UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

CENTRO DE FÍSICA APLICADA Y TECNOLOGÍA AVANZADA

ESTUDIO COMPUTACIONAL DE ESCENARIOS EVOLUTIVOS

T E S I S

Para obtener el grado de:

LICENCIADO EN TECNOLOGÍA

PRESENTA:

ALITZEL LÓPEZ SÁNCHEZ

Directora de Tesis:

DRA. MARIBEL HERNÁNDEZ ROSALES

Instituto de Matemáticas, UNAM

JURIQUILLA, QUERÉTARO MARZO, 2019

Estudio Computacional de Escenarios Evolutivos

Alitzel López Sánchez

Resumen

En el contexto de biología molecular y evolución, *homología* se define como una relación entre un par de genes que comparten un ancestro común. Se considera que es la relación más general que existe entre genes evolutivamente relacionados. Existen diferentes relaciones que se derivan de ella, las cuales fueron establecidas por W.M. Fitch [1], entre las cuales podemos encontrar: ortología, paralogía y xenología. **Ortología** es una relación que se define entre un par de genes homólogos, donde ambos genes han surgido a través de un evento de *especiación*. Por otra parte, **paralogía** es una relación sobre un par de genes homólogos que surge por medio de un evento de *duplicación*. Finalmente, **xenología** es una relación que existe entre un par de genes homólogos donde uno se ha transferido horizontalmente entre especies. Con el paso del tiempo, este tipo de relaciones son difíciles de conocer, y su inferencia muchas veces resulta errónea debido a fenómenos genéticos como la pérdida de genes.

En esta contribución, se presenta una herramienta computacional para auxiliar en la caracterización matemática de las relaciones de ortología encontradas en la historia evolutiva de familias de genes. Un enfoque para el estudio de este tipo de relaciones es mediante grafos, donde los vértices representan genes y las aristas las relaciones de ortología. En este trabajo se proporciona un ambiente controlado donde las relaciones evolutivas son conocidas que además es capaz de generar los grafos correspondientes a cada historia evolutiva.

Computational Study of Evolutionary Scenarios

Alitzel López Sánchez

Abstract

In molecular biology and evolution, *homology* is defined as the relationship between a pair of genes that share a common ancestor. It is considered to be the most general relation between evolutionarily related genes. There are subsets deriving from it, that were described by W.M. Fitch [1] among which we can find: orthology, paralogy and xenology. **Orthology** is a relationship between two homologous genes, where both genes emerged through a *speciation* event. Whereas, **paralogy** is a relationship between a pair of genes that emerged through a *duplication* event. Finally, **xenology** is the relationship between a pair of homologous genes in which one has been "horizontally" transferred to another specie. Through the natural course of time, the evidence of ancient homology relationships cannot be found, and its inference most of the times is done erroneously due to certain genetic phenomena, for example, gene losses.
In this contribution, we present a computational tool to aid in the mathematical characterization of the orthology relations found in the evolutionary history of a set of genes. An approach to study these kind of relations is through a graph, where vertices represent extant genes and edges the orthology relations. In this work, we have implemented a controlled environment where the evolutionary relationships are known and that allows the generation of the representative graphs that belong to each evolutionary history.

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

CENTRO DE FÍSICA APLICADA Y TECNOLOGÍA AVANZADA
FACULTAD DE ESTUDIOS SUPERIORES, CUAUTITLÁN

LICENCIATURA EN TECNOLOGÍA

Votos Aprobatorios

COMITÉ ACADÉMICO DE LA
LICENCIATURA EN TECNOLOGÍA

Presente

En cumplimiento del Artículo 26 del Reglamento General de Exámenes, nos permitimos comunicar a usted que revisamos la Tesis de título

Estudio Computacional de Escenarios Evolutivos

que realizó el (la) pasante

Alitzel López Sánchez

con número de cuenta: 311031469, bajo la opción de titulación por Tesis y Examen profesional en la Licenciatura en Tecnología.

Considerando que dicho trabajo reúne los requisitos necesarios para ser discutido en el EXAMEN PROFESIONAL correspondiente, otorgamos nuestro **VOTO APROBATORIO.**

	NOMBRE	**FIRMA**
PRESIDENTE	Dra. Beatriz Marcela Millán Malo	
VOCAL	Dra. Adriana Hansberg Pastor	
SECRETARIO	Dra. Lucía Guadalupe Morales Reyes	
1er. SUPLENTE	Dra. Maribel Hernández Rosales	
2º SUPLENTE	Dr. Peter Florian Stadler	

Atentamente
"POR MI RAZA HABLARÁ EL ESPÍRITU"

UNAM, Campus Juriquilla, Qro. a 05 de marzo de 2019.

En memoria de mi padre,
que vive por siempre en mis recuerdos
y que aún así sigue inspirándome todos los días,
escuchando en silencio todos mis logros y aspiraciones.
Gracias por el tiempo que compartiste conmigo.

“If we wait until we’re ready,
we’ll be waiting for the
rest of our lives.”

- Lemony Snicket
The Ersatz Elevator

Acknowledgements

" In the middle of the journey of our life
I came to myself within a dark wood
where the straight way was lost.
Ah, how hard a thing it is to tell
what a wild, and rough, and stubborn wood this was,
which in my thought renews the fear!"
-Dante Alighieri
The Divine Comedy

First of all, I would like to thank my thesis supervisor, Dr. Maribel Hernández for inviting me to participate in this project. Joining the recently created bioinformatics group has been an academic experience that encouraged me to pursue the path of doing research in this area.

My deepest appreciation goes to prof. Peter F. Stadler, not only for participating in my thesis committee (coming all the way from Germany for my thesis defense) but also for sharing his insightful feedback and beer.

Many thanks also to the Leipzig team: Manuela Geiß, who has been extraordinarily tolerant and supportive throughout the development of this project, as well as Jens Steuck, Steve Hoffmann and Ali Yazbeck.

Special thanks to my guides through Hell and Purgatory: Dr. Rafael Quintero Torres, with his amazing book recommendations and self-study encouragement style, and my thesis committee, from which I also experienced some of the most enjoyable classes: Dr. Beatriz Millán, Dr. Adriana Hansberg and Dr. Lucía Morales.

I would also like to thank to the mexican Consejo Nacional de Ciencia y Tecnología (CONACyT,278966 FONCICYT 2), the German Academic Exchange Service (DAAD PROALMEX grant no. 57274200) for making this project possible.

This work received support also from Carlos González of the Instituto de Matemáticas UNAM as well as Luis Aguilar, Alejandro de León, Carlos S. Flores, and Jair García of the Laboratorio Nacional de Visualización Científica Avanzada (LAVIS) UNAM.

Discussions with Nancy González, with her constant mathematical denying of reality, have made work and learning in general a joyful experience.

I owe my deepest gratitude to my friends: Fernanda Hernández, Octavio Rodríguez, Maritere Domínguez, Edgar Chávez, Ariel Cerón and José Antonio Ramírez with whom I shared some of the most important academic and personal moments of my life.

I would like to thank my family, especially my mother and sister for all the non-stopping support and encouragement throughout these years and for all the shared calls, hugs, laughs and tears that came out of this process. Last, and by no means least, I would like to thank Marcos Laffitte for walking through this wild, rough, and stubborn wood with me from beginning to end.

Contents

List of Figures

Chapter 1

Introduction

Interdisciplinary research has been having an ever increasing popularity since the last century. Disciplines like mechathronics, biophysical chemistry and bioinformatics have been emerging through worldwide conferences and scientific journals all around the scientific community. As a bachelor in technology, solving the multidisciplinary problems presented by these ubiquitous disciplines is not only of great interest but also achievable within the basis of our formation.

All of these areas are being shaped by the most recent technological advances, which have flooded databases with enormous quantities of information. Each area therefore needs a conceptual basis to assist in the organization of this information as well as to formulate new questions and theories.

This contribution lies within the realm of Bioinformatics, particularly in the field that covers the mathematical study of Evolutionary Biology. As we will see in the following sections, the use of mathematical tools such as statistics, probability and graph theory is not new in biology, nevertheless it is somehow hidden behind the overwhelming availability of experimental data, which most of the time leads us to forget the need of looking new scientific theories. There is a quote in [2] where the authors portray the need of theoretical foundations in biology in this "big data" era:

> The current absence of a strong theoretical foundation in biology means that there is weak guidance regarding what quantities or variables need to be understood to best inform a general understanding (an explanatory basis) for biological features of interest. An unfortunate result of the absence of theory is that some researchers confuse just having data with "understanding".
>
> For example there is a base for collecting and analyzing the most microscopic data: experimental procedures and measurements in a high-throughput transcriptomics study are built around the assumption that transcripts are the primary data to be explained, and in neuroscience, recording from numerous individual neurons. This bias reflects a rather naive belief that the most fundamental data provide a form of explanation for a system, as if enumerating the fundamental particles where equivalent to the standard model in physics...

In this sense, several theories have been proposed to infer evolutionary history of sets of genes, in particular orthology relations. Some focus on the reconciliation of species and gene trees while others use the aid of graph theory. A common strategy used to reconstruct species phylogenies is to use orthologous genes as "evolutionary markers". Since these genes follow species divergence, they can pinpoint how branching in a species tree happens through time.

A common technique in graph-based methods for orthology identification is to use reciprocal best matches as a first step. The basic idea of this heuristic is that if a gene a in species A is orthologous to a gene b in species B then a and b are reciprocally most closely related. However, the theoretical basis of reciprocal best matches have rarely been studied directly until very recently [3].

With this aim in mind, in this work we present a computational tool to further explore evolutionary scenarios from a graph-based point of view. A common strategy used to generate evolutionary scenarios is to generate the evolutionary history of a gene family along a species tree that is generated by a stochastic model. An interesting stochastic model for the generation of species tree is the *innovation model*, which generates trees with branching structures akin to trees reported in phylogenetic databases. Since this model has remained widely unexplored, we undertook this model to serve as basis for the generation of evolutionary histories. Along the branches of our species tree, we generate the gene tree representing the evolutionary history of a single ancestral gene. By using these trees, we further proceed to build the theoretical orthology graph, which represents the true orthology relations of our scenario and its corresponding theoretical best match graph and reciprocal best match graph. All these advantages make our framework particularly valuable since it will serve as a benchmark to test methods that aim to reconstruct the evolutionary relationships between sets of genes by using a graph-based approach. We illustrate the utility of our simulation tool with an application case where we explore the effects of horizontal gene transfer in the theoretical generated graphs.

This thesis is organized as follows: Chapter 1 presents basic concepts regarding the biological and mathematical parts of this contribution.

In Chapter 2, a complete description of the state of the art regarding the main problem of homology distinction will be presented. We will also state how this contribution helps to solve a particular aspect of this problem.

The current methods, models and tools to generate evolutionary scenarios *in silico* will be reviewed in Chapter 3.

In Chapter 4 we present the necessary theory to understand the relations of interest that can be extracted from our generated evolutionary scenarios. Best Match Graphs were developed in collaboration with the Bioinformatics Group of the University of Leipzig, Germany and are described in a currently accepted manuscript pending of publication. [3]

In Chapter 5 we will present an implementation of a simple stepwise procedure to produce evolutionary scenarios with their corresponding True Orthology Graphs, Best Match Graphs and Recip-

rocal Best Match Graphs. The usefulness of this procedure will be illustrated with an applied case, involving the edge edition of the Reciprocal Best Match Graph.

Finally, in the subsequent chapters, we will discuss the obtained results, and offer some concluding remarks, that will lead us to the new possible applications of these results.

Chapter 2

Basic Concepts and Definitions

Inheritance is a key concept to understand the vast diversity of living things and processes that surround us, and it has been the focus of many interesting and challenging questions along the history of science as we know it.

2.1 DNA, Genes, Genomes and Evolution

In 1865, the Austrian friar Gregor Mendel gave a mathematical model of inheritance in which he postulated that the basic unit of inheritance was a **gene**. For many years, nobody took notice of his work, until the beginning of the 20th century, when it rose up again and it passed through intense mathematical development. Then, the gene was known to be made of DNA, which we now consider as the basis of heredity and refer to as the ”information coding molecule” i.e. DNA is the mean by which most organisms transfer genetic information to their descendants. The whole set of DNA molecules contained in an organism is called **genome**.

The basic flow of genetic information was proposed by Francis Crick in 1958, which is referred to as the *The Central Dogma of Molecular Biology.*

2.1.1 The Central Dogma of Molecular Biology

Genomes on its own are unable to release their own information to a living cell, they need to coordinate with other molecules: RNA and Proteins to finally release this information in a process called **gene expression**.

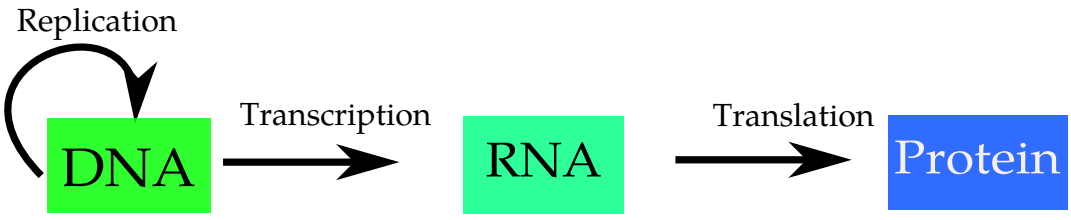


Figure 2.1: A schematic of the central dogma. It shows the basic flow of genetic information.

Each of these molecules can be seen as a word over a 4 letter alphabet (for DNA = $\{A,C,G,T\}$, for RNA = $\{A,C,G,U\}$), and for proteins as a word over an alphabet of 20 aminoacids. The central dogma's schematic's arrows as shown in fig. 2.1 indicate that we can make another molecule by using as a template the sequence of letters of an existing molecule by:

- **Replication:** In this process, DNA molecules can be copied.
- **Transcription:** In this process, a collection of RNA molecules are generated from certain genes of the genome called protein coding genes, this type of RNA is known as **messenger RNA (mRNA)**. In an organism, we can also find RNA molecules that do not come from protein coding genes. This whole set of RNA molecules is called the **Transcriptome**.
- **Translation:** In this process, from each mRNA molecule, a protein is created. The whole repertoire of proteins created from the Transcriptome is called **Proteome**. The Proteome dictates how and which biochemical reactions a cell is able to carry out.

2.1.2 Evolution

Evolution is a process by which populations of organisms change over generations. To understand the details of this process, we need to study **phylogenies**- the branching relationships of populations as they gave rise to new populations over time.

Evolution can be understood at different time scales: on a **grand scale**, it will show us the historical footprints of what led to major events in the tree of life, which is a model used in the exploration of life which describes the evolution of both living organisms and extinct ones. On a **smaller scale**, we can study phenomena such as the history of descents and relationships among species within a genus or even populations of the same species.

The basis of phylogenetic observations relies on traits displayed by organisms, which might give us hints about possible common ancestry. **Traits** are any characteristic an organism might have: from anatomical features to embryological processes and most commonly genetic sequences, such as genes.

The changes among populations are mostly due to genetic variation. These variations happen at different scales, from single letter changes to large chunks of letters, at different points in the life of an organism. When these changes alter the function of a gene or protein, they introduce new traits in an organism.

When the newly acquired trait is advantageous and helps an individual survive and reproduce, this variation is likely to be passed to the next generation of individuals. This process is known as **Natural Selection**. Then, in a bigger time scale, when the descendants of this individual propagate, the new trait becomes available over a population, making this population different from the ancestral one.

2.1.3 Homology and Analogy

In the vast diversity of living organisms, we can sometimes see that there are many similar traits shared among large groups of organisms. Some of these similarities are a consequence of shared ancestry. When a trait is found in two or more species because they shared a common ancestor, we call it a **homologous trait**. On the other hand, **analogous traits** are shared by two or more organisms but not because they had a common ancestor, they rather arose independently and were separately pushed by natural selection in each species. This process is known as **Convergent Evolution**.

Throughout this contribution, we will be using genetic material as the homologous traits upon our study. This aspect will be dealt with more detail in the next chapter.

2.2 Graph Theory

Graph Theory has a very wide range of applications in engineering, physics, biological sciences and in numerous other areas. In particular, graphs are the formal basis of the field of phylogenetics.

2.2.1 Graphs: Basic definitions and Properties

A **graph** G is an ordered pair (V,E) built by a set of elements called **vertices** $V = V(G)$ and a set of unordered pairs of vertices called **edges** $E = E(G)$. Graphs are usually depicted as diagrams where each vertex is represented by a dot, and each edge is represented by a line. For the purpose of this contribution, we will always assume **simple graphs**, which are graphs with no multiple edges and loops.

