UNIVERSITÀ DEGLI STUDI DI PADOVA

FACOLTÀ DI INGEGNERIA

Laurea Magistrale in

INGEGNERIA INFORMATICA

Minimization of the Crossing Number in a Reconciled Tree

October 17, 2012

Relatore

Cinzia Pizzi

Correlatore

Lars Arvestad

Candidato

Marco Pierobon

Anno Accademico 2011/2012

Contents

Abstract

The problem of building a reconciled tree from a gene and a species tree is of great interest in bioinformatics, since it allows to highlight the relation between the evolution of the genes with the evolution of one or more species. In particular, a reconciled tree with a minimized number of crossings can help tracing down accurately the biological events of interest and avoiding wrong assumptions on the available data. The way the operation is performed affects heavily the usefulness of the resulting tree.

The Minimized Crossing Number in a Reconciled Tree problem(MCNRT) problem is an NP-Complete problem. In this thesis we present an heuristic that aims to provide a solution for this problem. The proposed algorithm is based on the solution of the Generalized Tanglegram Layout problem, that has been adapted to work for the whole tree structure. Finally, the algorithm has been integrated in a running project, namely PrimeTV. The resulting code has been tested and proved able to optimize the number of crossings under different conditions.

Sommario

Il problema della costruzione di un albero riconciliato a partire da un albero di evoluzione dei geni e da un albero di evoluzione delle specie è di grande interesse in bioinformatica, dato che permette di evidenziare la relazione fra l'evoluzione dei geni e l'evoluzione di una o più specie. In particolare, un albero riconciliato con un numero di attraversamenti minimizzato può aiutare ad identificare gli eventi di interesse biologico e ad evitare di fare assunzioni sbagliate sui dati disponibili.

Il problema Minimized Crossing Number in a Reconciled Tree (MCNRT) è un problema NP-Completo. In questa tesi è presentata un'euristica che fornisce una soluzione a tale problema. L'algoritmo proposto è basato sulla soluzione del problema Generalized Tanglegram Layout, che è stata adattata in modo da poter lavorare sull'intera struttura dell'albero. Inoltre, l'algoritmo è stato integrato come parte di un progetto attivo, PrimeTV. Il codice risultante è stato testato e si è dimostrato capace di ottimizzare il numero di attraversamenti sotto differenti condizioni.

Introduction

This thesis is focused on the study of the Minimization of the Crossing Number in a Reconciled Tree problem and on its solution. In biology the study of species and gene trees allow scientists to have a better understanding of the evolution of specific genes and to determine the relation between species. A reconciled tree merges the informations a species tree and a gene tree provide. It allows also to identify event of evolutionary interest such as gene duplication, gene speciation and horizontal gene transfer. A reconciliation is defined by means of a map that connects the nodes in the gene tree with the nodes in the species tree. The reconciliation process is well known and is fully described in [8]. It can be performed using different metrics, like the count of the number of evolutionary event or a most likelihood criteria [4]. Such a reconciliation can be described graphically by a tree, where the gene nodes are positioned inside the species node they have been assigned to by the map. The layout obtained has usually several edges crossings. These crossings may mislead the scientists analyzing the problem, making them think biological events have occurred even when it is not what had happened in reality. It follows that a reconciled tree with a minimized number of edge crossings is desirable. The problem of the minimization of the number of edge crossings in reconciled trees has not been treated specifically. There is thus the need of a study of the problem and of an algorithm that solves it.

This thesis is organized as follows:

- The first chapter introduces the notions of Species Tree and Gene Tree. In addition two ways to perform a reconciliation are described. The first one involves tanglegrams, a graph where the leaves of the species tree and the leaves of the gene tree are connected with a set of inter-tree edges. The second one defines a tree-within-tree reconciliation, where the map between the two trees regards every node, not just the leaves. Finally, the chapter defines the Minimization of the Crossing Number in a Reconciled Tree problem and the Generalized Tanglegram Layout problem.

- The second chapter describes PrimeTV [10], the software in which the solution of the problem has been implemented, as well as the Newick format, that is the format in which the input data of PrimeTV is required to be written. PrimeTV has been developed by the Stockholm Bioinformatics Center of the Stockholm University and is a tool that produces graphical output of reconciled trees. PrimeTV relies on the plotutils [9] library to output the reconciled tree in a variety of formats.

- The third chapter describes the heuristic that solves the problem and describes the transformation, which maps every layer of the problem into an instance of the Generalized Tanglegram Layout problem. Every layer is composed of a node of the species tree and its two children, together with the gene nodes associated with them. Furthermore the basic property of the crossings in the layout is defined and the way the crossings are counted is formalized. This allows to determine when a node rotations reduces the number of crossings by comparing the original layout with the reversed layout. In addition a description of the way the algorithm has been implemented in the code is presented.

- The fourth chapter illustrates the results achieved by the proposed algorithm. Two different sets of tests are performed: the first set is intended to evaluate the performance of the heuristic with real biological data and the second set with random generated data. The tests are intended both to determine the number of instances that the algorithm is able to improve as well as the magnitude of the improvements, namely the number of crossings avoided.

1 Reconciled Gene and Species Trees

In this chapter we give the basic biological background needed to understand the problem on which the thesis focus, namely the reconciliation of gene and species trees.

1.1 Gene and Species Trees

Let us start with the definition of Gene trees and Species trees.

1.1.1 Gene Tree

Genes hold the information used by an organism to build and maintain its cells and to transmit genetic traits to its offspring. Every organism has several genes, each of which corresponds to one or more biological traits.

A gene can be defined as a sequence of nucleic acid (generally DNA), whilst one of its variant is called an allele.

Due to gene replication, the evolution of a gene is represented as a gene tree: every time there is a replication of a gene copy at a specific locus (the position of a gene in a chromosome) and its different versions (alleles) are passed to more than one offspring a new branch in the gene tree is created. Since that gene copy has a unique ancestral version, the resulting history is a branching tree. In addition a gene mutation may lead to an imperfect representation of the original tree; sexual reproduction and recombination reduce the genomic history into many small pieces, where each piece has a strictly treelike pattern of descent [6].

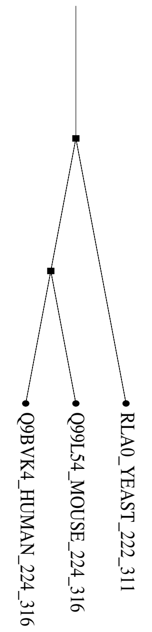


Figure 1.1: Example of a gene tree. Q9BVK4_HUMAN_224_316 and Q99L54_MOUSE_224_316 are more closely related to each other than to RLAO_YEAST_222_311

The study of a gene tree gives scientists many insights on the evolution of the genes, allowing them to track down events of interest and to get them visualized with a time reference.

1.1.2 Species Tree

A species is often regarded as a group of organism capable of interbreeding and producing a fertile offspring.

A species tree can be defined as the pattern used to branch species lineages through the process of speciation, namely the event that leads to the creation of two different species from a common ancestor.

When the reproductive communities are split by a speciation event the gene copies held by this communities are split likewise into separate bundles of descent. Inside each bundle, the gene trees keeps on evolving, by mean of branches and descends as the time passes [6].

