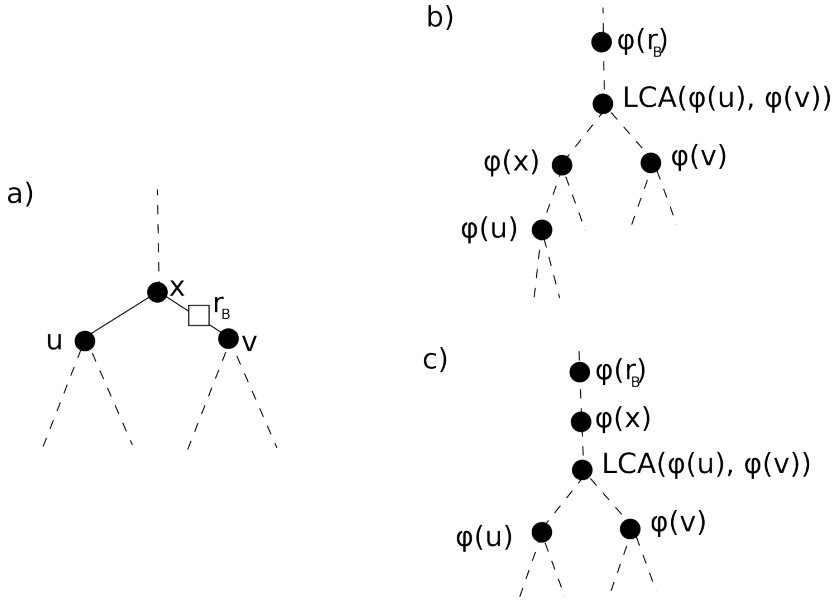

Figure 4.6: Example where root r_B of B can be a) The input unrooted gene tree B b) The input rooted species tree T c) The input rooted species tree T

correctly.

Proof 2.

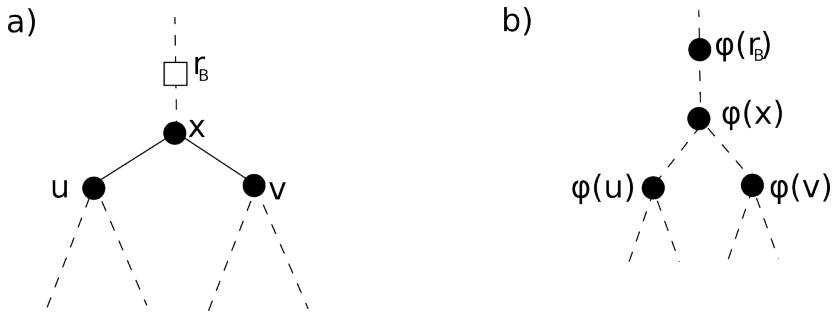

Figure 4.7: Example where root r_B of B can be a) The input unrooted gene tree B b) The input rooted species tree T

The Algorithm 4 iterates over all edges in a unrooted gene tree B. Because we assume that the isometric reconciliation exists, we can assume without loss of generality, that the roof r_B lies on edge with endpoints e and f. Let us investigate an edge with endpoints m and n which does not contain the root. In the correctly rooted version of B, either m is an ancestor of n or n is an ancestor of m. Thanks to the isometric reconciliation, this relationship is satisfied also after their mapping to the species tree T and $\phi(m)$ is an ancestor of $\phi(n)$ or $\phi(m)$ is an ancestor of $\phi(n)$. However then $\lambda = LCA(\phi(m), \phi(n))$ is $\phi(m)$ itself or $\phi(n)$ itself. Moreover we have to preserve distances and therefore $D(m,n) = D(\phi(m), \phi(n)) = D(\phi(m), \lambda) + D(\phi(n), \lambda)$ and therefore $\epsilon = D(m,n) - D(\phi(m), \lambda) - D(\phi(n), \lambda) = 0$. Thus Algorithm 4 will not root the tree at this edge (because condition at line 9 is not satisfied), which is correct.
Now we investigate the edge (e, f) which contains the root r_B. Let us mark $d_1 = D(e, r_B)$ and $d_2 = D(f, r_B)$, thus $d = d_1 + d_2$. Because root r_B lies between e and f, they are both descendants of r_B. But again thanks to the isometric reconciliation, $\phi(e)$, $\phi(f)$ and $\phi(r_B)$ have to satisfy this relationship as well. Therefore $\phi(r_B)$ has to be mapped above $\phi(e)$ and above $\phi(f)$ as well. However, that means that it has to be mapped at $\phi(r_B)$ above $\lambda = LCA(\phi(e), \phi(f))$, because otherwise $\phi(e)$ or $\phi(f)$ would not be the descendant of $\phi(r_B)$. Let us mark this distance as $D(\phi(r_B), \lambda)$. Based on this and a fact, that distances have to be preserved, we can derive that $d'_1 = D(phi(e), \lambda) < D(phi(e), \phi(r_B)) = D(e, r_b) = d_1$, thus $d'_1 < d_1$. Similarly, $d'_2 = D(phi(f), \lambda) < D(phi(f), \phi(r_B)) = D(f, r_b) = d_2$, thus $d'_2 < d_2$. Therefore we have $d = d_1 + d_2 > d'_1 + d'_2$. From $\epsilon = d - d'_1 - d'_2 = d - (d'_1 + d'_2)$ we have $d - \epsilon = d'_1 + d'_2 < d$ and finally $\epsilon \geq 0$. Moreover from the fact, that $\phi(r_B)$ is above λ and the fact that distances have to be preserved, we have $d_1 + d_2 =$