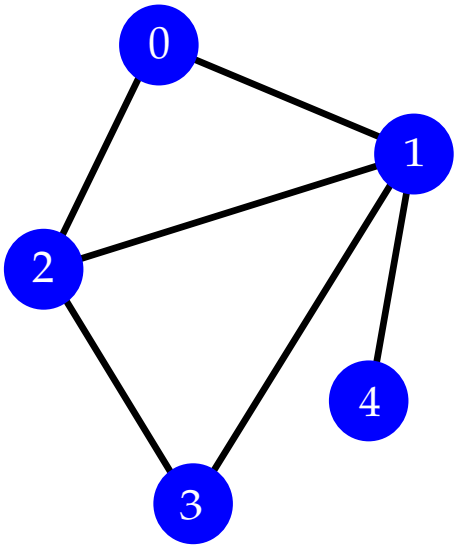
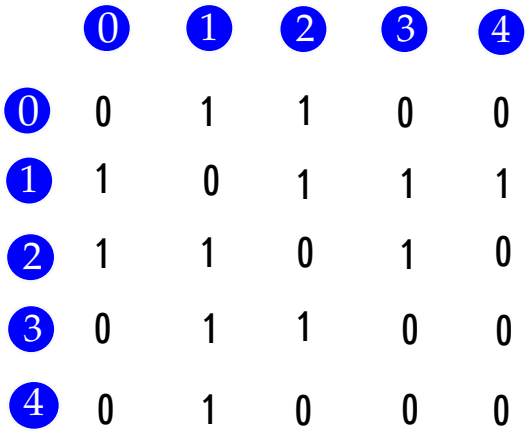


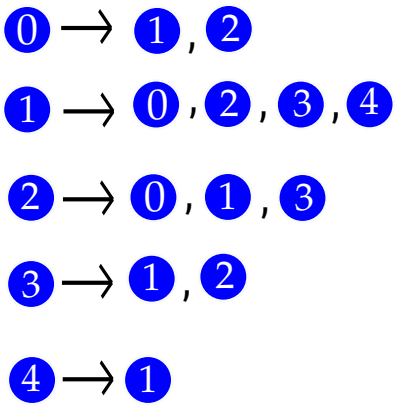
Figure 2.2: A graph **G** with vertex set $V(G) = \{0,1,2,3,4,\}$ and edge set $E(G) = \{\{0,1\},\{0,2\}$ $,\{2,3\},\{2,1\},\{3,1\},\{4,1\}$.

In a graph G, two vertices are **adjacent** if they are joined by an edge. Given $u \in V(G), \gamma(u)$ represents the set of vertices which are adjacent to vertex u, and $|\gamma(u)| = d(u)$ is called the **degree** of vertex u.

These concepts allow us to represent graphs computationally in two ways:



(a) Adjacency matrix representation of graph **G**.



(b) Adjacency list representation of graph **G**.

Figure 2.3: Computational Representations of a Graph G

Adjacency Matrix : For a graph with $|V|$ vertices, an adjacency matrix is a $|V| \times |V|$ matrix where, the entry in row i and column j is 1 if and only if the edge (i, j) is in the graph, else, we represent the absence of this edge with a 0. This representation of the graph in fig. 2.2 is depicted in fig. 2.3a.

Adjacency List : For each vertex in G, there is a set of adjacent vertices. Typically, we have an array of $|V|$ adjacency lists,one adjacency list per vertex. This representation of the graph in fig.2.2 is depicted in fig. 2.3b.

There are also certain types of sequences of adjacent vertices and edges of special importance:

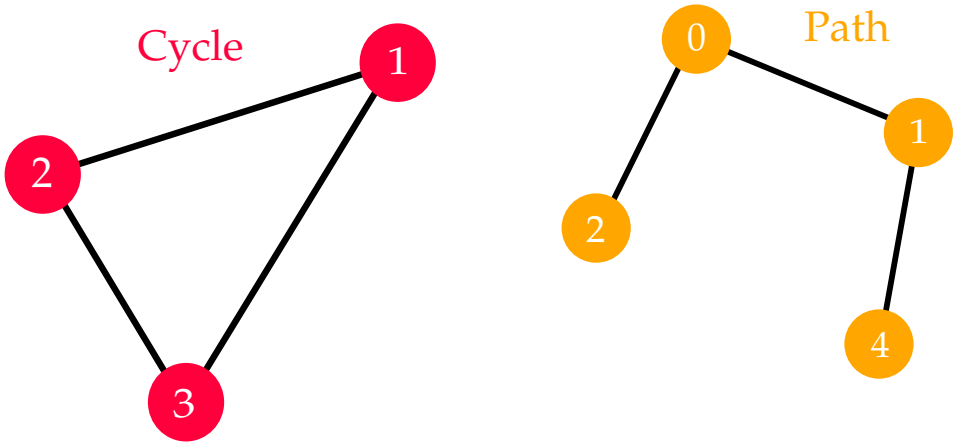


Figure 2.4: A cycle on 3 vertices and a P_4

Let G be a graph and v, w be vertices $\in G$. A **walk from** v **to** w is a finite sequence of adjacent vertices and edges of G:

$$ve_1v_1e_2...v_{n-1}e_nw$$

From here, the following concepts are derived:

- A **trail from** v **to** w is a *walk* from v to w that does not contain a repeated edge.
- A **path from** v **to** w is a *trail* that does not contain a repeated vertex. It is usually denoted as P_n, where n is the number of vertices in the path.
- A **closed walk** is a *walk* that starts and ends at the same vertex.
- A **circuit** is a *closed walk* that contains at least one edge and does not contain a repeated edge.
- A **cycle** is a *closed trail* where no other vertices are repeated apart from the start and end vertex.

Two vertices u and v of a graph G are said to be **connected** if there exists a path from u to v in G. A graph G is said to be **connected** when every pair of its vertices are connected.

In this contribution, the following graphs are of particular interest:

2.2.2 Complete Graph

A **complete graph** is a graph in which every pair of vertices is adjacent to each other. These graphs are usually denoted by K_n where n refers to $|V(K_n)|$. The complete graph on one vertex K_1, is also known as the **singleton graph**.

2.2.3 Induced Subgraph

An **induced subgraph** on a set of vertices $W = \{w_1, w_2, ..., w_k\}$ denoted by $G[W]$ is a graph that has W as its vertex set and its vertices preserve the adjacency of the original graph. An example is provided in the following figure:

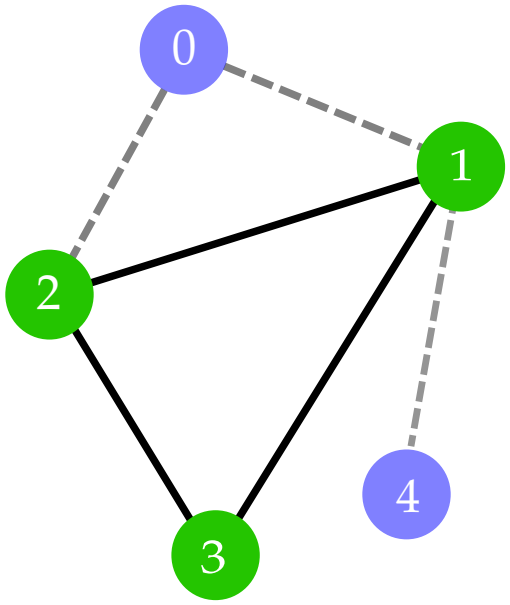


Figure 2.5: Let $W = \{1, 2, 3\}$, and let G be the original graph from 2.2. We show the induced subgraph $G[W]$ solid lines represent the edges of $G[W]$ while dashed arrows represent the remaining edges of G which are not in the edges set of $G[W]$.

2.2.4 Directed Graph

A **directed graph**, also called a *digraph*, is a graph in which its edges are called arcs and every arc xy has an initial vertex , called *tail* x, and a final vertex , called *head*, y.

A digraph $G = (V, E)$ is said to be **connected** if the underlying undirected graph is connected. For any pair of vertices $(u, v) \in V$ if there is a path from u to v, then v is said to be **reachable from** u. A digraph is **strongly connected** if all its pairs of vertices are *mutually reachable*.

For a vertex x in a digraph G we will write:

- $N(x) = \{z \mid xz \in E(G)\}$ for the **outneighborhood** of x
- $N^-(x) = \{z \mid zx \in E(G)\}$ for the **inneighborhood** of x

2.2.5 Trees

Trees are a special kind of graphs, since they are connected and have no cycles. According to their degree, we will distinguish the following types of vertices within a tree T as:

leaves: these are vertices of degree one.

inner vertices: these are vertices of degree > 1

We say a tree T is **rooted** when we designate an inner vertex to be identified as the **root** and we will denote this vertex as ρ_T. For the purposes of this work, all of our trees will be rooted.

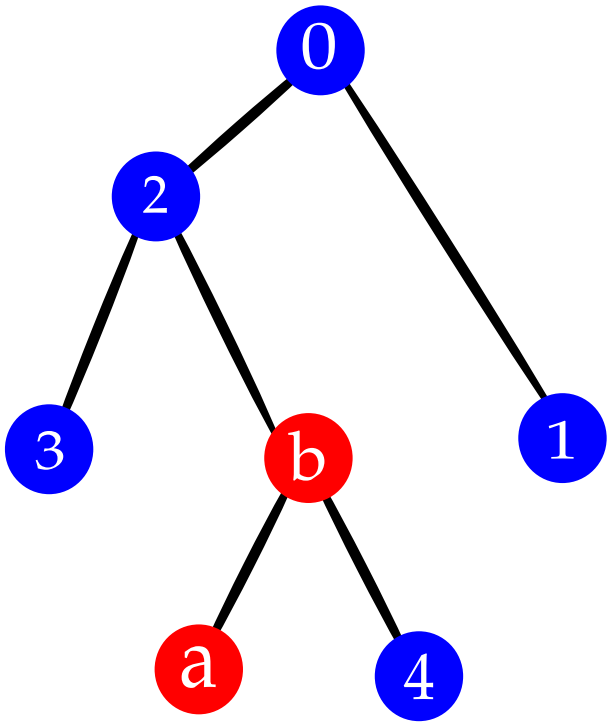


Figure 2.6: A binary Tree T. The vertices 0, 2 and b represent the *inner vertices* while a, 1, 3 and 4 represent its *leaves*. We choose the vertex 0 to be the *root* of T

A **k-ary tree** is a rooted tree, where the root has a degree at most k, and every other vertex has a degree at most $k+1$. When we have a $2-$ary tree, as in fig. 2.6 ,this tree is called a **binary tree.**

Computationally, a tree can be traversed in several ways [4]:

Preorder: we start by visiting the root, then the left subtree and finally the right subtree. This process is then applied recursively over the subtrees. In fig. 2.6, an ordering of the vertices of T by **Preorder traversal** will be $[0,2,3,b,a,4,1]$.

Inorder: we start by visiting the left subtree, then the root and finally the right subtree.This process is then applied recursively over the subtrees. In figure 2.6 an **Inorder traversal** of T will be done in the following order: $[3,2,a,b,4,0,1]$.

Postorder: here, we visit the left and right subtrees and the root is visited at the end. This process is then applied recursively over the subtrees. In fig. 2.6, the list of vertices of T visited by a **Postorder traversal** will be $[3,a,4,b,2,1,0]$.

A vertex a in a rooted tree, for example the one in fig. 2.6, is called a **descendant** of a vertex b if the path from a to the root passes through b, then we say that b is an **ancestor** of a. We can also define an **ancestor order** in a tree T by setting $x \prec_T y$, which means that y is a vertex that belongs to the unique path connecting x with the root of T, ρ_T.

Vertices that are adjacent to a vertex v and are descendants of v are called the **children** of v. An adjacent vertex that is an ancestor of some vertex u is called the **parent** of u.

The **Lowest Common Ancestor** (lca) of a set of vertices D is the unique ancestor of D that is a descendant of all the ancestors of D. Whenever there is a risk of confusion, we will use lca_T to denote the lca of a set of vertices in a particular tree T.

A binary tree on three leaves is called a **triple**. In particular, we write $xy|z$ for the triple on leaves x,y and z if the path between z and the root does not intersect the path from x to y.

An **event-labelled** tree, which will be denoted by (T,t), is a rooted tree T together with a map $t : V_i \to \mathbb{M}$ that assigns each inner vertex $v \in V_i$ an event $m \in \mathbb{M}$. This means that in an event-labelled tree where x, y belong to its leaf set, if we look for $lca(x,y)$ it will be marked with an event $t(lca(x,y)) = m$.

2.2.6 Phylogenetic Trees

Mathematically, a **Rooted Phylogenetic Tree** T on a set L is a rooted tree with leaf set L where no inner vertex has degree $= 2$. When we have a Rooted Phylogenetic Tree on a set of *Species* $\mathbb{S}$ we will call it **Species Tree** and when the leaf set represents a set of *Genes* $\mathbb{G}$, **Gene Tree**.

These definitions will constitute the formal basis of this study in the following way:

Species Tree $T_{\mathbb{S}}$: this is a mathematical model where the leaf set $\mathbb{S}$ represents extant species and the inner vertices represent ancestral species.

Gene Tree $T_{\mathbb{G}}$: in this model, we have an *event - labelled* tree in which the leaf set $\mathbb{G}$ represents extant genes and the event set will represent evolutionary events, which will be presented on the next chapter.

2.2.7 Reconciliation Map μ

When we want to mathematically establish a relationship between a gene tree $T_{\mathbb{G}}$ and a species trees $T_{\mathbb{S}}$ we can use a map $\sigma : \mathbb{G} \to \mathbb{S}$ to represent which genes in $\mathbb{G}$ belong to which species in $\mathbb{S}$. Moreover, we can extend this mapping so that it also maps the inner vertices of $T_{\mathbb{G}}$, i.e. the events of the gene tree into the vertices or edges of the species tree $T_{\mathbb{S}}$. This is called a **Reconciliation Map** μ, and it is defined as follows [5] :

Let $T_{\mathbb{S}} = (V_{\mathbb{S}}, E_{\mathbb{S}})$ and $T_{\mathbb{G}} = (V_{\mathbb{G}}, E_{\mathbb{G}})$ be two phylogenetic trees on $\mathbb{S}$ and $\mathbb{G}$, respectively, and let $\sigma : \mathbb{G} \to \mathbb{S}$ be a surjective map, i.e. for all $s \in \mathbb{S}$ there is a $g \in \mathbb{G}$ such that $\sigma(g) = s$. A **reconciliation from $T_{\mathbb{G}}$ to $T_{\mathbb{S}}$** is a map $\mu : V_{\mathbb{G}} \to V_{\mathbb{S}} \cup E_{\mathbb{S}}$ such that it satisfies the following constraints:

1. *Root Constraint*: $\mu(v) = \rho_{T_{\mathbb{S}}}$ if and only if $v = \rho_{T_{\mathbb{G}}}$.

2. *Leaf Constraint*: if $x \in \mathbb{G}$, then $\mu(x) = \sigma(x)$.

3. *Ancestor Preservation*: for $x, y \in V_{\mathbb{G}}$, $x \prec_{T_{\mathbb{G}}} y$ implies that $\mu(x) \prec_{T_{\mathbb{S}}} \mu(y)$.

4. *Speciation Constraints*: Suppose $\mu(x) \in V_{\mathbb{S}} \setminus \mathbb{S}$, i.e. $\mu(x)$ belongs to the inner vertices of $T_{\mathbb{S}}$.

   - $\mu(x) = lca_{T_{\mathbb{S}}}(\mu(v'), \mu(v''))$ for at least two distinct children v' and v'' of x in $T_{\mathbb{G}}$.
   - $\mu(v')$ and $\mu(v'')$ are incomparable in $T_{\mathbb{S}}$ for any two distinct children v and v'' of x in $T_{\mathbb{G}}$.

As has been previously reported in literature [6], it is always possible to reconcile a gene tree with a species tree. Furthermore, this reconciliation map completely determines an *event labeling* t on $T_{\mathbb{G}}$.