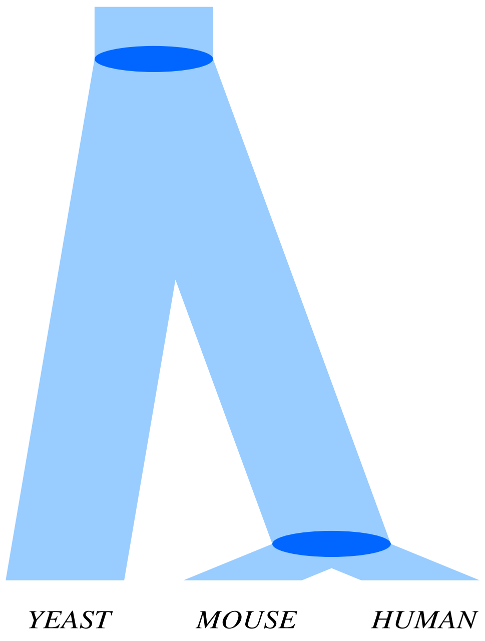


Figure 1.2: Example of a species tree. Mouse and Human are species more closely relate to each other than to Yeast.

Species trees give the opportunity to study the relation between species and to determine their distance in an evolutionary context, allowing to determine which species are closer to each other and to identify common ancestors.

1.2 Reconciliations

Merging the information contained in the gene and species trees may be useful to identify cross species gene transfer, to determine the rate of evolution and to make hypothesis on the evolution of species.

This can lead, for instance, to a better analysis of phenomenas such as:

- gene duplication
- gene speciation: the transmission of different version of the gene to the offspring
- gene loss: the loss of the gene by a species
- deep coalescence: the failure of ancestral genes copies to look back in time until deeper than the previous speciation event

- horizontal gene transfer: the event of transmission of genetic content independent from reproduction.

The latter is particularly difficult to analyze because when such an event occurs (for instance, a cell ingesting large DNA molecules) the resulting DNA sequence may be unpredictable. Even if this kind of event affects mostly not complex living forms, it has been shown that even higher organisms can be subjects of it[11].

Usually the reconciliation process described above is obtained in two different ways: by using tanglegrams or by defining a tree-within-tree reconciliation.

1.2.1 Tanglegrams

The first approach, as described in [1] is carried out using tanglegrams.

A tanglegram is a graph containing two trees (in this context the species and the gene trees), where their leaves have been firstly aligned on two parallel lines. Then a two dimensional layout is specified, in order to connect every gene leaf to the species leaf that very gene was sampled from. Every such connection is usually referred to as a tangle, or a tangle edge, and it is represented as a straight line. This operation defines a series of inter-tree edges, whose number depends on the two trees.

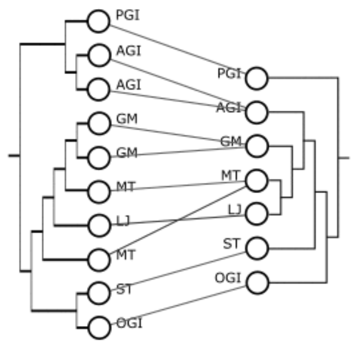


Figure 1.3: Leaves layers of a gene tree (on the left) and of a species tree (on the right) joined by tangles showing the relation between them. Picture is from [1] by courtesy of Bansal.

1.2.2 Tree within tree reconciliation

The second approach requires to define a tree-within-tree structure using a reconciliation. By doing this, a map between the nodes (both leaves and internal nodes) of the two trees is specified. This map allows to establish a connection between a gene and the species that gene was sampled from and to study its evolution along with the evolution of the species. This process permits to represent hypothesis on the evolution of a set of species by means of phylogeny trees[12]. It should be noted that the reconciliation process stemmed independently from different areas of study in biology, such as biogeography and molecular systematics, aiming the description of historical associations and is therefore suited also for different uses, like for instance the study of host-parasite cospeciation [8].

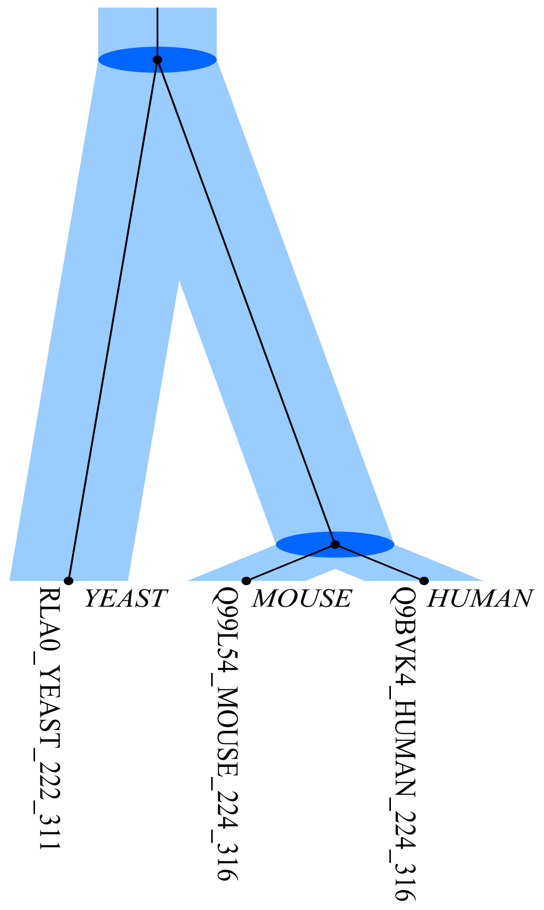


Figure 1.4: Example Figure of a tree within tree reconciliation from gene and species tree of Figure 1.1 and Figure 1.2, respectively. The bottom layer contains the leaves of both the trees. The name of the species is written horizontally, the name of the gene is written vertically.A circled dot in the bottom layer means there is a gene leaf there, in an upper layer means a gene speciation. The blue oval means the species has speciated.

Finally, it needs to be remarked that a gene tree can disagree with its containing species tree: one could think that sister species will have sister copies of genes tree and that the other aspects related to the gene tree will be congruent with the species tree, but this is not always the case [2]. This last aspect is the reason why the reconciliation process is difficult to tackle.

The basic algorithm performing the construction of a reconciled tree is well known and described accurately in [8].

This operation can be carried out using different evaluation metrics, such as the number of evolutionary event or a most likelihood criteria [4]. In this work we will suppose the reconciliations are performed with the most parsimonious criteria, which assumes the smaller number of genetic events.

The problem of minimizing the number of crossings in such a tree has been named Minimizing Crossings Number in a Reconciled Tree and will be referred to as MCNRT hereafter.

1.3 Minimizing Crossing Problem

In order to fully understand the nature of the problem it is necessary to introduce some amount of formalism. In the present chapter we will try to keep this formalism at a minimum, although in some cases a more detailed model would be preferred or required. Such a model can be found in [5].

Definition 1 (**Species Tree**). *Let $\mathcal{I}$ containing $\mathcal{N}$ different species. An evolutionary species tree $\mathcal{S}$ defined over it is a tree where:*

a. *$\mathcal{S}$ is a rooted binary directed tree*

b. *$\mathcal{S}$ has $\mathcal{N}$ leaves, each one uniquely labeled by an element of $\mathcal{I}$*

Definition 2 (**Gene Tree**). *Similarly an evolutionary gene tree $\mathcal{G}$ is a tree such that:*

a. *$\mathcal{G}$ is a rooted binary directed tree*

b. *the leaves of $\mathcal{G}$ are labeled by elements of $\mathcal{I}$*

Let $L(\mathcal{T})$ indicate the set of leaves that occurs in the subtree rooted at $\mathcal{T}$.

Usually when working with reconciliations the assumption $L(\mathcal{S}) = L(\mathcal{G})$ is made. As shown in [3] this does not imply any loss of generality.