$D(e, r_B) + D(f, r_B) = D(\phi(e), \lambda) + D(\lambda, \phi(r_B)) + D(\phi(f), \lambda) + D(\lambda, \phi(r_B))$ and $\epsilon = d - d'_1 - d'_2 = (d_1 + d_2) - (d'_1 + d'_2)$. If we combine those equations, we get $\epsilon = D(\phi(e), \lambda) + D(\lambda, \phi(r_B)) + D(\phi(f), \lambda) + D(\lambda, \phi(r_B) - (D(\phi(e), \lambda) + D(\phi(f), \lambda)) = 2 * D(\phi(f), \lambda = \epsilon$. Thus $D(\phi(f), \lambda) = \epsilon/2$. Therefore algorithm correctly roots the tree at this edge (because condition at line 9 is satisfied) which is correct.

Claim 3. *If the algorithm finishes without rejection and roots the gene tree, the result reconciled tree T^* fulfils the definition of isometric reconciliation.*

Proof 3.

Because the last step of Algorithm 2 is running of Algorithm 5 that checks if the conditions of the isometric reconciliation are fulfilled, there cannot occur a situation where the algorithm finishes without rejection, roots the gene tree and the output does not fulfil the definition of the isometric reconciliation.

Claim 4. *The time complexity of this algorithm is $T = O((n + m)^2)$, where n is the number of vertices in a processed gene tree and m is the number of vertices in a species tree.*

Proof 4.

It is known that the finding the last common ancestor in a tree with n vertices can be done in $T = O(1)$ if the tree was preprocessed before. This preprocessing takes $T = O(n)$ [co00]. Although each vertex is mapped only once, in order to find the correct place for mapping, we need to traversal the species tree. Therefore the upper-bound complexity is $T = O((n+m) * (n+m)) = O((m+n)^2)$, where n is the number of vertices in a processed gene tree and m is the number of vertices in a species tree. Nevertheless we assume that by using the appropriate data structures, there is a space for improving this complexity.

Chapter 5

Conclusion

Our work consist of two parts. In the first part we explained what a reconciliation of gene trees is and how it works. Then we described the different approaches to reconciliation, starting with processing rooted binary gene and species trees and ended up with reconciliation of arbitrary gene and species trees. We considered only basic evolutionary events: speciation, gene duplication and gene loss. More complicated models taking into account other evolutionary events as horizontal gene transfer, lineage sorting or deep coalescence were skipped due to their complexity.

In the second part, we focused on reconciliation algorithm used in the work of Ma at al. [ha08]. Firstly, we showed that their definition of an isometric reconciliation is not strong enough and we proposed stronger definition. Secondly, we showed that their reconciliation algorithm does not work correctly. Therefore we proposed our reconciliation algorithm, that solves the same problem, with the proof of its correctness. Our algorithm has polynomial-time complexity, but there is an open space for its further improvements.

Bibliography

[ki81] J.F.C. Kingman: *The Coalescence*

[ma97] W. P. Maddison: *Gene Trees in Species Trees* Syst. Biol. (1997) 46(3):523-536

[zh97] L. Zhang: *On a Mirkin-Muchnik-Smith Conjecture for Comparing Molecular Phylogenies* J. of Comput. Bio. Vol. 4 (1997), No. 2, Pages: 177-187

[mi98] R. Durbin, S. Eddy, A. Krogh, G. Mitchison: *Biological sequence analysis* Cambridge University Press (1998)

[co00] M. A. Bender, M. Farach-Colton: *The LCA problem revisited*LATIN 2000: Theoretical Informatics. Springer Berlin Heidelberg, (2000), Pages: 88-94.