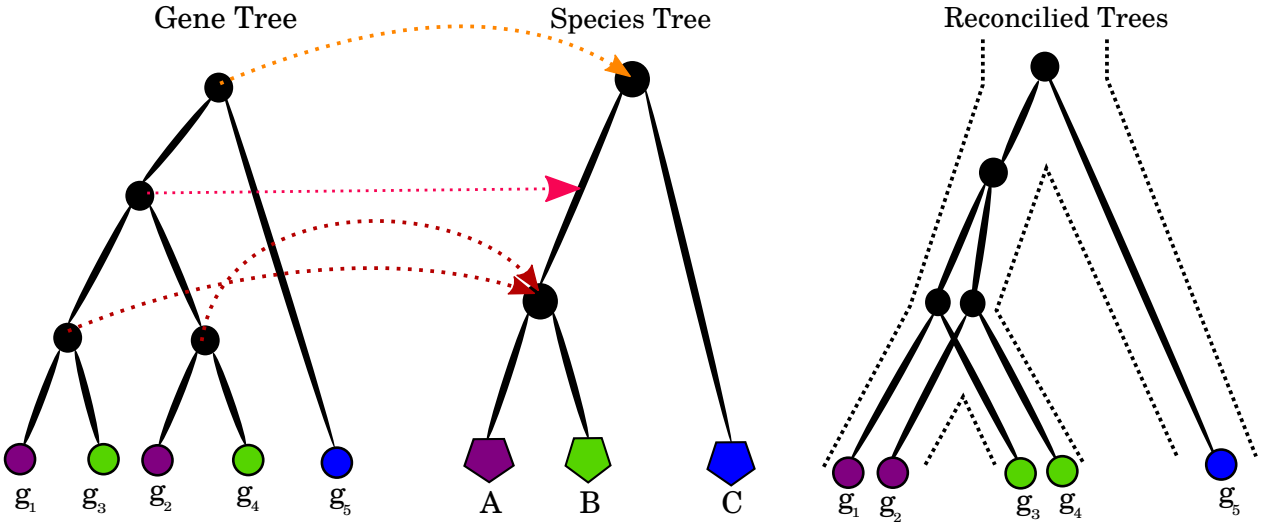


Figure 2.7: Reconciliation between a Gene Tree and a Species Tree. The dashed coloured arrows represent the mapping μ. To represent the *Leaf Constraint*, we use a colour to denote each species in $\mathbb{S}$. Reconciliation can also be seen as an "embedding" of the Gene Tree into the Species Tree, represented as a black dashed line in the third panel.

2.2.8 Cographs

Before looking at the definition of cograph, there are some notions we need to present beforehand:

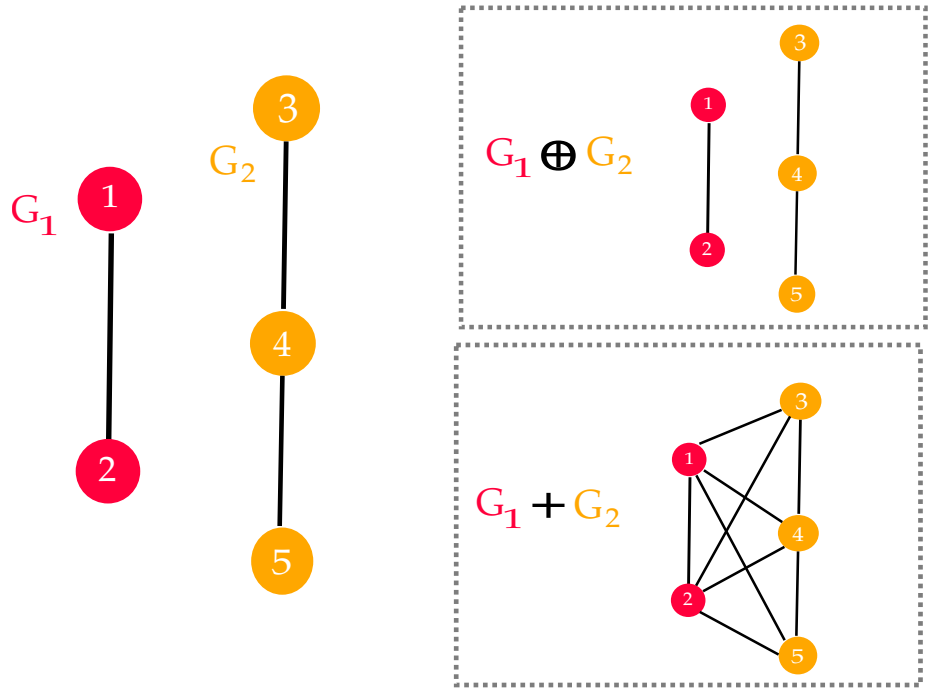


Figure 2.8: Two graphs G_1 and G_2, their **Disjoint Union** $G_1 \oplus G_2$ and their **Join** $G_1 + G_2$.

The **Complement** of a graph $G(V,E)$, usually denoted by $\overline{G}(\overline{V},\overline{E})$, is the graph with vertex set V and edge set $E(\overline{G}) = \{xy | x,y \in V, x \neq y, xy \notin E\}$.

Given two disjoint graphs, G_1 and G_2 their **Disjoint Union** $G_1 \oplus G_2$ is the graph with vertex set $V(G_1) \cup V(G_2)$ and edge set $E(G_1) \cup E(G_2)$, and the **Join** $G_1 + G_2$ is the graph with vertex set $V(G_1) \cup V(G_2)$ and edge set $E(G_1) \cup E(G_2) \cup \{uv | u \in V(G_1)$ and $v \in V(G_2)\}$.

The name **cograph** comes from *Complement Reducible Graphs*, as by definition cographs can be *reduced* by stepwise complementation of connected components to their corresponding K_1's. A growing body of literature has characterized cographs throughout the years in many different ways [7] but, for this contribution, we will focus on the following definition:

A graph G is a **Cograph** if any of the following equivalent conditions hold [8]:

- G can be constructed from K_1's by **disjoint union** and graph **join** operations.
- G does not contain P_4 as an induced subgraph.

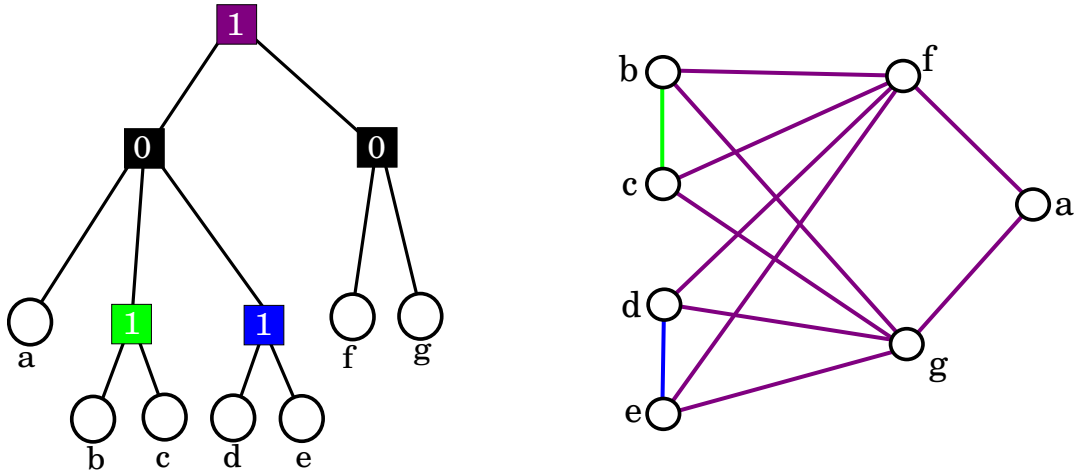


Figure 2.9: A Cograph C and its associated CoTree T_C. Each edge (v,w) in C has a matching color to that of the $lca(v,w)$ in T_C.

A cograph C has a particular tree associated with it, called **CoTree** T_C. It is a tree whose inner vertices are labelled with either 0 or 1 and the leaves are the vertices of C. If we take a look at the subtree rooted at a vertex $n \in V(T_C)$ with leaf set W, we will find that it corresponds to the the induced subgraph $C[W]$. To better understand this idea, in figure 2.9, each edge (v,w) of the cograph C has a matching color to that of the $lca(v,w)$ in its corresponding cotree.
Cographs are widely studied in computer science. Since they can be recognized in linear time, they have been used in algorithms for community detection in complex networks [9] and most importantly, as we will see in the next chapter, they are closely related to the concept of **orthology**.

2.3 Binary Relations and their Representations as Graphs and Trees

In order to understand Binary Relations, the following fundamental concepts of set theory are necessary:

2.3.1 Cartesian Product

Let a,b be elements of a set $\mathbb{X}$. An **ordered pair** formed by a and b is denoted by

$$(a,b) = \{\{a\},\{a,b\}\},$$

which is the formal way of saying that in this pair of elements, we can distinguish the first element from the second element:

$$(a,b) = (c,d) \iff a = c \text{ and } b = d \tag{2.1}$$

Let $\mathbb{A},\mathbb{B}$ be sets. The **Cartesian Product** of $\mathbb{A}$ and $\mathbb{B}$ is the set of ordered pairs where the first element belongs to $\mathbb{A}$ and the second element belongs to $\mathbb{B}$:

$$\mathbb{A} \times \mathbb{B} = \{(a,b) | a \in \mathbb{A} \text{ and } b \in \mathbb{B}\}$$

2.3.2 Relations

A **Relation R** between sets $\mathbb{A}$ and $\mathbb{B}$ is a subset of the Cartesian Product $\mathbb{A} \times \mathbb{B}$. Relations on sets in general can follow some important properties:

Let $\mathbf{R}$ be a relation on a set $\mathbb{A}$

1. $\mathbf{R}$ is *reflexive* if and only if, for all x in $\mathbb{A}$, $(x,x) \in \mathbf{R}$
2. $\mathbf{R}$ is *symmetric* if and only if, for all x and $y \in \mathbb{A}$, **if** $(x,y) \in \mathbf{R}$ then $(y,x) \in \mathbf{R}$
3. $\mathbf{R}$ is *transitive* if and only if, for all $x,y,z \in \mathbb{A}$, **if** $(x,y) \in \mathbf{R}$ and $(y,z) \in \mathbf{R}$ then $(x,z) \in \mathbf{R}$

Relations that satisfy these three properties are called **Equivalence Relations**. Suppose there is an equivalence relation on a certain set $\mathbb{A}$. If we take an element $a \in \mathbb{A}$ and ask for the subset of all elements related to a, this subset is called the **equivalence class** of element a.

2.3.3 Binary Relations

A **Binary Relation R** over a set $\mathbb{A}$ is a subset of $\mathbb{A} \times \mathbb{A}$.

2.3.4 Graph Representation of Binary Relations

We say that a graph $G_{\mathbf{R}}$ is a Representation of the binary relation $\mathbf{R}$ on the set $\mathbb{A}$ if and only if:

1. The vertex set of $G_{\mathbf{R}}$ is $\mathbb{A}$;
2. An edge connecting two vertices $a,b \in G_{\mathbf{R}}$ exists when $(a,b) \in \mathbf{R}$.

2.3.5 Tree Representation of Binary Relations

While graph representations of binary relations look straightforward and natural, tree representations of a binary relation $\mathbf{R}$ are tricky, since not all binary relations have a tree representation. The binary relations presented in this work will be represented by trees that satisfy that:

1. The leaf set of T is $\mathbb{A}$
2. T has a labelling such that it uniquely tells us whether $(x,y) \in \mathbf{R}$ or not.

Binary relations and their graph representations, as we will see in the next chapter, will be the main protagonists of this contribution, since we can use them to model the evolutionary relationships between genes.

2.4 Markov Chains: Basic Notions

Markov Chains are not only used in mimicking sequence evolution but also in several other areas. They typically describe a system which changes between different states through time, according to some random mechanism.

The first element of a Markov chain is a collection of states, called **state space** I. The represented system will be always in one of these states. Sometimes we will know what state the system is in and sometimes we will know that the system is in a state i with some probability. Hence, we can assign a probability measure, also known as a **probability distribution** λ to every element of the **state space**.

The mechanism by which a change of state happens is described by a **transition matrix** P with entries P_{ij} , $i, j \in I$. An entry p_{ij} will give us the the probability that the system changes from state i to j at the next time state. Also we need to specify at which state the system was at time $= 0$.

Let X_n denote the state of a system at time n. A Markov chain with initial distribution λ and transition matrix P is specified by:

1. X_0 has a distribution λ.
2. For all n and $i_0, i_1, \ldots, i_n \in I$ the probabilities $\mathbb{P}(X_0 = i_0, X_1 = i_1, \ldots X_n = i_n)$ that a system occupies the states $i_0, i_1, \ldots, i_n$ at times $0, 1, \ldots n$ is written as:

$$\mathbb{P}(X_0 = i_0, X_1 = i_1, \ldots X_n = i_n) = \lambda_{i_0} p_{i_0 i_1} \ldots p_{i_{n-1} i_n}$$

Markov chains, are essential mathematical tools to simulate the evolution of nucleotide sequences *in silico*, as we will see in chapter 4.

Chapter 3

Distinguishing among the different types of homology

Evolution is understood by building Evolutionary trees or phylogenies which are hypotheses about evolutionary relationships among organisms. The main framework in which these trees are built is:

1. Select a number of species for which we wish to build a tree;

2. Collect information about the traits of these species;

3. Look at which species have characters in common. The idea behind tree building is that species with many traits in common are more likely to be closely related between them than with any other species with fewer common traits.

The traits that will be addressed in the rest of this contribution will be genes. Genes are ubiquitous in the study of evolutionary relationships, as the rise of highthroughput sequencing technologies has made these kind of data accessible for many more organisms day by day. As we have seen in the previous chapter, by taking a molecular biology and evolutionary approach we will be using *homology* as a relationship between two genes that share a common ancestor.

3.1 Homology and its flavours

Homology and its subset relations were defined by Walter M. Fitch in his 1970's famous article entitled *'Distinguishing Homologous from Analogous Proteins'* [1]:

> Where the homology is the result of gene duplication so that both copies have descended side by side during the history of an organism, (for example alpha and beta hemoglobin) the genes should be called paralogous (para = parallel). Where the homology is the result of the speciation so that the history of that the history of the gene reflects the history of the species (for example alpha hemoglobin in man and mouse) the genes should be called orthologous (ortho = exact)...

This distinction between orthology and paralogy is important since both relationships are tightly related to the function of genes. In most cases, genes that are orthologous share the same functions while paralogs on the other hand, have related but clearly distinct functions. Therefore, the importance of these distinction falls upon the realms of genome annotation and it also conveys information regarding the species and gene tree. Nevertheless, in the rest of this contribution we will adopt an approach in which homology (and therefore all its flavours) refer only to the evolutionary history of a gene family and not to its function.

The third flavour of homology is *xenology*, which was also stated by Walter M. Fitch in [10] as:

> Xenology is defined as that condition (horizontal transfer) where the history of the gene involves an interspecies transfer of genetic material. It does not include transfer between organelles and the nucleus. It is the only form of homology in which the history has an episode where the descent is not from parent to offspring but rather from one organism to another...

Horizontal gene transfer is the non-sexual transmission of genetic material, which often involves gene transfer across species boundaries via mechanisms like **transformation**, **conjugation**, **transduction** or **introgression**[11].

Every flavour of homology plays a key role in the evolution of genomes. This has led to the wide diversity of gene families and gene functions. The resulting changes range from being in charge of the molecular machinery of the cell to helping the organisms adapt to their environment. Gene duplication, is frequently involved in the creation of novel genes. Since it produces copies of a gene, these are often the source of genetic differences which result in both genes having a different evolutionary fate that sometimes leads to the acquisition of novel functionalities [12]. Horizontal Gene Transfer, on the other hand, reshapes an organism's repertoire of genes by acquiring new genetic material from other species. This acquired material most of the time has some sort of beneficial characteristics associated to it, an example of this are genes that help bacteria become resistant against antibiotics. Moreover, the genes resulting from this kind of event, tend to have different evolutionary histories, which most of the time complicate research involving the evolutionary relatedness of species and gene trees.

To infer the different flavours of homology, several statistical and combinatorial methods have been released (see for example [13] for a complete review in horizontal gene transfer methods).
Using Fitch's definitions as basis, we will consider the so-called *lca*-**definitions of homology** in T as in [14]. Using the notion that:

Ortology: Two genes a, e are *lca-orthologous* in T if the $lca(a,e) =$ speciation event and $\sigma(a) \neq \sigma(e)$ i.e. a and e belong to different species.

Paralogy: Two genes a, b are *lca-paralogous* in T if the $lca(a,b) =$ duplication event.