Definition 3 (**Reconciled Tree**). *A reconciled tree $T_{\mathcal{R}}(\mathcal{G},\mathcal{S})$ is the smallest tree having leaves labeled such that:*

a. *it contains only sets from $\mathcal{S}$*

b. *$\mathcal{G}$ is contained as a subtree*

c. *for two children a and b of $g \in T_{\mathcal{R}}(\mathcal{G}, \mathcal{S})$ we have either $L(a) \cap L(b) = \emptyset$ or $L(a) = L(b) = L(g)$*

Given a rooted tree T, $V(T)$ and $E(T)$ are respectively its node and edge set. A node in $V(T)$ that is not a leaf is called an internal node. The root node of T is denoted by $rt(T)$. Given a node $v \in V(T)$, $pa(v)$ denotes the parent of v in T , $Ch(v)$ is the set of children of v, and $T(v)$ denotes the subtree of T rooted at v. If two nodes in T have the same parent, they are called siblings.

Let us define $E(S, G)$ as the embedding of the gene tree inside the species tree, such that $E(S, G) \subseteq S \times G$, where each node of S is incident on at least one edge in $E(S, G)$, and each node in G is incident on at least two edges in $E(S, G)$.

Furthermore, we define an order τ over the set $E(S, G)$, such that if

$(u, v) <_{\tau} (u, w)$, $u \in S$, $v, w \in G$, then the gene node v is located before w in the mapping inside the species node u. The result is a tree R embedding G inside S with an order defined by τ.

An example is shown in Figure 1.5. The elements in the figure would contribute to $E(S, G)$ with a subset

$$\{(s1, g1), (s1, g2), (s1, g3), (s1, g4), (s1, g5),$$
$$(s2, g1), (s2, g3), (s2, g4), (s3, g2), (s3, g5)\}$$

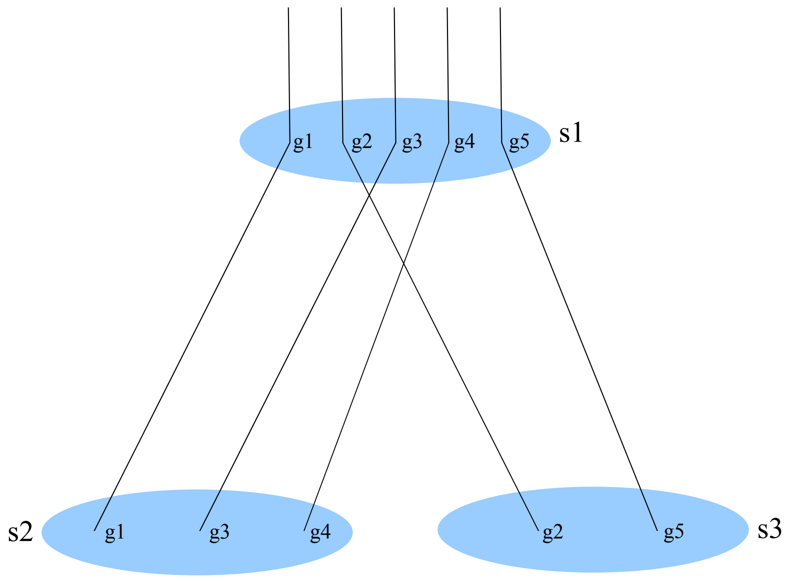


Figure 1.5: Part of a reconciled tree

Similarly, let us call σ the order on S that defines the layout of the tree S, which allows to determine for every $v \in S$ which son is the left or right one.

Finally, we need to define the function to evaluate the quality of the layout. Given the above mentioned σ and τ defined, respectively, over S and $E(S,G)$ the number of crossing between σ and τ is denoted as $cr(\sigma, \tau, E(S,G))$ and equals to

$$|\,(u,v)\,,(u,w) \in E(S,G) : (((u,v) <_\tau (u,w)) \curlywedge ((u',w) >_\tau (u',v))) \curlyvee$$
$$\curlyvee\,(((u,w) >_\tau (u,v)) \curlywedge ((u',v) <_\tau (u',w)))\,|$$

where u' is one of the two children of u.

Now we can define the basic problem of the minimization of the crossings number in a reconciled tree.

Definition 4 (**Minimizing Crossings Number in a Reconciled Tree (MCNRT)**)**.** *Given an instance $< S, E(S,G) >$ find compatible linear orders σ and τ, respectively on S and on $E(S,G)$, such that $cr(\sigma,\tau,E(S,G))$ is minimized.*

Theorem 5 (**MCNRT is NP-COMPLETE**)**.**

Wotzlaw et al. showed in [12] that the Generalized k-ary tanglegrams on level graphs is a NP-Complete problem for $k > 1$.

Being MCNRT its restriction where the trees are binary (k=2) it follows it is a NP-Complete problem.

The solution of MCNRT relies on the solution of the Generalized Tanglegram Layout (GTL) problem, that will now be stated.

Definition 6 (**Generalized Tanglegram Layout Problem**)**.** Given an instance $< S, T, I(S,T) >$, where S and T are trees and $I(S,T)$ is the set of the intertree edges connecting the leaves of S with the leaves of T, find compatible linear orders σ and τ on trees S and T , respectively, such that $cr(\sigma,\tau,I(S,T))$ is minimized.

2 PrimeTV

PrimeTV(PRobabilistic Integrated Models of Evolution Tree Viewer) is a software developed by the Stockholm Bioinformatics Center of the Stockholm University. It allows to get a graphical representation of a tree-within-tree reconciliation, by the use of the plotutils library.

It is written in C++ and accepts two files as input: the first one is the reconciled tree and the second one is the species tree. Both the trees have to be specified in the Newick format.

The produced output is a graphical file showing the reconciliation between the species and the gene tree. The file format of the output can be chosen to be an X window or an image in raster or vector format.

2.1 PrimeTV Input

The Newick format allows to define the structure of the trees in a very straightforward and intuitive way. It was introduced in 1857 by the English mathematician Arthur Cayley. The current accepted standard is defined in [7]. We will limit ourselves to an introduction of the basic notation that allows to define the tree structure along with the lengths of the branches. We then explain the extended Newick format, that is what is required to run PrimeTV.

2.1.1 Newick basic format

Each internal node is identified by a pair (id_1, id_2), where $id_1$and id_2 are the children of the node.

The distance of a leaf node from its father is specified by placing a colon and the distance after the node, e.g. $id_1 : length$.

The distance of an internal node from the father is specified by placing a colon and the distance after the parenthesis that identify the node,
e.g.$(id_1, id_2) : length$.

A semicolon is put at the end of the tree definition.

So for example for the tree depicted in Figure 2.1 we would obtain a notation like:

```
(
    (MOUSE:0.0110,
    HUMAN:0.0110):0.1466,
YEAST:0.1576);
```

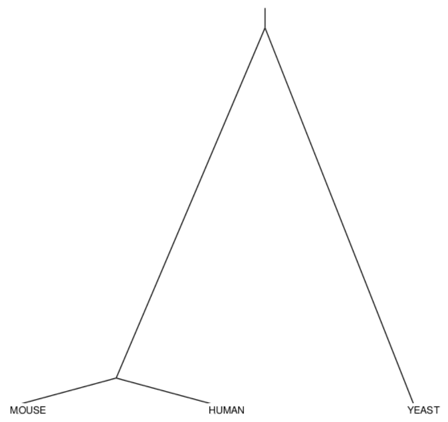


Figure 2.1: Tree built from Newick notation. Mice and Humans are 0.011 far apart from each other and 0.1466 from Yeast.