[sa01] D. Sankoff: *Gene and genome duplication* Current Opinion in Genetics and Development 2001, 11:681-684

[ed01] Ch. M. Zmasek, S. R. Eddy: *A simple algorithm to infer gene duplication and speciation events on a gene tree* Bioinformatics, Vol. 17, No. 9 (2001), Pages: 821-828

[do03] P. Bonizzoni, G. D. Vedova, R. Dondi: *Reconciling Gene Trees to a Species Tree*

[se03] L. Arvestad, A. Ch. Berglund , J. Lagergren, B. Sennblad: *Bayesian gene/species tree reconciliation and orthology analysis using MCMC* Bioinformatics (2003), Vol. 19, Suppl. 1, Pages: i7-i15

[go04] P. Górecki: *Reconciliation Problems for Duplication, Loss and Horizontal Gene Transfer*

[ha07] M. W. Hahn: *Bias in phylogenetic tree reconciliation methods: implications for vertebrate genome evolution*Genome Biology 2007, Vol. 8, Issue 7, Article R141

[na07] C. Than, L. Nakhleh: *SPR-based tree reconciliation: Non-binary trees and solutions*

[ha08] J. Ma, A. Ratan, B. J. Raney, B. B. Suh, W. Miller, D. Haussler: *The infinite sites model of genome evolution*

[ha08] J. P. Doyon , C. Chauve, S. Hamel: *Algorithms for exploring the space of gene tree/species tree reconciliation*

[le08] A. Kuzniar, R. C.H.J. van Ham, S. Pongor, J. A.M. Leunissen: *The quest for orthologs: finding the corresponding gene across genomes* Trends in Genetics, Vol.24, No.11

[la10] A. Tofigh, J. Lagergren: *Inferring Duplications and Lateral Gene Transfers - An Algorithm for Parametric Tree Reconciliation*

[se11] L. Arvestad, A. Ch. Berglund, J. Lagergren, B. Sennblad: *Gene Tree Reconstruction and Orthology Analysis Based on an Integrated Model for Duplications and Sequence Evolution*

[eu11] P. Górecki, O. Eulenstein: *Algorithms for Unrooted Gene Trees with Polytomies*

[be11] J. P. Doyon, V. Ranwez, V. Daubin, V. Berry: *Models, algorithms and programs for phylogeny reconciliation* Briefings in Bioinformatics . Vol 12., No 5., 392- 400 doi:10.1093/bib/bbr045

[zh12] Y. Zheng, T. Wu, L. Zhang: *Reconciliation of Gene and Species Trees With Polytomies* Bioinformatics, Vol. 00 (2011), No. 00, Pages 1-10

[mab12] M. Lafond, K. M. Swenson, N. El-Mabrouk: *An optimal reconciliation algorithm for gene trees with polytomies*

[ma12] K. M. Swenson, A. Doroftei, N. El-Mabrouk: *Gene tree correction for reconciliation and species tree inference* Algorithms for Molecular Biology (2012), 7:31

[eu12] P. Górecky, O. Eulenstein: *Deep Coalescence Reconciliation with Unrooted Gene Trees: Linear Time Algorithms*

[cha12] J. P. Doyon, S. Hamel, C. Chauve: *An Efficient Method for Exploring the Space of Gene Tree/Species Tree Reconciliations in a Probabilistic Framework*

[ly13] L. Y. Rusin, E. V. Lyubetskaya, K. Y. Gorbunov, V. A. Lyubetsky: *Reconciliation of Gene and Species Trees*

[na13] L. Nakhleh: *Computational approaches to species phylogeny inference and gene tree reconciliation* Trends in Ecology and Evolution (2013), Vol. 28, No. 12

[de13] J. H. Degnan: *Anomalous Unrooted Gene Trees* Syst. Biol. (2013), 62(4):574-590

[ti13] P. Górecki, O. Eulenstein, J. Tiuryn: *Unrooted Tree Reconciliation: A Unified Approach*IEEE/ACM Transaction on Computational Biology and Bioinformatics, Vol. 10, No. 2

[zh14] Y . Zheng, L. Zhang: *Reconciliation with Non-binary Gene Trees Revisited*

[ke14] Y. Ch. Wu, M. D. Rasmussen, M. S. Bansal, M. Kellis: *Most Parsimonious Reconciliation in the Presence of Gene Duplication, Loss, and Deep Coalescence using Labeled Coalescent Trees* Genome Research (2014) 24: 475-486