Xenology: Two genes c, d are *lca-xenologous* in T if the $lca(c,d) =$ horizontal gene transfer event (HGT).

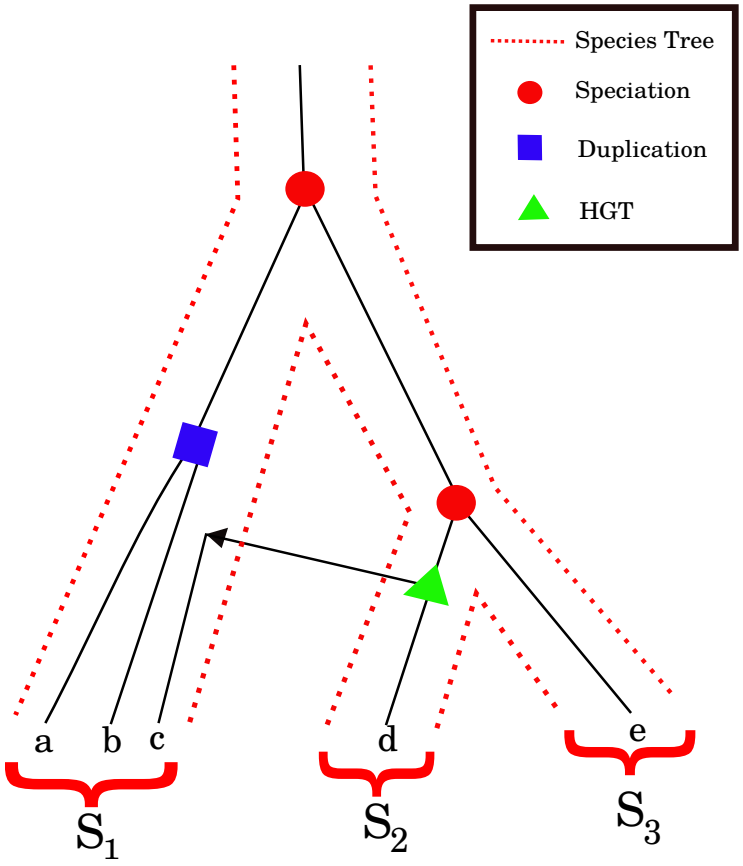


Figure 3.1: *lca*-definitions of the homology flavours. Genes a, e are *lca-orthologous* in T, a, b are *lca-paralogous* in T and c, d are *lca-xenologous* in T.

In this sense, we can study *gene families*, which involve all genes, in the set of genomes under consideration, that share an evolutionary history. The different flavours of homology can be studied through the evolutionary history of a set of genes using: a species tree, an underlying gene tree of a gene family and the mutual relation between these two. The first step towards the correct identification among the different flavours of homology is to know the orthology relations among the genes we are studying.

3.2 Current methods for Orthology inference

The importance of orthology inference is at the heart of various genomics studies, from comparative genomics to the reconstruction of ancestral proteomes. Most importantly, orthologs convey important information about the underlying species tree. These methods can be, according to [15], classified under two main categories:

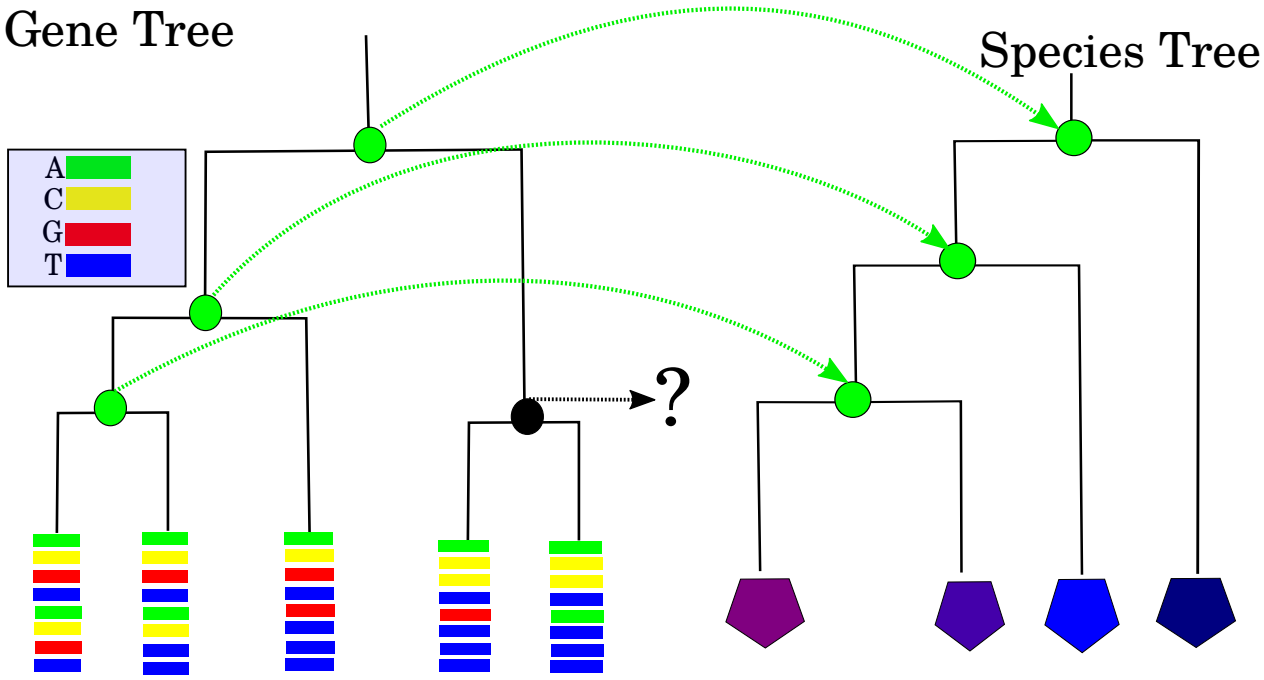


Figure 3.2: A small example of tree-based orthology inference methods. Groups of color bars represent different nucleotide sequences while different colour pentagons represent different species.

3.2.1 Tree-Based Methods

These methods rely on the **reconciliation** of a gene tree with a species tree. The gene trees used here are not event-labelled trees but rather trees constructed by the identification of genes across genomes that share a certain sequence similarity. In fig 3.2 we exemplify this process by using coloured rectangles which represent each nucleotide. The species tree is usually inferred from extant species combining the use of fossil evidence, which are available in most phylogenetic databases. This *reconciliation* process uses an "embedding" of the Gene Tree into the Species Tree. In fig. 3.2 we represent this as dashed green arrows that go from the gene tree to the species tree. Whenever an incongruence in this "embedding" takes place, it is interpreted as if the gene family involved has undergone events such as duplication, loss or HGTs [16]. In fig. 3.2 this is depicted as a black dashed arrow.

Some shortcomings of this methods are:

- This method assumes both trees are "correct". Nevertheless, there are several cases in which the species tree is unknown or when a few misplaced leaves lead to scenarios with significantly more duplications and losses. As an immediate consequence of this, this method works with selected gene families which have small rates of duplication events.

- The trees used for this kind of inference need to be rooted. It is possible to achieve a rooted tree by choosing an outgroup. Let $\mathbb{X}$ be the set of leaves of a tree T (gene or species tree), and let $X \subseteq \mathbb{X}$ be a set of leaves that represents either genes or species under consideration.

Consider a leaf $z \in \mathbb{X}$. We say z is an **outgroup** for X if $\texttt{lca}(X) \prec \texttt{lca}\ (X,z)$. The choice of outgroup is most of the time done manually, so automatization is limited in this approach.

- These methods have a high computational cost when dealing with large datasets.

Driven by the limitations of these methods, a new kind of methods for orthology inference emerged,not only for the need to improve the extant runtime capabilities but also because of the simple question: *Can we infer the phylogenetic relationships without having to build species and gene trees?*

3.2.2 Graph-Based Methods

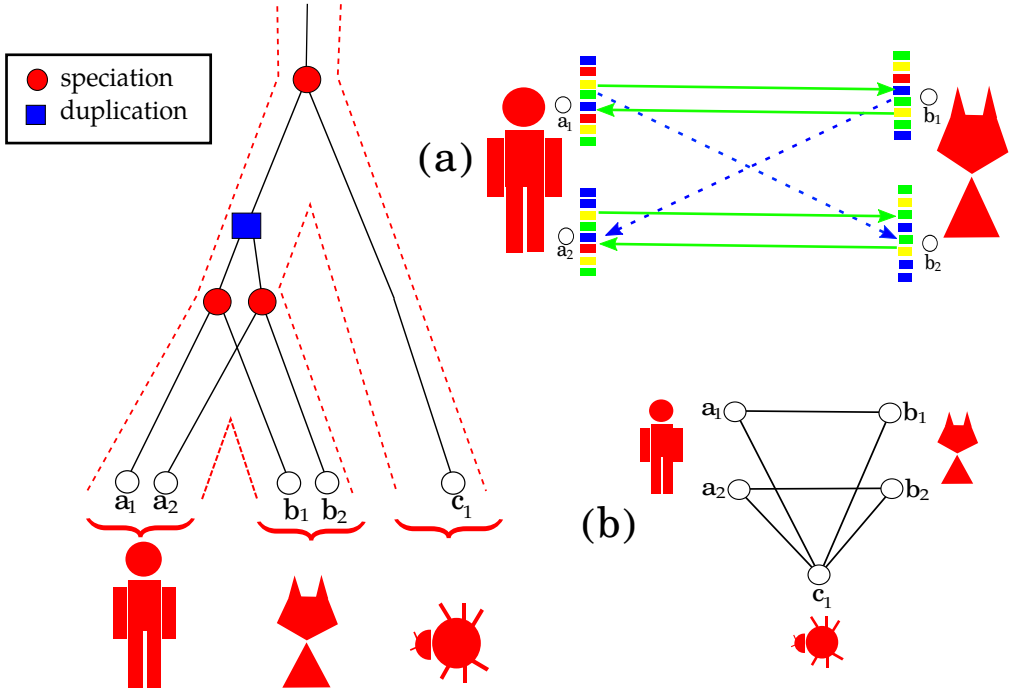


Figure 3.3: A small illustration of graph-based orthology inference methods. The left tree represents the evolutionary history of genes a_1, a_2, b_1, b_2 and c_1. We use three different species: A human, a cat and a bug.

In these methods, bypassing the need of building Species Trees and Gene Trees it is possible to know the orthology relations among genes by building a digraph whose vertices represent either genes or proteins and the arcs have a weigh that represents the confidence of the orthology relationships. These edges are inferred by pairwise sequence similarity and an operational definition of orthology. In fig. 3.3 (a) we illustrate the process of sequence comparison between the genes of the human and the cat. Each arrow represents a comparison. Solid green arrows symbolize arcs with higher confidence, and dashed blue arrows are arcs with smaller confidence. In fig. 3.3(b) we represent only the "highest confidence" bidirectional arrows among all species as an undirected graph, which represents the orthology relations among these genes. Some of these methods also use clustering techniques to further extend the orthology predictions to more than two species and

construct clusters of orthologous groups. We will give an insight into the *how it works* details in the following sections.

3.3 Limitations of current methods

The caveats of current orthology detection methods are mainly triggered by other evolutionary events, such as gene losses, or duplications [17] which are mostly the source of false positive predictions. As shown in the figure below:

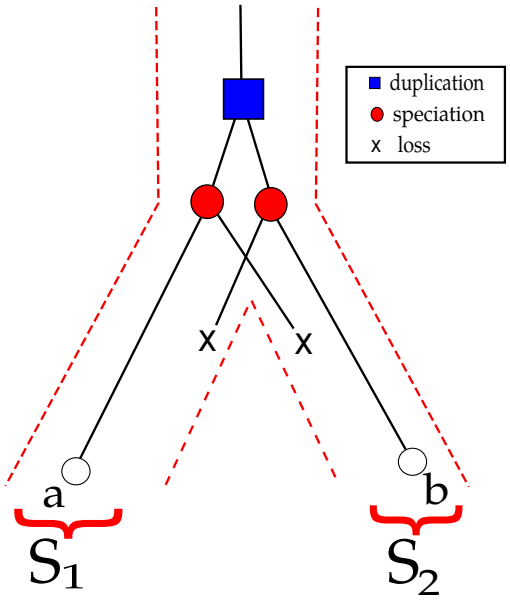


Figure 3.4: A popular small example of undetectable paralogy. Genes *a* and *b* are erroneously taken for *orthologous genes* by most orthology inference tools.

In fig. 3.4, we can see a small example of undetectable paralogy where genes *a* and *b* are erroneously taken for *orthologous genes* by most orthology inference tools.

To evidence these false predictions, the true evolutionary relationships must be known. Therefore, to asses the accuracy of new methods for orthology inference is the main topic of this contribution.

In this work, we will present a tool for simulating true evolutionary scenarios. We also present an applied study case which involves a formalization of the best match heuristic used in graph-based orthology detection methods. In this application, we also explore the effects of HGTs in the theoretical graphs involved in the generated evolutionary scenarios.

Chapter 4

Simulation of Species Trees and Gene Trees

Since Darwin published his work *On the origin of the species*, evolution has been acknowledged as a dynamical and complex process involving broad temporal and spatial scales as well as stochastic events. One important feature about evolution is its highly inhomogeneous rate of change in speciation and extinction events. With these ideas in mind, species trees presented in this section will be branching processes that mimic the evolution of species through time. This process is considered within the following framework:

4.1 Stochastic Models of Evolution

Evolution happens within points in time t, where there is a set of species $\mathbb{S}(t)$, which serves as a representation of the species that exist at this current point in time. Then, we will say that "evolution" happens when a species $s \in \mathbb{S}(t)$ chosen by a certain rule, which most of the times is a probability distribution $\pi(s,t)$ on $\mathbb{S}(t)$. We define a **speciation** as the replacement of a species s by two species s' and s'':

$$\mathbb{S}(t+1) = \mathbb{S}(t) \setminus \{s\} \cup \{s', s''\}$$

Evolution is then represented by a phylogenetic tree $T(t)$ on the set $\mathbb{S}(t)$ of species. At each time step t the leaves of $T(t)$ are the species in $\mathbb{S}(t)$. When a species s undergoes speciation, two new leaves are attached to the leaf s.

There are several Stochastic Models of evolution:

4.1.1 Yule Model

This model represents the simplest speciation case. The probability of a species to evolve is $\frac{1}{|\mathbb{S}(t)|}$, this means that any species in $\mathbb{S}(t)$ has the same probability of speciation and thus it belongs to a uniform distribution $\pi(s,t)$.

4.1.2 Activity Model

In this model, the set of species $\mathbb{S}(t)$ is partitioned into a set of active species $S_A(t)$ and a set of inactive species $S_I(t)$. At each time step, a species $s \in S_A(t)$ is drawn uniformly. When speciation takes place, the two new species s' and s'' independently enter either to the active set $S_A(t+1)$ with probability p or to the inactive set $S_I(t+1)$ with probability $p-1$. If the set of active species is empty, then a species is drawn from the set of inactive species $S_I(t)$ and the resulting s' and s'' enter the set of active species.

4.1.3 Birth-Death Process

This process was formalized by Kendall in 1948 [18]. It emerged as a problem of population growth where they represented the expected birth and death rates per head of population per unit of time. It starts with an initial species. A species gives *birth* to a new species after exponential waiting time, which is represented by rate λ and dies at an exponential waiting time, rate μ. There are several variants of this model:

- When $\mu = 0$ we get the Yule Model.
- When $\mu = \lambda$ this is called **Critical Branching Process** [19].
- When the process is conditioned to have n extant species it is called **Conditioned Birth Death Process** [19].

4.1.4 The Innovation Model

In this model, proposed in [20], each species $s \in \mathbb{S}(t)$ represents a finite set of features: $s = \{\varphi_1, \varphi_2, \ldots, \varphi_n\}$. These features are represented as integer numbers. The set of all extant features at time t is denoted by $\mathbb{F}(t)$. Each speciation

$$\mathbb{S}(t+1) = \mathbb{S}(t) \setminus \{s\} \cup \{s', s''\}$$

can happen as a result of two possible events:

Innovation: this is the addition of a new feature $\varphi \in \mathbb{N}$ which is **not contained in any species** at the time t. One of the resulting species carries the new feature $s' = s \cup \{\varphi\}$ while the other species has the same features as the ancestral one, $s'' = s$.