2.1.2 Newick extended format

The Newick extended format needed to run PrimeTV requires to add a label to each node. This label allows to refer to the node from the file where the reconciliation is defined.

In order to show how such labels are applied let us see a small example. In the Figure 2.2 we have:

a. Species tree S in Newick format

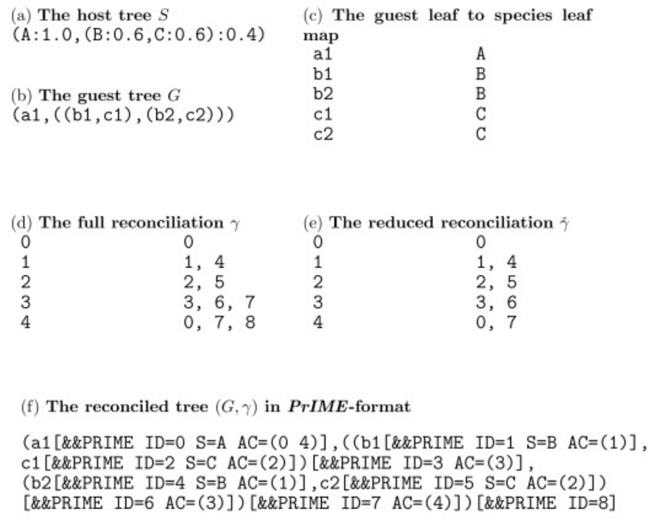
b. Gene tree G in Newick format

c. Leaf map between the species tree in a) and the gene tree in b) in a tabular form

d. The full reconciliation γ in a tabular format. For a node x of the species tree (on the left column),$\gamma(x)$ (on the right column in the same row) includes all the gene tree nodes whose incoming edges appear on the incoming edge of x in the reconciled tree. Notice a gene node can be mapped to several species vertices

e. The reduced reconciliation $\tilde{\gamma}$ in tabular format, obtained from the full reconciliation γ by removing from $\gamma(x)$ all gene vertices that are ancestral to other vertices in $\gamma(x)$

f. The reconciled tree (x, γ) in Prime format. This is a Newick tree with Prime tags added to its nodes; every sequence of Prime tags are always given within brackets and are preceded by the tag $\&\&PRIME$. For a gene vertex, v, the tag ID indicates a unique number identifying v. The tag AC indicates the ID of the species tree node that v maps to in the reduced reconciliation; these species tree nodes should always form a path in the species tree. Finally, for a leaf l, the tag S indicate the label of the gene tree leaf that the leaf l maps to.



(a) **The host tree** S

```
(A:1.0,(B:0.6,C:0.6):0.4)
```

(b) **The guest tree** G

```
(a1,((b1,c1),(b2,c2)))
```

(c) **The guest leaf to species leaf map**

a1	A
b1	B
b2	B
c1	C
c2	C

(d) **The full reconciliation** γ

0	0
1	1, 4
2	2, 5
3	3, 6, 7
4	0, 7, 8

(e) **The reduced reconciliation** $\tilde{\gamma}$

0	0
1	1, 4
2	2, 5
3	3, 6
4	0, 7

(f) **The reconciled tree** (G, γ) **in** ***PrIME*-format**

```
(a1[&&PRIME ID=0 S=A AC=(0 4)],((b1[&&PRIME ID=1 S=B AC=(1)],
c1[&&PRIME ID=2 S=C AC=(2)])[&&PRIME ID=3 AC=(3)],
(b2[&&PRIME ID=4 S=B AC=(1)],c2[&&PRIME ID=5 S=C AC=(2)])
[&&PRIME ID=6 AC=(3)])[&&PRIME ID=7 AC=(4)])[&&PRIME ID=8]
```

Figure 2.2: Input formats of PrimeTV. Image is from [10], by courtesy of Lars Arvestad.

The extended Newick representation for the tree shown in Figure 2.1 is:

```
(
    (MOUSE:0.0110[PRIME ID=1],
    HUMAN:0.0110[PRIME ID=2]):0.1466[PRIME ID=3],
YEAST:0.1576[PRIME ID=0]);
```

Similarly as it has been done for the species tree we can define the reconciliation in the newly defined notation:

```
(
    ( Q99L54_MOUSE [&&PRIME S=MOUSE AC=(1)],
    Q9BVK4_HUMAN [&&PRIME S=HUMAN AC=(2)] )
        [&&PRIME AC=(3)],
RLA0_YEAST [&&PRIME S=YEAST AC=(0) ] ) [&&PRIME AC=(4) ];
```

By doing so we get the full input required to run PrimeTV. It can be seen how the reconciliation specifies a genes tree and how each node is mapped into the species tree by using the previous defined labels.

2.2 PrimeTV Processing and Output

In order to create a graphical visualization of that tree the program needs to gather some more data from the input files, like the maximum number of genes nodes within a species node. Then the coordinates where to put each species and gene node are calculated.

When those information have been brought together PrimeTV uses the plotutils library [9] to get a graphical output, by printing the two trees separately [10].

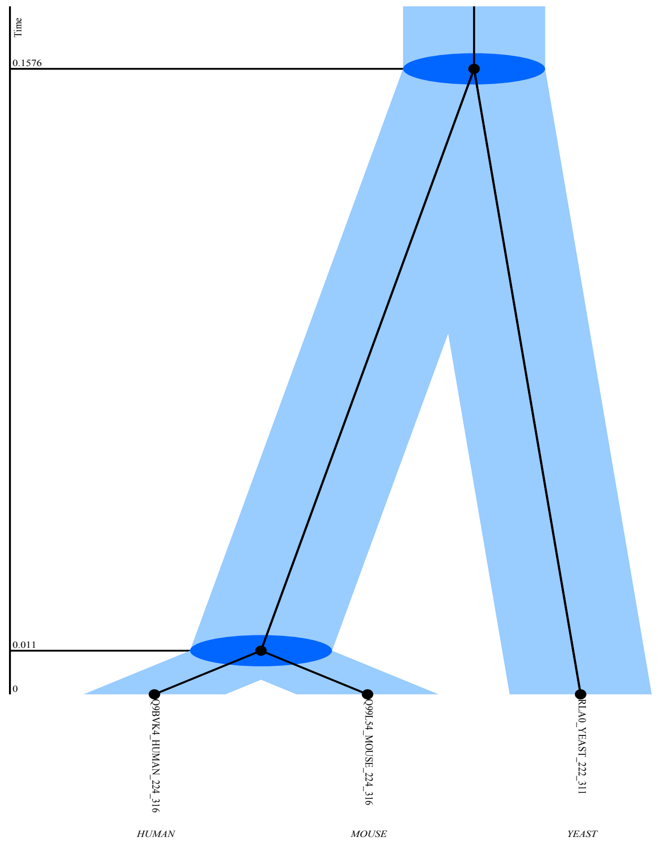


Figure 2.3: PrimeTV output from the above defined species and reconciled trees

If the input is simple, in terms of number of gene and species nodes in the reconciled tree, PrimeTV faces no problems in displaying the graphical output in the most correct way. However, when the input size starts to grow, and consequently its complexity, some limitations of the software emerge.