Loss: a new species is created by the elimination of a feature. A species is chosen uniformly from the set $\mathbb{S}(t)$. A feature from s is chosen randomly. The loss event is performed only if a species $s \setminus \{\varphi\}$ does not exist in $\mathbb{S}(t)$. In this sense, the elimination of φ from s actually generates a new species. In this case, one of the resulting species is the one carrying the loss $s' = s \setminus \{\varphi\}$ while the other represents the unaltered specie $s'' = s$.

With these events, the resulting shapes of the generated trees are very similar to those found in databases, as we can see in the following figure:

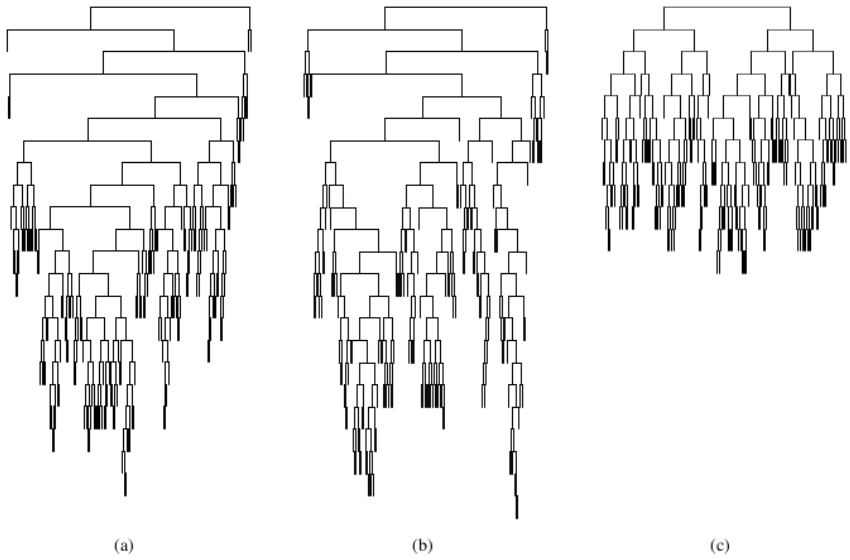


Figure 4.1: Tree shapes of a real tree (a) and two stochastic models: the innovation model (b) and the Yule model(c). Each tree has 161 leaves. Image taken from [20].

Figure 4.1 shows tree phylogenetic trees with 161 leaves. The tree depicted in (a) is from `TreeBASE` Matrix ID M2957, which represents the phyologenetic relationships between Rosids (flowers) based on their mitochondrial matR sequences. The tree in (b) is a tree generated by the innovation model and the tree in (c) is a tree grown by the Yule Model. A striking feature of the innovation model is that it generates species trees with branching patterns very similar to those found in phylogenetic databases. For this reason, this is the stochastic model we chose to implement.

4.2 Methods for Generating Evolutionary Scenarios

As we have seen in the last chapter, computational evolutionary biology methods tend to infer past events from current available data. To validate these inferences different methods have to be tested via in silico generation of Evolutionary Scenarios, where all relations among the constituent parts are known. In general, an evolutionary scenario represents the overall context in which studied traits experience changes that lead to the vast genetic diversity present in extant organisms.

Via in silico generators, we can represent this process at different levels:

1. At **population level**, simulations generate changes within and across populations. The produced individuals are shaped by different models that take into account sex, gene recombination, among other factors.

2. At **species level**, simulations happen by taking single representatives of each species. In this methods, evolution is performed at genome or gene level. For the purpose of this contribution, we will be focusing on the latter..

Numerous tools regarding species-level simulations have been implemented. Just to name a few: `Darwin` [21], which is a programming language per se intended for the life sciences but it includes various models for simulating evolution. `ALF: Artificial Life Framework` [22] which is a simulator intended for genome evolution which covers a wide range of evolutionary events. `EvolSimulator` [23] which is a simpler software package that combines gene family and sequence evolution as well as models to simulate Horizontal Gene Transfer events.

Within the framework of these criteria, we will be focusing on species level simulations. A common practice in this field is to simplify genomes as sets of genes, where genomes containing one gene are also very frequent. The usual pipeline for simulation tools at this level is:

1. Generation of a root species genome. This is usually generated by the simulator or this can be a set of protein or nucleotide sequences given by the user. This genome will denote our initial set of genes, which will undergo evolutionary changes along a designated species tree.

2. Generation of species tree. The species tree is generated by either a stochastic model, a random sampling from the tree of life or user-defined. An internal vertex is denoted as the *root species*.

3. Generation of ancestral sequences. In this step, the generation of ancestral sequences from the root species genome can be done at different levels:

   single nucleotide level : at this level, there are many standardized models [24] where nucleotide substitution happens, and only a few which also accept insertions and deletions.

   gene level : genes, either single or groups of genes can be duplicated or lost.

   genome level : this level comprises the most complex of interactions between genes, the whole genome can be duplicated or chunks can be rearranged.

Since our evolutionary scenario will involve only the true evolutionary history of a single ancestral gene, it is sufficient to consider this four building blocks [25]:

1. A true gene tree $\hat{T}$.

2. A true species tree $\hat{S}$.

3. A *reconciliation map* μ, which assigns every internal vertex of $\hat{T}$ to a vertex or edge of $\hat{S}$. For the purposes of this work, the inner vertices of $\hat{T}$ which correspond to speciations must be mapped to the inner vertices of $\hat{S}$.

4. An assignment of an event type to each vertex of $\hat{T}$.

4.3 Sequence Evolution

In the previous sections, we have spoke of genes as abstract entities which can be symbolically represented as variables in the models aforesaid. Genes are entities that are not yet fully defined in the

biological literature. What we know is that they are hereditary units that are made from sequences of nucleotides, and that these sequences change through time. These changes are the main forces that drive evolution and are vastly responsible for the current biological diversity. Experimentally, the simplest difference between two nucleotide sequences is a nucleotide substitution. For this reason, as seen in [24], we can define the distance between two sequences as an expected number of nucleotide substitutions per site. Substitutions at any particular site are described by a Markov chain where the four nucleotides (A,C,G,T) are the **states** of the chain.

4.3.1 Models of Nucleic Acid Evolution

These models use the Markovian property which is ubiquitously phrased as: "given the present, the future does not depend on the past" which means that the probability with which the Markov chain jumps into other states depends solely on the current state.

Another thing we have to consider while using Markov Chains to model nucleic acid evolution is when $t \to \infty$. The probability that the chain is in state j when $t \to \infty$ is represented by π_j, and the distribution $(\pi_A, \pi_C, \pi_G, \pi_T)$ is known as the **limiting distribution**. When the states of the chain reach this distribution, they will remain in that distribution. For this reason, this distribution is also known as **steady state distribution** or *stationary distribution*.

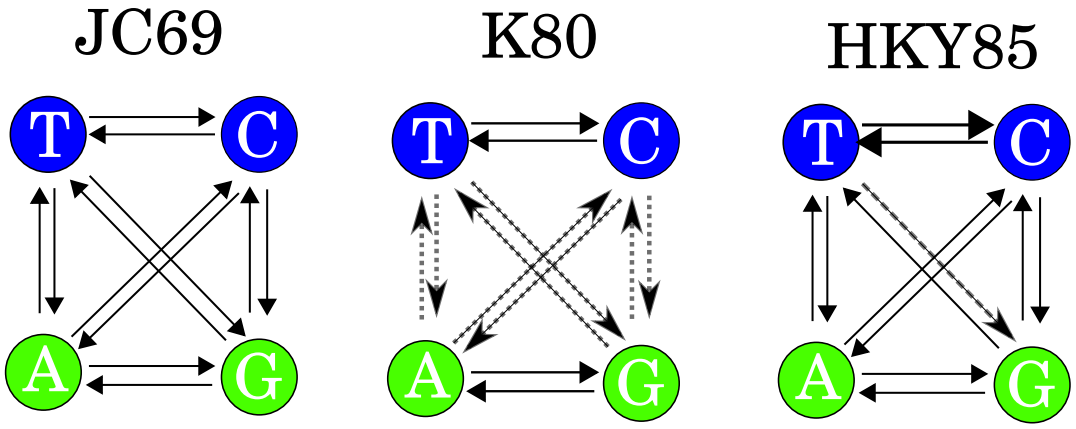


Figure 4.2: Relative Substitution Rates between nucleotides in 3 different Markov Chain Models. Jukes - Cantor Model (JC69), Kimura (K80) and Hasegawa (HKY85). Solid arrows indicate higher substitution rates while dashed arrows indicate lower substitution rates.

JC69: Jukes - Cantor Model. This model assumes that every nucleotide has the same rate of changing into any other nucleotide. In figure 4.2 this is depicted as solid thin arrows between the possible states.

K80: Kimura 1980. In molecular biology, substitutions between (T $\leftrightarrow$ C) or (A $\leftrightarrow$ G) are called **transitions** while substitutions of the form (T,C, $\leftrightarrow$ A,G) are called **transversions**. Kimura used the fact that in experimental data, transitions are present at higher rates than transversions to propose this model. In 4.2, solid arrows represent transitions while dashed arrows represent transversions. This model takes two parameters: a rate for transitions and a rate for transversions.

Hasegawa *et.al.*1984, 1985. In this model, we will find that transitions of the form (T $\leftrightarrow$ C) have a greater substitution rate than any other substitution. Besides this, in this model the steady state distribution of the nucleotides is also different, with a greater prevalence of T and C over A and G.

GTR: The general time-reversible model. A Markov chain is **time reversible** if and only if:

$$\pi_i q_{ij} = \pi_j q_{ji} \text{ for all } i \neq j,$$

Which means that the "flow" between any two states in the opposite direction is the same. All previously presented models fulfil this condition.

Sequence evolution methods also serve as test-beds for distance estimation and thus sequence comparison which serve as means of identifying orthologs. In the next chapter, we will overview a different procedure for identifying orthologs, by using binary relations and graph theory.

Chapter 5

Graphs and Orthology Inference

Using graph theory as basis to study the orthology relations is already an existing field of research which has lead to several interesting inference algorithms. In this chapter we will briefly overview the methods and theory behind current graph based methods for orthology inference and introduce some new results of the formalization behind the main heuristic used by this approaches.

5.1 Orthology in a Mathematical Context

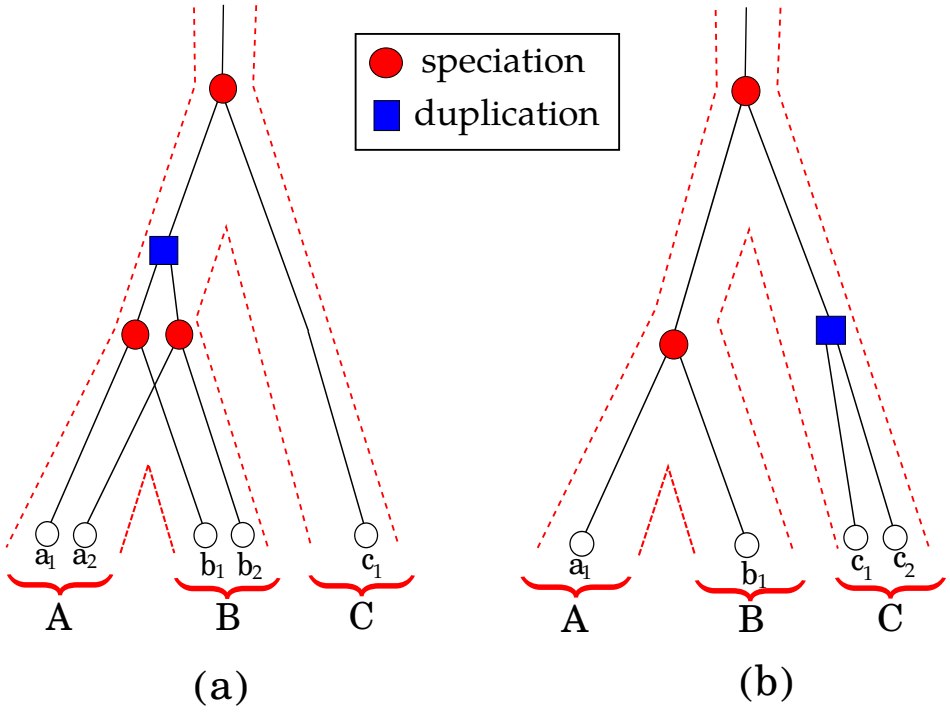


Figure 5.1: An Evolutionary Scenario (left) where G_1 represents the Orthology relation Θ and G_2 all the Paralogous Relations among genes $a_1, a_2, b_1, b_2, c_1, c_2$.

Orthology Θ has been widely acknowledged as a symmetric and non transitive relation i.e. if genes g_1 and g_2 are orthologs and genes g_2 and g_3 are orthologs, then it is not always true that g_1 and g_3 are orthologs. Thus, an orthology relation Θ contains all pairs (x,y) of orthologous genes.

Given a valid orthology relation Θ, we can obtain an event-labelled gene tree (T,t) according to theorem 5 of [26]. The graph that represents Θ is $G_\Theta(V,E)$, where V represents genes and E pairs of genes that are orthologs. If the inner node labelling t of T consisting of duplications and speciations, fulfils that $(x,y) \in E(G_\Theta) \iff lca_T(x,y)$ is a speciation event, then $G_\Theta(V,E)$ is a cograph.

It is worth noticing that empirical estimations of orthology will in general contain erroneous orthologous pairs, i.e. false positives as well as false negatives. Hence, not for all estimated orthology relations there is such a tree (T,t).

In an experimental context, it is also very common to hear that genes are **1-to-1**, **1-to-many**, **many-to-1**, **many-to-many** orthologs. The qualifiers *1* and *many* are used to indicate whether each gene involved in the orthology relation underwent an additional duplication event after their speciation or not. As an illustration of this classification:

- In fig. 5.1 genes a_1 and b_1 are an example of *1-to-1* orthologs, since there are no additional duplication events involved.
- In fig. 5.1 genes a_1 and c_1 are an example of *many to many* orthologs, since their $\texttt{lca}(a_1,c_1)$ is an speciation event, but notice that in the corresponding gene tree there are two preceding duplication events.

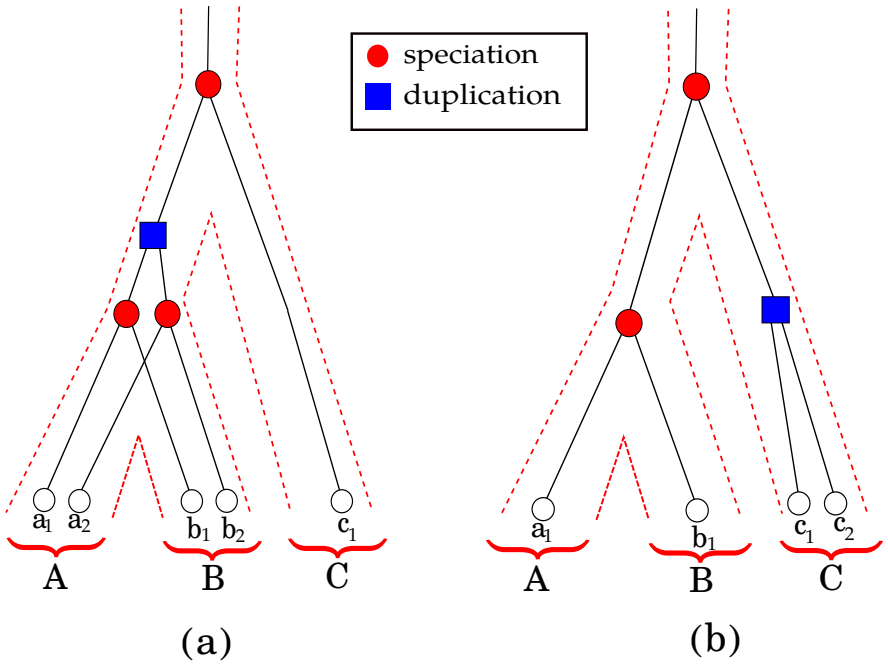


Figure 5.2: Subclassification of **out-paralogs** (a) and **in-paralogs** (b).

Let's now take two different species in the gene tree and their corresponding genes. Analogous to orthologs, paralogs can be further sub-classified as [25]:

- **out-paralogs:** consider fig 5.2 (a). Let $a_1 \in A$ and $b_1, b_2 \in B$. We say that b_1, b_2 are *outparalogs relative to* a_1, if:

  $$lca(b_1, a_1) \prec lca(b_1, b_2) = lca(lca(b_2, a_1)) \text{ or } lca(b_2, a_1) \prec lca(b_1, b_2) = lca(lca(b_1, a_1))$$

  Namely, out-paralogs are genes that diverged *before the speciation* of A and B.

- **in-paralogs:** consider fig5.2 (b). Let Let $a_1 \in A$ and $c_1, c_2 \in C$. We say that c_1, c_2 are *in-paralogs with respect to* a_1 if:

  $$lca(c_1, c_2) \prec lca(c_1, a_1) = lca(c_2, a_1)$$

  In the same fashion, in-paralogs are genes that diverged by a duplication event *after the speciation* of A and C.

5.2 A typical graph-based orthology inference method

Now that we understand the preliminaries of orthology detection, we can now proceed to look at some examples of graph-based orthology detection methods and tools.These methods consist of mainly two phases:

5.2.1 Similarity Graph Construction Phase

Evolutionary closeness in these methods is defined as *sequence similarity score* [27]. The most commonly used tool to perform this kind of analysis is `BLAST: Basic Local Alignment Search Tool` [28]. This tool compares genes or protein sequences and tests the statistical significance of each sequence alignment result. Thus, using the idea that orthologs have the most similar sequences, finding the corresponding ortholog of a gene is equivalent to finding its genome-wide best hit.

Since orthology is a symmetric relation, it is required that its genome-wide best hit is **reciprocal**. This means that orthology can be inferred for a gene pair (a, b) if and only if a is the genome-wide best hit of b ans *vice versa*. This relationship is known as **best bidirectional hit (BBH)**.

Depending on the measurement used to define *sequence similarity*, the reciprocal best hit takes many other different names. For instance when using *Maximum Likelihood* distance estimates, and we take the significant scoring pair-wise alignment, it translates to **reciprocal smallest distance (RSD)** [29].

Hence, by computing any the aforementioned *sequence similarity measurements* between all the genes involved in our study, we can now proceed to properly build a **similarity graph** G_{S}. The vertices of G_{S} represent genes while edges between two genes (a, b) are created only if: a**BBH**b

i.e. their BBH relationship surpasses a statistical threshold or *a***RSD***b* i.e. they fulfil a *minimum length of the alignment* relative to the length of the two input sequences.

Nevertheless, as it has been shown in [17] that building G_{S} on its own is not always sufficient for inferring orthologs. A well-known drawback comes from the existence of *1-to-many* or *many-to-many* orthologs. Since **BBH** takes only the *best hit*, when duplications take place in one or both linages, only part of the true orthology relations is captured. Another limitation are the so-called **differential gene losses**, which are losses that happen along two linages. A small example of this was depicted in fig.3.4.

5.2.2 Clustering Phase

Due to the limitations of the previous phase, most graph-based orthology inference tools cluster orthologous genes into groups which are called **Clusters of Orthologous Groups** (COGs). There are several approaches for doing this:

1. **Markov Clustering:** this method simulates a random walk along the orthology graph where edges are weighed according to similarity scores. The Markov Clustering process then generates probabilities that two genes belong to the same cluster. The graph is partitioned according to this probabilities and each partition forms an orthologous group. This process is used by `OrthoMCL` [30].

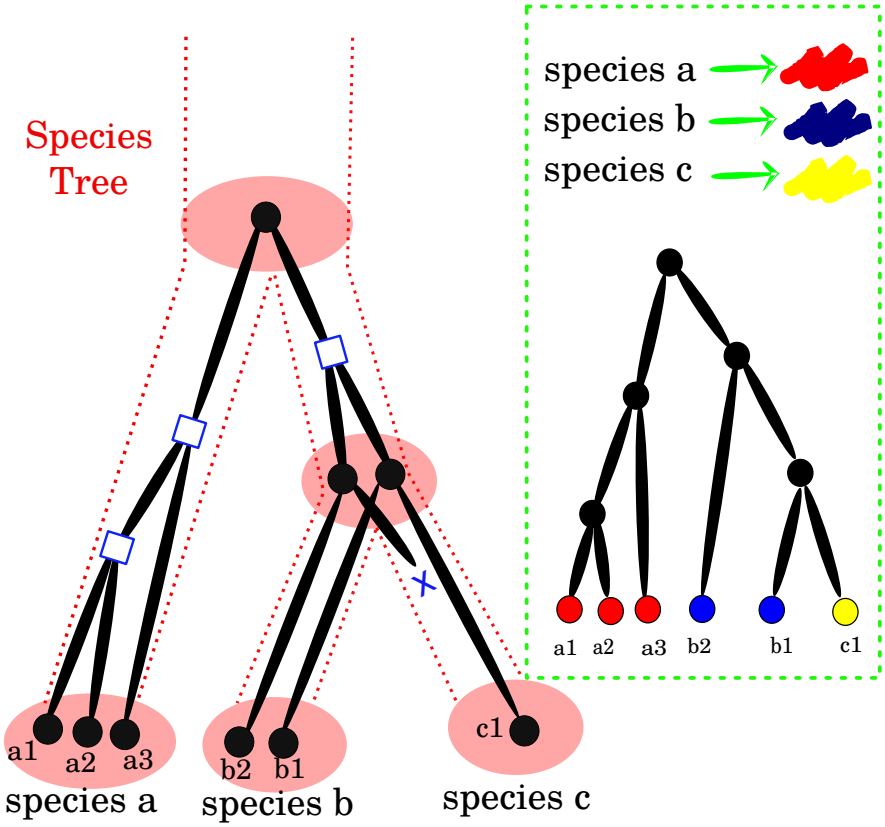
2. **Clique identification:** this method looks for cliques (fully connected subgraphs), which is neatly intuitive, but hard to compute since clique finding belongs to the class of NP-complete problems and no polynomial time algorithm is known.

3. **Spectral Clustering:** this method uses the second eigenvalue Laplacian matrix of G_{S} to determine a bipartition of the vertex set. This process is used by `ProteinOrtho` and its extension `POFF` [31].

4. **Hierarchical Groups:** when grouping strategies are less stringent, this leads to orthologous and paralogous genes residing within the same group. However, it is possible to control what type of paralogs to include in these groups. This analysis is useful when there is a particular *speciation* event of interest[32]. For this reason, *hierarchical groups* are groups in which orthologs and in-paralogs are included with respect to a reference speciation. An example of this, is used by `OMA`[33].

It is interesting to note that graph-based methods and tools rely as a common factor, firstly on defining a measure of *evolutionary closeness.* The source of their main differences comes in most cases from the clustering phase of their respective pipelines. These differences also highlight their main applicability in orthology prediction context [34] which can be : accuracy, higher speed or whether or not it is applicable in large datasets.

5.3 Best Match Graphs



In this section we will abstract further the notions of species tree and its underlying gene tree presented in the previous chapter as follows:

- Take the underlying gene tree without losses. We do not care about the internal labels, only the topology (branching structure) of the tree and the leaf set $\mathbb{G}$ which contains the genes of the underlying gene tree.
- To distinguish each gene, we will assign a colour map to the set of species $\sigma : \mathbb{G} \to \mathbb{S}$ so each species is represented by a colour.
- We now have a Tree (T, σ) with color set $\mathbb{S}$. We will write $\sigma(L') = s$ to represent the colour set of a leaf subset L'.

We can now define a best match relation as:

Definition 5.3.1. The leaf y is a **best match** of the leaf x in T: $x \to y$ if and only if $lca(x,y) \preceq lca(x,y')$ for all leaves y' from the same species as y, i.e. $\sigma(y') = \sigma(y)$.

Note that this definition can also be expressed in terms of divergence times as: y is a best match for x if and only if $t(x,y) \leq t(x,y')$ for all y' with $\sigma(y') = \sigma(y)$. For this reason, the *best match relation* serves as a generalization of *evolutionary closeness*.

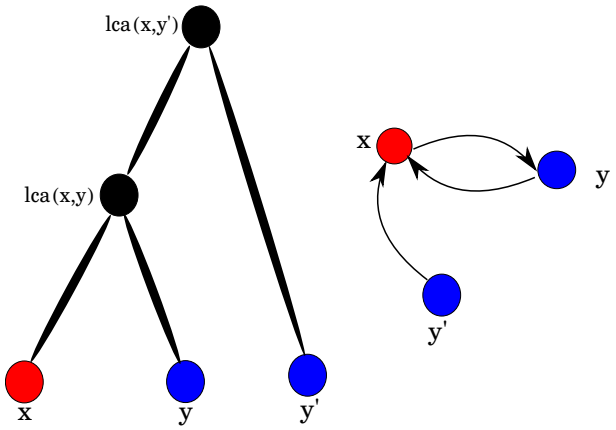


Figure 5.3: A tree T with a colour map σ and its associated Best Match Graph $G(T,\sigma)$.

Definition 5.3.2. Given a tree and a colour map $\sigma : \mathbb{G} \to \mathbb{S}$, the **coloured Best Match Graph (cBMG)** $G(T,\sigma)$ is a digraph which has vertex set $\mathbb{G}$ and arcs $xy \in E(G)$ if $x \neq y$ and $x \to y$. Each vertex $x \in \mathbb{G}$ has the colour $\sigma(x)$.

5.4 Reciprocal Best Matches

As we have seen at the beginning of this chapter, the main heuristic of graph-based orthology detection tools relies upon reciprocal best matches, which correspond to all bidirectional edges of a cBMG. We will therefore refer to this symmetric part of a cBMG as **coloured Reciprocal Best Match (cRBMG)**.

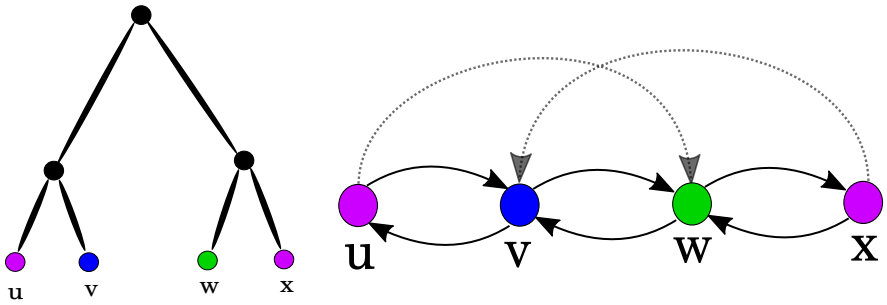


Figure 5.4: A tree with its associated RBMG, dashed arrows represent the non-symmetrical edges contained in the corresponding BMG.

Recall that orthology relations are associated to cographs, since orthology has a cograph structure. Nevertheless, cRBMG are not necessarily cographs. Take as example 5.4. In this example we can see that there is a P_4 path $u-v-x-w$ in the symmetric part.

This issue is furtherly addressed in [5]. In the next chapter, we will use this characterization to illustrate the utility of our simulation tool.

Chapter 6

Results

In this section, we present an implementation of a simple stepwise procedure for the generation of Evolutionary Scenarios which includes the necessary theoretical orthology graphs and its application study case regarding the graphs presented in section 5.

6.1 Evolutionary Scenario Generation

As seen in chapter 4, to represent an evolutionary history, we use the main components of an Evolutionary Scenario: a true species tree $\hat{S}$, a true gene tree $\hat{T}$ and a reconciliation map μ. Since, while dealing with biological data we cannot find the lost genes explicitly, i.e. extant genes are the only things we can "observe" we will incorporate to this components an *observable tree* T, which is a gene tree extracted from $\hat{T}$ in which the losses and vertices of degree 2 are pruned. We will now review every component in detail.

6.1.1 True Species Tree $\hat{S}$

For this part, we undertook the *innovation model* presented in section 4. Although this stochastic model is interesting, throughout the literature, there are no reported tools that use this model to generate species trees. Thus, given a user-defined number of species N, we generate a species tree $\hat{S}$ under the innovation model. Recall that in this model, every species s is represented as a set of features. The branching process is expanded by the generation of novel features and an exhaustive combination with available ones.

To represent this process we use the following auxiliary sets:

- $\mathbb{A}(t) :=$ available species for performing feature losses i.e. this set contains all the species whose loss of a feature φ will trigger the generation of a new species.
- $\mathbb{F}(t) :=$ set of available features at time t.
- $\mathbb{S}(t) :=$ set of extant species at time t.

Algorithm 1 Species Tree via the Innovation Model

Require: number of desired leaves in species tree N

Set $t = 1, \mathbb{F}(0) = \{0\}, \mathbb{S}(0) = \{\{0\}\}$
while $|\mathbb{S}(t)| < N$ **do**
  Compute $\mathbb{A}$
  if $|\mathbb{A}| = \emptyset$ **then**
    {Innovation Event}
    Draw $s \in \mathbb{S}(t)$ uniformly
    Set $\varphi = 1 + \texttt{max}\ \ (F(t) \cup \{0\})$
    Set $\mathbb{S}(t+1) = \mathbb{S}(t) + \{s \cup \{\varphi\}\}$
    Set $F(t+1) = F(t) \cup \{\varphi\}$
    $t = t + 1$
  else

    Draw $s \in \mathbb{S}(t)$ uniformly
    Draw $\varphi \in s$ uniformly
    if $s \setminus \{\varphi\} \notin \mathbb{S}(t)$ **then**
      {Loss Event}
      $\mathbb{S}(t+1) = \mathbb{S}(t) \cup \{s \setminus \{\varphi\}\}$
      $\mathbb{F}(t+1) = \mathbb{F}(t)$
      $t = t + 1$

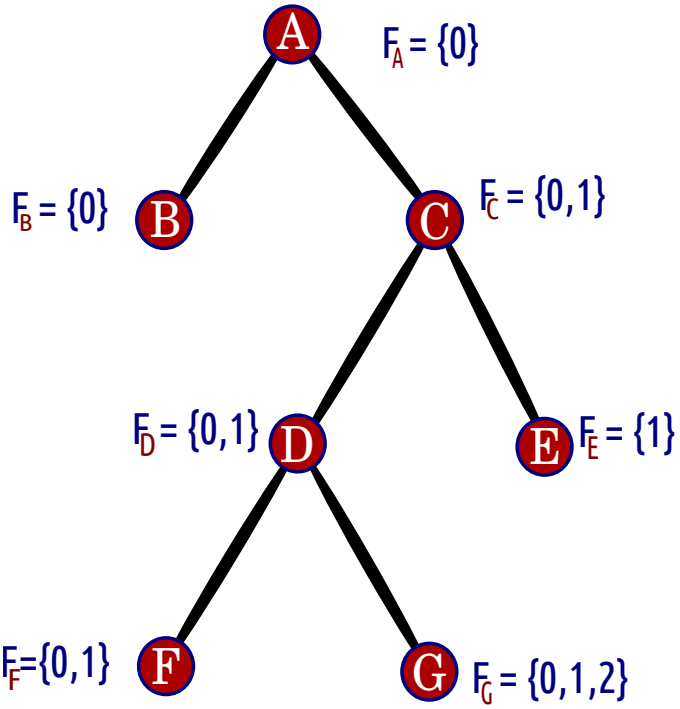


Figure 6.1: An example of a species tree growing via the *innovation model*. F_N represents the set of features of each species. Species A diverges into $B = \{0\}$ and $C = \{0,1\}$ by the generation new feature $\varphi = 1$. On the other hand, species C diverges into $D = \{0,1\}$ and $E = \{1\}$ by the dissapearance of a feature $\varphi = 0$.

6.1.2 Branch Length Assignation

Since we regard $\hat{S}$ as an ultrametric tree, i.e. its branch lengths are interpreted as real time, we assign $\hat{S}$ an **Evolutionary Time Normalization**. This is, every edge in the Species Tree must have a branch length associated with it that represents a time scale. This is done as follows:

For every edge $e = (u, v)$ where u is the immediate ancestor of v in the species tree there is a branch length function ω such that:

$$\omega(e) = \begin{cases} \frac{1}{|P|} & e \in P \\ \frac{1 - \sum_{n=\rho}^{u} l(n)}{\min_{h \in \mathbb{S}}\{|p \setminus (v,u)|\}+1} & \text{otherwise} \end{cases} \tag{6.1}$$

where:

- ρ is the root of the species tree
- p is the path from v to a leaf h
- $\mathbb{S}$ is the leaf set of the species tree
- h_0 is the closest leaf to v

With this function it is easy to see that there are two types of edges:

- An edge (u, v) that belongs to a longest path P. In that case, its branch length is inversely proportional to the length of P.
- An edge (u, v) which is not contained in a longest path P. In this case we need to take into consideration the sum of the branch lengths in the path from the root of the species tree to the vertex u and the number of edges in the path to the closest leaf.