If for instance we would provide the program with the following normal sized inputs:

Species Tree

```
(
    (
        (
            (
                (
                    (
                        (ANASP_6:0.0200000[&&PRIME ID=7],
                        ANAVT_1:0.0200000[&&PRIME ID=8])
                        100:0.260000[&&PRIME ID=9],
                    TRIEI_1:0.280000[&&PRIME ID=6])
                    99:0.110000[&&PRIME ID=10],
                SYNY3_4:0.390000[&&PRIME ID=5])
                99:0.110000[&&PRIME ID=11],
            SYNE7_1:0.500000[&&PRIME ID=4])
            84:0.160000[&&PRIME ID=12],
                (PROM9_1:0.380000[&&PRIME ID=13],
                    (SYNS3_1:0.220000[&&PRIME ID=14],
                        ((SYNSC_1:0.100000[&&PRIME ID=17],
                        SYNPX_1:0.100000[&&PRIME ID=16])
                        50:0.0300000[&&PRIME ID=18],
                    SYNS9_1:0.130000[&&PRIME ID=15])
                100:0.0900000[&&PRIME ID=19])
            100:0.160000[&&PRIME ID=20])
        100:0.280000[&&PRIME ID=21])
        100:0.130000[&&PRIME ID=22],
            (SYNJB_1:0.0700000[&&PRIME ID=1],
        SYNJA_1:0.0700000[&&PRIME ID=2])
    100:0.720000[&&PRIME ID=3]):0.210000[&&PRIME ID=23],
GLVIO1_1:1.00000[&&PRIME ID=0]):0.00000
[&&PRIME ID=24][&&PRIME NAME=G]
```

(Reconciliation omitted)

we will get an output with an unoptimized layout, as the one depicted in Figure 2.4. It can be seen how the program does not handle the minimization of the crossing edges in the reconciled tree, since even by some simple rotations of the species tree numerous crossings could be avoided. (pick for instance the two species nodes where the gene SYNJA_1_PE1339 and the gene SYNPX_1_PE1525 are mapped: a rotation there would avoid a crossing).

Since each crossing can be traced back to an evolutionary event, it is important to have a layout that minimizes them, in order to get a better understanding of the data.

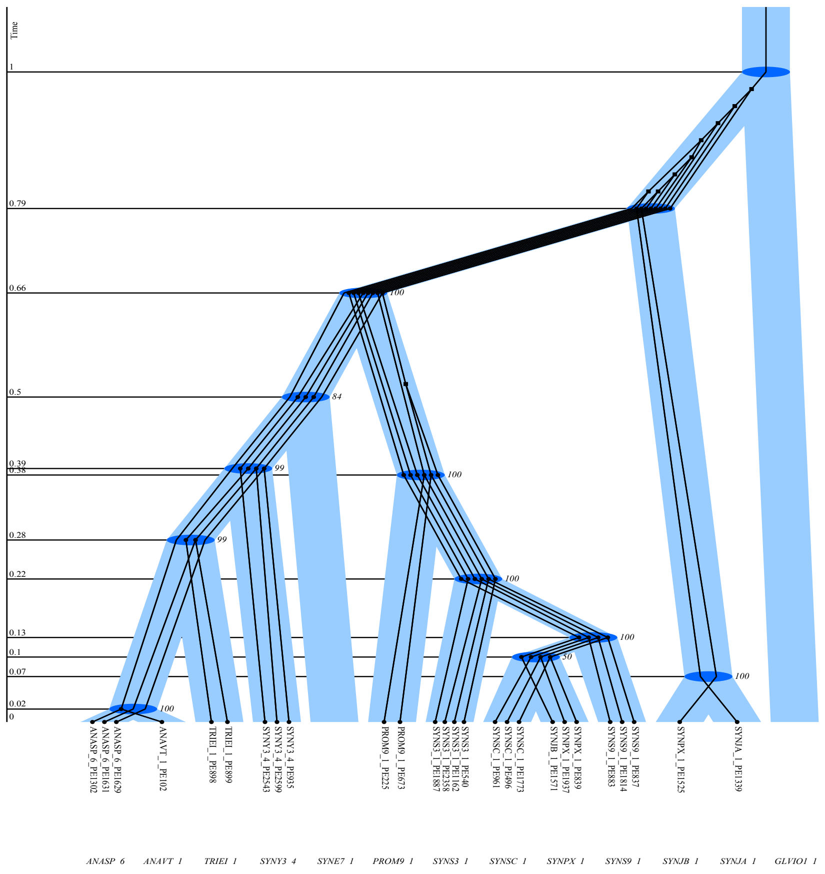


Figure 2.4: Layout of the middle sized example input

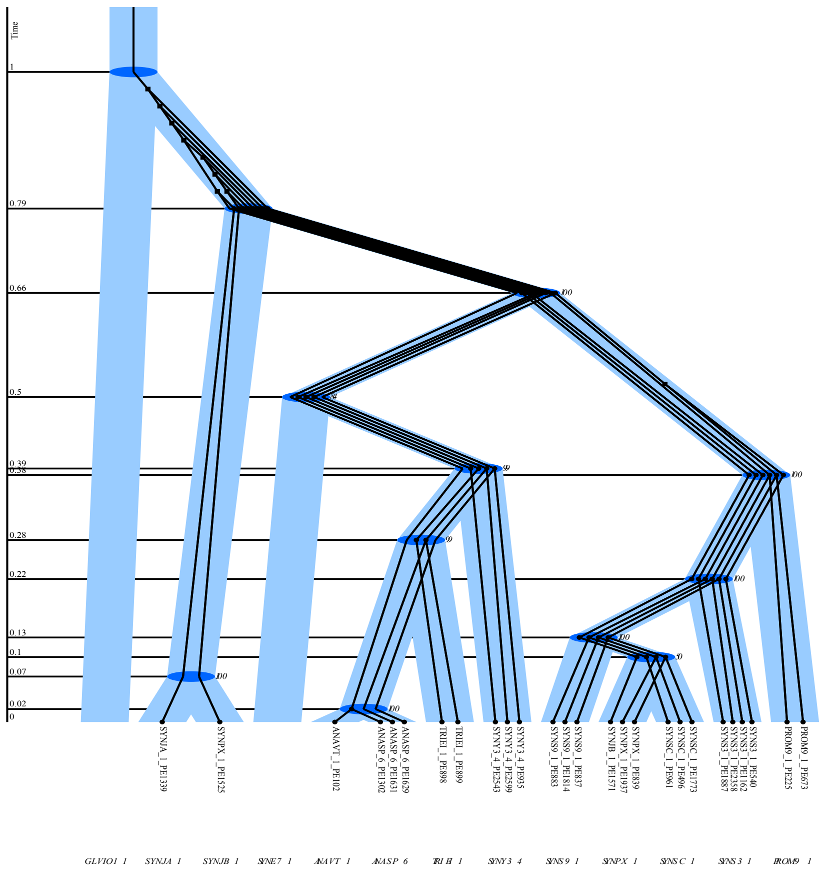


Figure 2.5: Optimized layout of the middle sized example input

3 Crossing minimization

The output of PrimeTV shows how simple trees rotations could lead to significant reductions in the number of crossings, and hence in the improvement of the overall quality of the visualization.

Detecting the right conditions when to apply such an intervention is not as straightforward as it may appear. This is caused both by the nature of the problem (the minimization of the number of crossing in a graph is a NP-complete problem) and by the difficulty in predicting how evolutionary data changes over time. There is thus the need of an heuristic, which will set a reasonable tradeoff between the quality of the final output and the time required to attain it.

3.1 Proposed Heuristic

As a contribution to this thesis we propose an algorithm that tries to identify when rotations should be applied to the reconciled layout. The idea is based on the solution of the tanglegram problem, described in [1]. This algorithm is supposed to provide a solution that minimizes the number of crossings in the mapping between the leaves of the gene and the species tree. We will extend this approach to work with the other layers of the reconciled tree. In order to do so the algorithm will operate iteratively on the tree structure, by working on one layer at a time. Each layer is composed by an internal node of the species tree and by its left and right childs. An example of such a layer is shown in Figure 3.1.

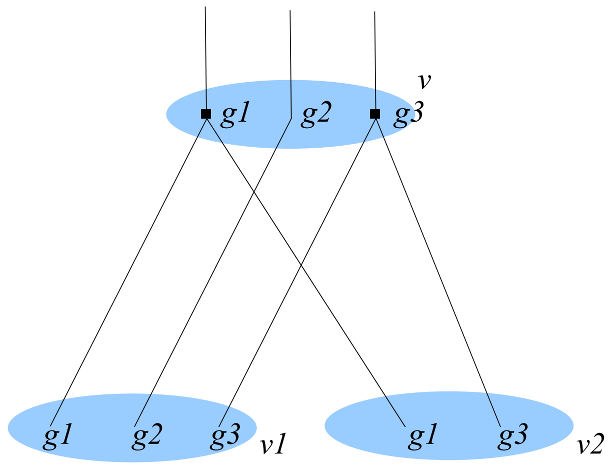


Figure 3.1: Example of a layer of the problem, gene evolution is described by lines, a square indicates a gene speciation. A blue oval indicates a species.

At every iteration we will minimize the number of crossings of the gene edges in the current layer. Repeating the process for every layer of the reconciled tree, we will obtain an heuristic that works on the entire structure.

By considering one layer at a time, the algorithm optimizes the layout for the current layer, not caring about the constraints it will impose on the following ones. Since the number of crossings is related to the number of gene edges and the number of branches of the gene tree increases with the height of the tree, a non-optimal layout for an upper layer is preferred over a non-optimal layout for a lower one. For this reason a bottom up approach has been chosen.

For every layer the algorithm determines the best species tree structure, by comparing the number of crossing in the default configuration with the number of crossings after the rotation, when the left child becomes the right child and viceversa (in Figure 3.1 with such a rotation we would have *v2* to be the left child and *v1* the right child).

The best configuration among the two will be the starting point of the optimization of the layout. To perform the next step of the optimization we put the layer in the configuration required by the GTL algorithm to minimize the number of crossings for the current layer. This is done in two steps:

- we ensure a one to one mapping between the gene nodes that lie in the father species node and the ones in the children species node
- we build an additional binary tree structure for the nodes mapped into the species father

The first step allows to count the number of crossings by using the position of the gene node in the species node, as described in 3.2.4. The second is required to determine the number of crossings for different layouts without changing the reconciled tree structure.

This process allows to handle every layer of the reconciled tree as an instance of the GTL problem.

3.2 Implementation of the algorithm

PrimeTV is written in C++ and therefore all the modification described has been implemented in this programming language.

However, to allow the reader to easily understand the way the algorithm works, all the code will be described in the pseudocode language.

We will now describe the basic steps performed by the heuristic, namely how a gene node is mapped inside a species one, how the crossings are counted and how the structure to perform such an operation is built.

3.2.1 Implementation of the Map

As described in Chapter 1.3 and in Chapter 2, in the reconciled tree gene nodes are mapped into species nodes.

This mapping is performed by a function γ, and it is implemented in the code by the class *GammaMap*. As described in the comments of the class, written by the original author, *GammaMap implements the map between the nodes of the species tree and the ones of the gene tree, by defining such a mapping as the function* γ*:V(S)*$\rightarrow 2^{V(G)}$*, where* $\gamma\,(x)$ *comprises all the vertices in* G *that has incoming edges whose pass through the edge* $(x, parent\,(x)) \subset S$*.*

The first thing we have to do is to transform the problem instance in order to have a 1-to-1 mapping between the gene nodes in the current layer, obtaining a structure similar to the one described in [1]. Figure 3.1 shows an example of a layer where such a condition is not met.

The code described in Algorithm 1 starts from the last layer and builds two layouts: σ and τ, according to the definitions provided in Chapter 1.3, ensuring they are of the same size. Both layouts contain the gene nodes to be ordered in the current layer together with an index. The index for an element of σ is just its position in the array, whilst for an elements in τ it is the position the same node has among the gene nodes in the two child species nodes of the layer, where their elements are considered as a unique set. Every gene has its descendant checked to see if they belong to the level. If it is the case they are added to the layouts and set as siblings in the additional tree structure, that will be explained in Section 3.2.2.

It needs to be remarked that the gene trees is not affected by these operations, since they are performed in a secondary data structure.

Starting from Figure 3.1, after the execution of the algorithm we would end up with the configuration shown in Figure 3.2. The resulting layouts σ and τ are shown in Table 1.

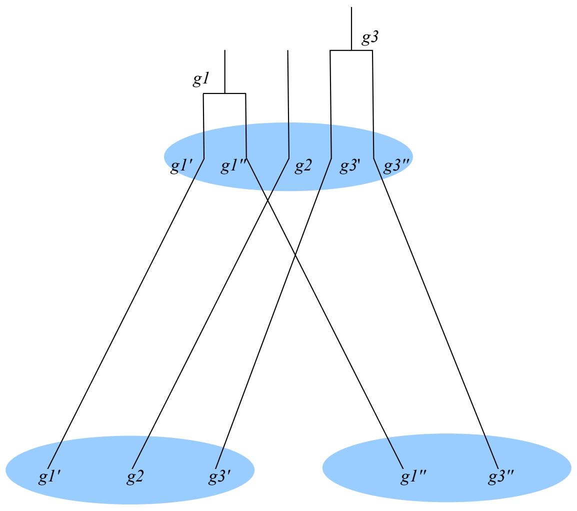


Figure 3.2: The instance of Figure 3.1, modified in order to have a one to one mapping between the nodes of the gene tree

```
Data: reconciled tree $T_{\mathcal{R}}$ ($G$,$S$)
Result: $\sigma$ and $\tau$ layouts
1  read the leaf layer;
2  while the root has not been reached do
3      N← number of gene nodes in the current layer;
4      s← father species node in the current layer;
5      for $i \leftarrow 0$ to $N-1$ do
6          if $node_i.children \in current\ layer$ then
7              add them to sigma and tau and set them as siblings;
8          else
9              $\sigma_i$ = node$_i$;
10             p = find(node$_i$, s.leftSon, s.rightSon);
11             $\tau_p$ = node$_i$ ;
12         end
13     end
14     move up one layer
15 end
```

Algorithm 1: Build σ and τ

Layout	1	2	3	4	5
σ	1	2	3	4	5
τ	1	4	2	3	5

Table 1: σ and τ for the layer shown in Figure 3.2. Column i represents the *i-th* position in the father species node.