An example of this method of branch length assignation is depicted in fig. 6.2.

6.1.3 Gene Tree Generation

Previous to the creation of the Gene Tree, as in [35], we assign the number of events that must be generated per edge:

6.1.4 Poisson Vector Creation

Required Input The user must give a probability of Duplication $P(D)$, HGT $P(H)$ and gene loss $P(L)$. This will represent the probability of that event happening along an edge in the species tree.

Number of Events For each event $e = D, H, L$, we draw random samples from a poisson distribution with $\lambda = P(e) \times \omega(u, v)$

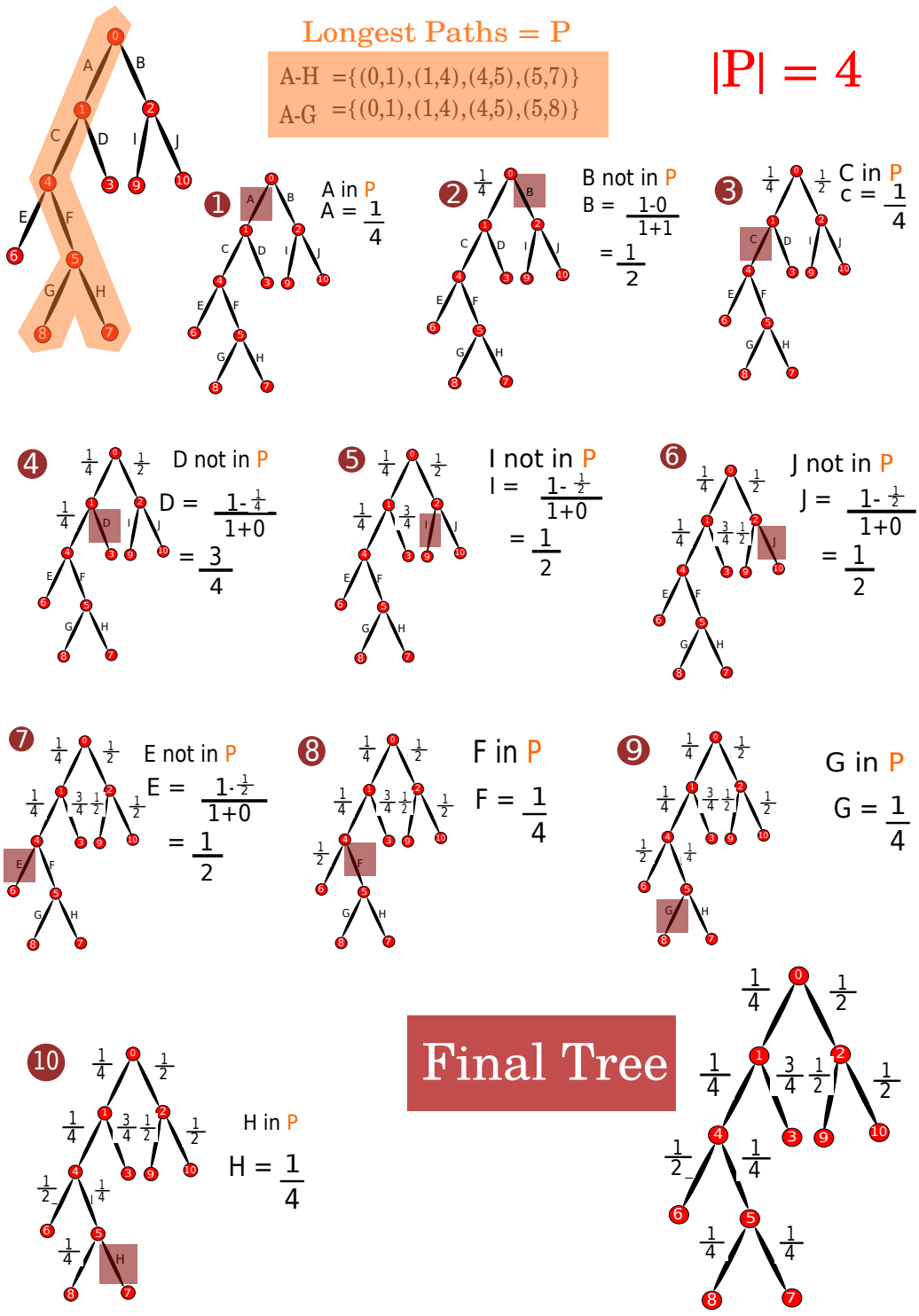


Figure 6.2: An example of branch length assignation using ω

By top down traversal of $\hat{S}$, we generate $\hat{T}$, starting from a single lineage, which is represented by placing a single ancestral gene as the gene of the ancestral root species of $\hat{S}$.
Thus, we generate an event-labelled tree where:

- All inner vertices are labelled by the event type: duplication, speciation or HGT.
- Duplications can only happen inside edges of $\hat{S}$. Both resulting copies remain in the same edge than their ancestor.
- HGTs happen along the edges of $\hat{S}$ as well. From the two resulting vertices of this event, one descendant travels along the linage it was generated and another one is sent to a different species that exists at the same time as the donor species. This time is given by the branch length assignation described in the previous section.
- Loss events are regarded as leaves of $\hat{T}$ which can not be extended in further steps. We use different labels to distinguish extant genes from losses.

Using this procedure, we get also the reconciliation map μ.

$\hat{T}$ is then created by using the aforementioned rules and the Poisson Vector. Finally $\hat{T}$ is printed as a **newick string** which is a standard for the computational representation of trees and is used by several tools for further processing and visualization of phylogenetic trees. We assign the following newick notation for vertices that are not extant genes:

(a,b)D : For a duplication happening between genes *a*,*b*.

(a,b)S : For a speciation happening between genes *a*,*b*.

(a,b)H : For a horizontal gene transfer between genes *a*,*b*.

* : For a lost gene.

Leaf labels have the form `x.y` where `x` is an integer designating the gene family and `y` enumerates the genes within a family. This notation is intended for further updates of this simulation, that will incorporate the simulation of more than one gene family. In this work, we consider only the evolutionary history of one gene family.

(((1.8,(1.10,1.9)**D**)**S** , (((1.11,1.12)**S** , ((1.15,1.14)**H**, 1.13)**S**)**H** ,1.5)**H**)**S** ,1.2)**S**;

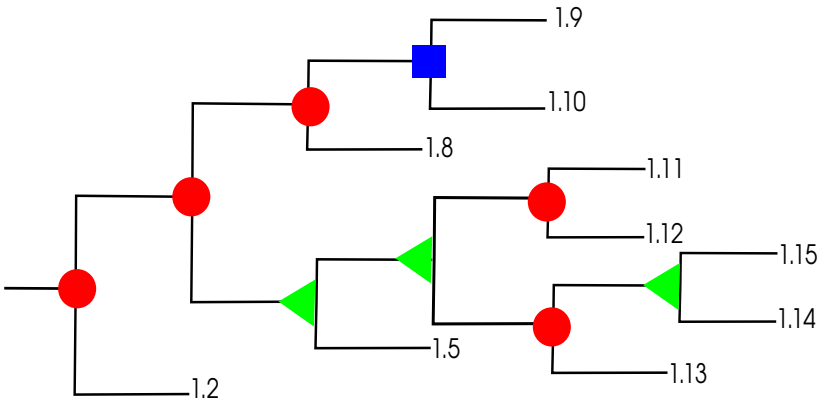


Figure 6.3: A true gene tree $\hat{T}$ with is corresponding newick string representation.

6.1.5 Construction of Observable Tree T

For the construction of the Observable Tree, we will take the following notions as basis for this construction:

(i). We want a phylogenetic tree on $\mathbb{G}$, i.e. the set of extant genes of $\hat{T}$, thus, we need to remove every edge in $\hat{T}$ that leads to a gene loss.

(ii). If we remove these edges, we leave the remaining incident vertex with a degree of 2, an thus, these vertices will lead to a tree that is not a phylogenetic tree.

Whenever a vertex v fulfils either (i) i.e. it is a leaf labelled as gene loss or (ii) it is the immediate ancestor of a gene loss, we will label it as 1, otherwise we will label it as 0. For every vertex $v \in V(\hat{T})$, we will assign a "provisional root set": $\rho(v)$ based on this labelling.

We start by a **Post-order** traversal of $\hat{T}$, where we have the following cases:

- $v =$ leaf. If its label is $= 1$, then assign $\rho(v) = \emptyset$. Else, $\rho(v) = \{v\}$.
- $v =$ inner vertex. Let x, y be *descendants* of v. If v's label is $= 1$, then assign $\rho(v) = \{\rho(x), \rho(y)\}$. Otherwise, add the edges $\{(v, \rho(x)), (v, \rho(y))\}$, then assign $\rho(v) = v$.

Thus, all added edges in these steps will belong to T.

6.1.6 Sequence Generation

As seen in section 4, the simplest difference between two *homologous* DNA sequences is a **base substitution**. By using `Pyvolve`, we can implement a **General Time Reversible (GTR)** model of nucleotide evolution along our generated trees. The general pipeline for using Pyvolve is:

1. Store a gene tree as a newick string with the associated branch lengths.
2. Definition of an evolutionary model for each evolutionary event. Essentially, all events evolve with the same model (the `nucleotide` model), the difference lies within the substitution rate and the chosen equilibrium frequencies.
3. Evolution of sequences according to phylogeny and models. Sequence length is a user-defined parameter.
4. Sequence post processing: This process results in the sequence generation for the extant genes.

Since the number of substitutions depends also on the branch length of the gene tree we can define a model that follows:

$$\omega'(D_{(a,b)}) > \omega'(S_{(a,b)}) > \omega'(H_{(a,b)})$$

where:

$\omega'(D_{(a,b)})$ **:** Returns the weight associated to the edges (D,a) and (D,b) i.e. the edges between the descendants of a duplication-labelled vertex in $\hat{T}$. For the purpose of this contribution, we set this parameter as $\omega'(D_{(a,b)}) = 2$.

$\omega'(S_{(a,b)})$: Returns the weight associated to the edges (S,a) and (S,b) i.e. the edges between the descendants of a speciation-labelled vertex in $\hat{T}$. For the purpose of this contribution, we set this parameter as $\omega'(S_{(a,b)}) = 1$.

$\omega'(H_{(a,b)})$:Returns the weight associated to the edges (H,a) and (H,b) i.e. the edges between the descendants of a HGT-labelled vertex in $\hat{T}$. For the purpose of this contribution, we set this parameter as $\omega'(H_{(a,b)}) = 0.5$.

Following the theory described in sections 3 and 4, we want duplications to generate a bigger number of substitutions, followed by speciations and lastly HGTs. Considering only branch length though, all events would evolve using the same default evolutionary `Pyvolve` model (the `nucleotide` model). To better complement our branch-length model, we define a `Pyvolve` evolutionary model for each event type as:

Speciations For this we define a JukesCantor model, with limiting distribution $\pi_A = \pi_C = \pi_G = \pi_T = 0.25$ and transition probabilities $P(A \to C) = P(A \to G) = P(A \to T) = P(C \to G) = P(C \to T) = P(G \to T) = 0.5$.

Duplications In this model we define a model with limiting distribution $\pi_A = \pi_C = \pi_G = \pi_T = 1.5$ and transition probabilities $P(A \to C) = 0.5, P(A \to G) = 0.25, P(A \to T) = 1.23, P(C \to G) = 0.55, P(C \to T) = 1.22, P(G \to T) = 0.47$.

HGTs In this model we define a model with limiting distribution $\pi_A = \pi_C = \pi_G = \pi_T = 1.5$ and transition probabilities $P(A \to C) = 0.5, P(A \to G) = 0.25, P(A \to T) = 1.23, P(C \to G) = 0.55, P(C \to T) = 1.22, P(G \to T) = 0.47$.

All generated sequences are outputted as .phy files (the PHYLIP standard for multiple sequence alignment [36]) and can be also generated as a python dictionary where the keys are the extant gene identifier and the value its corresponding sequence for further post-processing.

6.2 True Orthology Graph Generation

The graph that represents the Orthology relation G_Θ is build recursively from the True Gene Tree $\hat{T}$:

1. For each $v \in \hat{T}$ we assign a graph $G(v)$ in the following way:
   - if v is an extant gene, then $G = K_1$ with $V(K_1) = \{v\}$.
   - if v is a loss, then $G = \emptyset$ (an empty graph).
   - if v is an inner vertex then $G(v) =$ join $+$ of all the children of v if v is labelled as an speciation, otherwise $G(v) =$ disjoint union $\oplus$ of all the children of v.

2. The graph assigned to the root of $\hat{T}$ will be the graph that represents the orthology relation of the scenario.

To ensure that the our join operation between two graphs G_1 and G_2 follows the definition of *lca-orthologous* presented in chapter 3, the edge set of $G_1 + G_2$ will be $E(G_1) \cup E(G_2) \cup \{uv | u \in V(G_1), v \in V(G_2)$ and $\sigma(u) \neq \sigma(v)\}$. Also, it is important to note that since $G + \emptyset = G \oplus \emptyset = G$, there is no contribution of the losses to G_Θ.

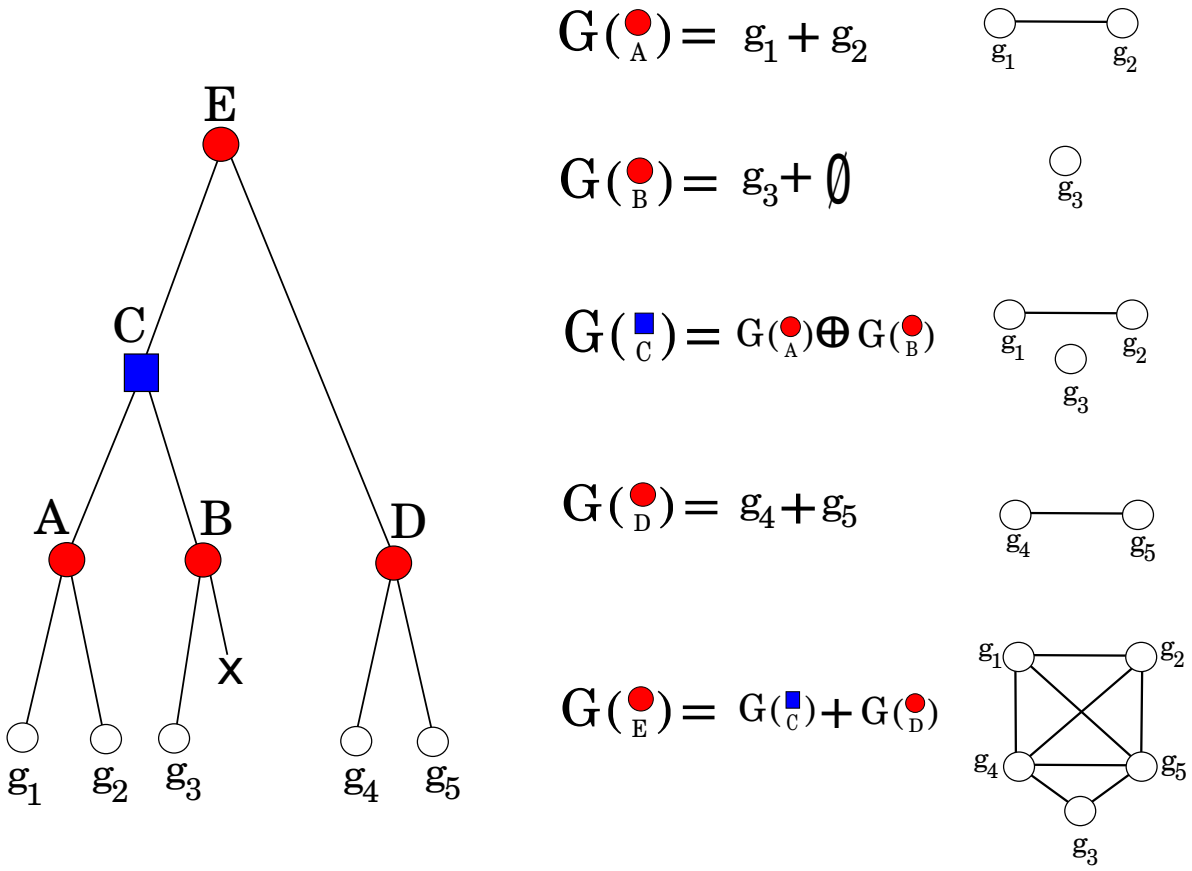


Figure 6.4: True Orthology Graph Construction example. Only graphs assigned to inner vertices are shown. In the r.h.s. we show the recursive construction of the true orthology graph. The graph assigned to vertex E represents the true orthology graph of this scenario.