In order to determine whether it is better to rotate the left and right children for a given gene node, a supplementary double pointer is needed: *reversedTau* allows to compare the current layout and the rotated layout by using the property on the number of crossing stated in [1]:

Definition 7. [**Crossing equivalence**] Let $\bar{\tau}$ denote the linear order obtained by reversing τ. Then, for any $v \in V(S)$, we must have

$$cr(\bar{\sigma_v}, \tau, I(S,T), v) = cr(\sigma, \bar{\tau}, I(S,T), v)$$

3.2.2 Additional binary tree

By comparing the number of crossings brought about by each layout, we can determine whether it is better to perform the rotation or not. In order to use the property stated in Definition 7 we need to have the nodes in the layout σ organized in an additional binary tree. They already belong to a binary tree structure (the gene tree), but we cannot use it for two reasons:

```
Data: $\sigma$ layout
Result: additional binary tree for the nodes in sigma
$M \leftarrow$ sigma.size;
$U \leftarrow \emptyset$;
for $i \leftarrow 0$ to sigma.size do
    U.add(sigma$_i$)
end
while $U \neq \emptyset$ do
    for $i \leftarrow 0$ to U.size do
        if $\sigma_i$ and $\sigma_{i+1}$ have not been set as brothers then
            $\sigma_i$.sibling = sigma$_{i+1}$;
            $\sigma_{i+1}$.sibling = sigma$_i$;
            create new node $n$;
            $\sigma_i$.parent = $n$;
            $\sigma_{i+1}$.parent = $n$;
        end
        U.add($\sigma_i$.parent);
        U.remove($\sigma_i$) ;
        U.remove($\sigma_{i+1}$) ;
        if $\sigma_{i+2}$ is last then
            $i \leftarrow$ U.size
        end
    end
end
```

Algorithm 2: Build an additional binary tree

- we do not want to affect nodes of other layers when we perform a rotation
- determining the common parent of two nodes may require to visit all the gene tree

Algorithm 2 performs the construction of the additional binary tree for the actual layer. The result of Algorithm 2 applied to the graph of Figure 3.2 is shown in Figure 3.3.

3.2.3 Computing the number of crossings

In order to compare the number of crossings in the regular layout layer and in the rotated layer two auxiliary vectors are created: *rotatedTau*, which store the rotated version of *tau* and *revRotTau*, that is the reversed version of *rotatedTau*.

The algorithm then determines for every species node whether rotating the two species children is the best choice to make, by the comparison of the number of crossings the two choices imply. In order to compute the number of crossings,

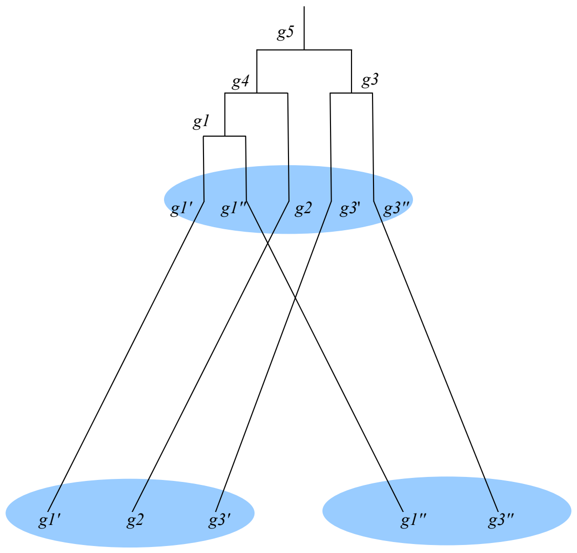


Figure 3.3: Modified instance, where an additional tree structure has been built

we need to let the algorithm undertake a preliminary phase where no tree is modified, since a modification at this stage would force the choice over which layout to choose. We then determine the better option for the layout of the species tree in the current layer. We then use it to optimize the layout of the embedded gene edges, having made the assumption that if we start from a better initial point we will end up with a better solution.

3.2.4 Adaptation of the GTL algorithm

Every node is stored together with its position in the layout, making him self-aware of how many nodes are on its left and on its right.

Let us recall that$L(\mathcal{T})$ indicate the set of leaves that occurs in the subtree rooted at $\mathcal{T}$.

Starting from the father of the left most node in the layout σ we climb the tree by choosing at every iteration the father of the current node y, until the root is reached. At every iteration we consider all the leaves that belongs to $L(r)$, where r is the right son of y.

Let z be the the largest leaf label in $L(l)$, where l is the left son of y. It is seen that every node with a label smaller than z has to belong to $L(l)$.

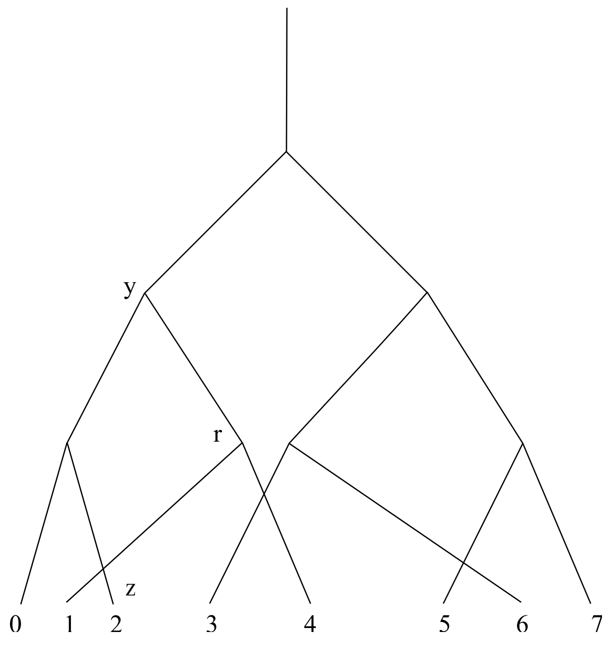


Figure 3.4: Example of a layout

The algorithm goes through all the possibles z, counting the numbers of nodes that appear after the node x in the linear layout τ, having label smaller or equal to z. Every such an event represents a crossing and is therefore added to the counter.

Once that information is collected, the algorithm establishes whether a rotation of that node could reduce the number of crossings. So, using the property stated in the Definition 7 in this chapter, we can decide whether a gene node rotation would prove useful or not by comparing the crossings of the regular layout with the crossings of the rotated layout. For every considered internal node z two counters are hold: *sum* and *reversedSum*, which keep track, respectively, of the crossings induced by all its children in the regular and reversed layout. Then those numbers are compared: if *reversedSum* is greater than *sum* for an internal node v, then an internal node rotation would improve the quality of the layout and is therefore performed. Such an operation requires two step:

a. a swap of the left and right son of the internal node in the additional structure

b. the rotation of $T(v)$ in the sigma layout.

The described procedure is operated recursively layer after layer, until the root is reached.

3.2.5 Carrying out the rotation in the gene tree structure

Due to the usage of the *plotutils* library, in the code every gene node has its own vertical and horizontal coordinates as well as a list of "extra" coordinates, which represents the evolution of the gene down the species tree, until a speciation or a duplication event happens. Every gene node in the reconciled tree can be considered in the program as an instance of the Node class or just as a set of coordinates.

Hence, when exchanging two nodes inside a layout this aspect needs to be taken into account in order to not interfere with the natural functioning of the output library, and the form of the node to be swapped has to be chosen accordingly.

To accomplish this the class that defines a *Node* has been modified and a struct *valueInfo* has been added. This struct is responsible for keeping track of the change of position inside the mapping the node may be subject to. Everytime the node needs to be moved a new item is added to the map, specifying the type of node it will have to replace.

Before invoking plotutils to have the print operations performed, there is the need of a phase where these swaps take place. This is done along a tree exploration performed by primeTV, so it does not require almost any additional time.

4 Performance analysis

In this chapter, we report the results achieved by the algorithm in terms of the quality of the solution.

4.1 Methodology

As it may be expected by an heuristic algorithm, the quality of the outputs PrimeTV gives vary according to the input it receives.

The auxiliary structure is build coupling the nodes pairwise from left to right (as shown in 3.2.2). The way this tree is built affects directly the quality of the rotations the algorithm is able to perform. In every layer a comparison between the number of crossings of the original layout and of the optimized one is made. The algorithm decides then to perform the changes that bring an improvement on the quality of the solution. However, the minimization of the number of crossings in a layer is seen as a process independent from the solution of the problem in the other layers. Therefore every iteration is not aware of the effects its decision will have on the following ones. For this very reason the algorithm could be deceived by a local optima and discard the best solution, preferring a non optimal one.

Anyhow we will see how this will not affect greatly the overall behavior of the algorithm and how, in the general case, the attempt of the algorithm to reduce the number of crossings is effective.

4.2 Tests

The tests have been performed in two rounds. The first set of tests used real species trees and the second set used random generated species trees.

4.2.1 Test with real species trees

In the first part of the tests 10 reconciled trees have been generated for each of the 6 different species trees. This lead to 60 different trees, that have been sorted by the number of gene nodes in the root of the species tree. In Table 2 the different classes used for the first set of tests are outlined. In every test the algorithm keeps track of the number of crossings the original layout produces and of the crossings produced by the optimized version.

Class	Number of nodes in the root	Number of reconciled trees
Small	5-7	20
Medium	8-11	20
Big	12-14	20

Table 2: Different classes of reconciled trees from real species trees used in the first type of tests

Table 3 shows how the algorithm performed with real species trees. It can be seen that the number of optimized instances is not greatly affected by the size of the input and that the heuristic improves the quality of the layout in the 55-60 percent of the cases.

Class	Optimized	Non-optimized	Optimized %	Unchanged %
Small	12	8	60	25
Medium	11	9	55	30
Big	11	9	55	15

Table 3: Number of instances where the algorithm improves the quality of the solution in the first set of tests with the percentage of improved and unchanged solutions (out of the total)

The average improvement of the solution for every class has been tracked, together with the total number of crossings removed. The obtained results are illustrated in Table 4 and show that the algorithm performs better when the size of the input is small or large.

Class	Optimized instances	Crossings removed	Improvement %
Small	12	186	32.41
Medium	11	182	16.12
Big	11	715	39.82

Table 4: Number of total crossings removed and average improvement by class in the first set of tests

4.2.2 Test with generated species trees

In the second set of tests 10 different species trees have been generated. Then for each of them 60 different reconciliations have been created. This led to 600 different reconciled trees, that have been sorted by the number of gene nodes in the root of the species tree. In Table 5 the different classes used for the second round of tests are described. The classes are the same used in the first round, but the number of tested trees is increased by a ten factor. In every test the algorithm keeps track of the number of crossings the original layout produces and of the crossings produced by the optimized version.

Table 6 shows how the algorithm performs with generated species trees. The results achieved by the algorithm are steady when the size of the input changes. The number and percentage of improved solution are illustrated, together with the number of non-optimized solution and the percentage of unchanged solutions (out of the total).

Class	Number of nodes in the root	Number of reconciled trees
Small	5-7	200
Medium	8-11	200
Big	12-14	200

Table 5: Different classes of reconciled trees from generated species trees used for the second type of tests

Class	Optimized	Non-optimized	Optimized %	Unchanged %
Small	147	53	73	12
Medium	147	53	73	6
Big	137	63	68	1

Table 6: Number of instances where the algorithm improves the quality of the solution in the second round of tests with the percentage of improved and unchanged solutions

The total numbers of crossings removed in every class has been calculated. Furthermore the improvement of the solutions in the optimized instances has been averaged by class. Those results are shown in Table 7.

Class	Optimized	Crossings	Improvement %
Small	147	1715	24.14
Medium	147	3150	14.68
Big	137	3438	14.96

Table 7: Number of total crossings removed and average improvement by class in the second set of tests

The algorithm achieves good results in all the tested configurations. It should be noted that the algorithm provides solution for a wider range of inputs when is provided with random data, but the improvements are greater when the data are not computer generated. This may suggest that the heuristic is suited to work with true biological data.

5 Conclusions

In this thesis we have proposed an heuristic that minimizes the number of crossings in the layout of a reconciled tree. The algorithm has been implemented and integrated in an active project aiming to improve the quality of its result.

We have also evaluated the quality of this heuristic, by comparing the number of crossings it produces with the number yielded by the original program. It has been shown that the algorithm is able to produce fewer crossings than the original version of the program. In addition it has been tested that the algorithm has good performances both with real biological data and with random generated data.

However, there is still room for improvement. In the reconciled tree there are many independent layers, that could be solved in parallel. In fact, the limited number of data that needs to be exchanged between layers makes the algorithm interesting for parallelization.

References

[1] Mukul S. Bansal, Wen-Chieh Chang, Oliver Eulenstein, and David Fernández-Baca. Generalized binary tanglegrams: Algorithms and applications. In *Proceedings of the 1st International Conference on Bioinformatics and Computational Biology*, BICoB '09, pages 114–125, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-00726-2. doi: 10.1007/978-3-642-00727-9_13. URL `http://dx.doi.org/10.1007/978-3-642-00727-9_13`.

[2] Jeff J. Doyle. Gene Trees and Species Trees: Molecular Systematics as One-Character Taxonomy. *Systematic Botany*, 17(1):144–163, 1992. ISSN 03636445. doi: 10.2307/2419070. URL `http://dx.doi.org/10.2307/2419070`.

[3] O. Eulenstein, B. Mirkin, and M. Vingron. Duplication-based measures of difference between gene and species trees. *J Comput Biol*, 5(1):135–148, 1998. ISSN 1066-5277. URL `http://view.ncbi.nlm.nih.gov/pubmed/9541877`.

[4] J. Felsenstein. Inferring phylogenies from protein sequences by parsimony, distance, and likelihood methods. *Methods in Enzymology*, 266:418–427, 1996.

[5] Pawel Górecki. Reconciliation problems for duplication, loss and horizontal gene transfer. In *Proceedings of the eighth annual international conference on Resaerch in computational molecular biology*, RECOMB '04, pages 316–325, New York, NY, USA, 2004. ACM. ISBN 1-58113-755-9. doi: 10.1145/974614.974656. URL `http://doi.acm.org/10.1145/974614.974656`.

[6] Wayne Maddison. Gene Trees in Species Trees. *Systematic Biology*, 46(3):523–536, 1997. doi: 10.2307/2413694. URL `http://dx.doi.org/10.2307/2413694`.

[7] Gary Olsen. Gary olsen interpretation of the newick tree format standard, August 1990. `http://evolution.genetics.washington.edu/phylip/newick_doc.html`.

[8] Roderic D. M. Page. Maps between trees and cladistic analysis of historical associations among genes, organisms, and areas, 1994.

[9] Nick Tufillaro Robert Maier. The plotutils package, October 2011. `http://www.gnu.org/software/plotutils/`.

[10] Bengt Sennblad, Eva Schreil, Ann Charlotte Berglund Sonnhammer, Jens Lagergren, and Lars Arvestad. primetv: a viewer for reconciled trees. *BMC Bioinformatics*, 8(1):148+, May 2007. ISSN 1471-2105. doi: 10.1186/1471-2105-8-148. URL `http://dx.doi.org/10.1186/1471-2105-8-148`.

[11] Michael Syvanen. Cross-species gene transfer; implications for a new theory of evolution. *J Theor Biol*, 112:333–343, 1985.

[12] Andreas Wotzlaw, Ewald Speckenmeyer, and Stefan Porschen. Generalized k-ary tanglegrams on level graphs: A satisfiability-based approach and its evaluation. *Discrete Applied Mathematics*, 160(16-17): 2349–2363, 2012. URL `http://dblp.uni-trier.de/db/journals/dam/dam160.html#WotzlawSP12`.