6.3 Best Match Generation

Using the Observable Gene Tree T with leaf set $\mathbb{G}$ we propose a simple algorithm for the generation of a n-coloured BMG by following definition **5.3.1**:

First as an auxiliary variable, for every vertex in T we define a collection of n sets called $\mathbb{L}$ defined for every vertex $v \in V(T)$ which corresponds to the leaves of each colour $1, 2, \ldots, n$ respectively that "hang" from the subtree rooted at v : $T_{(v)}$. We first initialize the sets corresponding to $\mathbb{L}$ as empty sets. By traversing T in **Post-order**, we face the following options:

- If $v =$ **leaf**, then add v to the corresponding colour set $\sigma(v)$ in $\mathbb{L}$. Notice that in this step, the set of leaves that are from colour $\neq \sigma(v)$ will remain as empty sets.

- If $v =$ **inner vertex**, then, assign to $\mathbb{L}(v)$ the union of their descendant's colour sets.

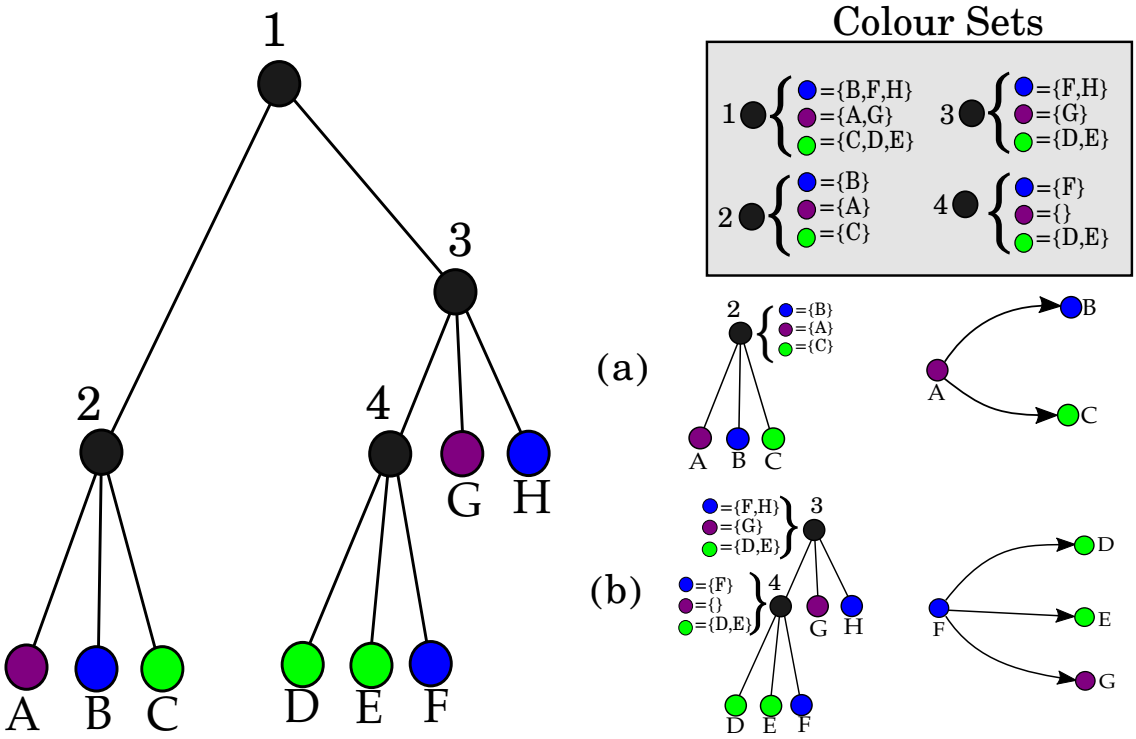


Figure 6.5: Best Match Graph construction example. The precomputed colour sets of every inner vertex in T are shown in the upper r.h.s. In (a) to assign the best matches of vertex A we look at the colour sets of 2 and add the edges (A,B) and (A,C). In (b) to assign the best matches of vertex F by looking at the colour set of 4 we assign the edges (F,D) and (F,E). Since we miss one colour, it is necessary to "ascend" to vertex 3 and add the missing colour best matches. In this case (F,G)

With these precomputed variables, we can now proceed to the generation of the n-cBMG by using **Algorithm 2**, where:

`a_colour` list of available colours removing the colour of `leaf`. This will retrieve the vertex's best matches.

`parent_colour_set` Non empty colour sets compatible with available colours (`a_colours`) i.e. colour sets which are not σ(leaf).

As shown in [5], this algorithm runs in $O(|\mathbb{G}|^2)$ time using a single post-order traversal of T. The Reciprocal Best Match is then computed by taking the symmetrical part of each BMG. This step is also bounded by $O(|\mathbb{G}|^2)$.

Algorithm 2 Obtain n-cBMG

Require: Species Tree $\hat{S}$,Observable Gene Tree T
1: For every vertex v in T, compute $\mathbb{L}$
2: BMG $\leftarrow$ DiGraph()
3: `BMG.vertices` $\leftarrow$ Coloured leaves of T
4: **for each** leaf l **in** BMG.vertices **do**
5: current $\leftarrow$ leaf
6: a_colour $\leftarrow \{s | s \in \sigma(T_s) \setminus \sigma(l)\}$
7: **while** a_colour **do**
8: current $\leftarrow$ `current.parent()`
9: parent_colour_set $\leftarrow$ Non empty colour sets of parent
10: **if** parent_colour_set **then**
11: **for each** colour $c \in$ parent_colour_set **do**
12: **for each** vertex $n \in$ colour **do**
13: `BMG.add_edge` (l,n)
14: `a_colour.remove(`c`)`

6.4 Execution Time

The time performance of this simulation environment , without the sequence generation part, was evaluated using different event parameters:

- Since duplication-loss scenarios are of particular interest in this field, a test regarding $P(D) = P(L) = 0.75$ and $P(H) = 0.0$
- Scenarios containing all possible events where then tested using different event probabilities:
  1. $P(D) = P(L) = P(H) = 0.25$
  2. $P(D) = P(L) = P(H) = 0.50$
  3. $P(D) = P(L) = P(H) = 0.75$

Each of these parameter sets was evaluated for 100 scenarios where the true species tree $\hat{S}$ had 25 leaves. Time measurements comprise from the generation of the evolutionary scenario, their true orthology graph to the creation of the best match graph and the reciprocal best match graph. These results are summarized in fig. 6.6.

As expected, our stepwise procedure time performance behaves non-linearly with the increase of leaves in the species tree. Error bars where plotted every 5th measurement. The associated error, increases with the number of leaves in the species tree. One downside of our methodology comes from the fact that the generation of large numbers of HGTs is time-wise expensive, as shown in fig. 6.6. The observed increase in the error bar for HGTs ranging from 40 to 50 on the inferior r.h.s. panel where $P(D) = P(L) = P(H) = 0.75$ might be a result of the independent generation of HGT branches in the true gene tree $\hat{T}$. Since our method generates events sequentially, HGT events that happen in already generated lineages have to evolve independently.

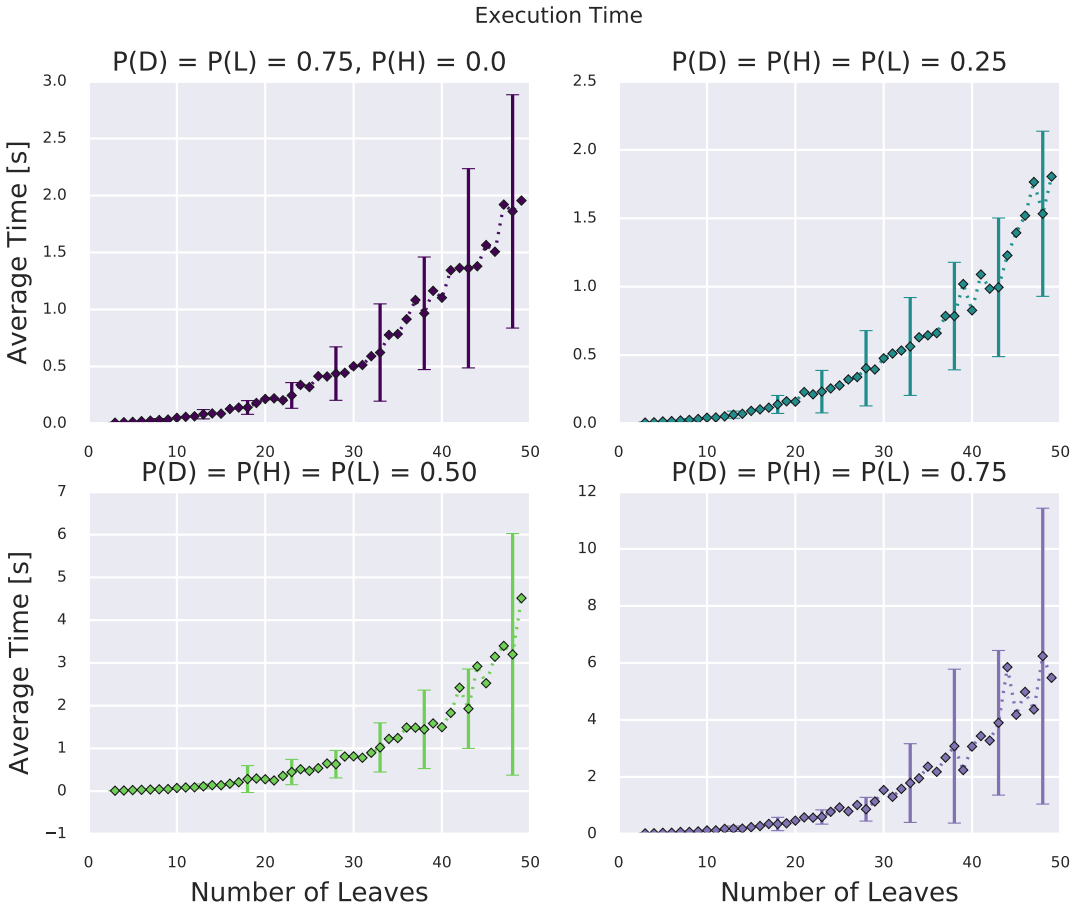


Figure 6.6: Execution time test. The simulation behaves non-linearly as the amount of species tree leaves and event probabilities increase. Error bars are plotted every fifth measurement.

6.5 Application: Effects of HGTs in Orthology Estimations

We illustrate the utility of the Evolutionary Scenarios generated by this simulation tool with the following application.

In the last section we have seen that Reciprocal Best Matches are not necessarily cographs and provided the graph in 5.4 as counterexample. We will denote this graph with G_4, i.e. any digraph isomorphic to the 3-coloured digraph with:

- $V(G_4) = \{u, v, w, x\}$.
- $E(G_4) = \{(u,v), (v,u), (v,w), (w,v), (w,x), (x,w), (u,w), (x,v)\}$.
- $\sigma(u) = \sigma(x)$

In [5], it is shown that in the absence of HGTs, the True Orthology Graph $\Theta(T, \sigma)$ is a subgraph of the Reciprocal Best Match $\vec{G}(T, \sigma)$. Moreover, in [37] it is also shown that the undirected edge (v, w) of an induced coloured sub-digraph isomorphic to G_4 cannot be part of an Orthology Graph. Using our simulation tool we address the new issue of finding G_4 as an induced coloured sub-digraph of a Best Match Graph G and removing all the (v, w) edges, without loss of generality.

As presented in [5], we propose a simple way of finding this graph based on its *degree sequence*, which is the list $\alpha = ((\alpha_x^+, \alpha_x^-) | x \in V(\vec{G}))$ of pairs (α_x^+, α_x^-) where α_i^+ is the out-degree and α_i^- is the in-degree of the vertex $x \in V(\vec{G})$, since G_4 is is uniquely defined by its degree sequence $((2,1),(2,1),(2,3),(2,3))$. Also, to examine the colouring, it suffices to check that the two vertices with out-degree 1 have the same colour $\sigma(u) = \sigma(x)$.

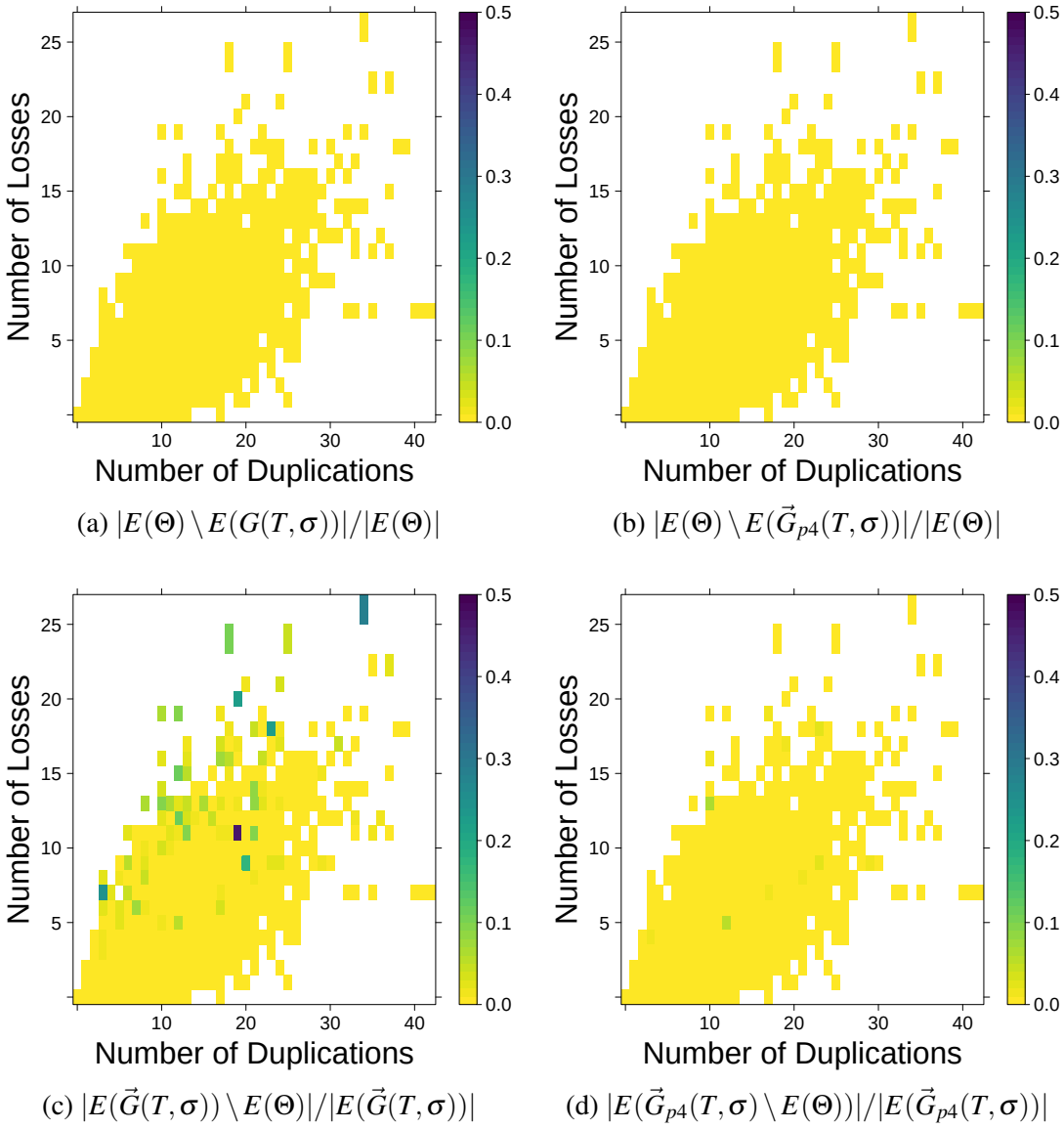
Thus, we will remove from the Reciprocal Best Match Graphs the undirected edge (v, w), which is the one connecting the two vertices with outdegree 3. This edition was performed firstly by marking this edge in the Reciprocal Best Match Graph $\vec{G}(T, \sigma)$ and then removing all the marked edges.

We will test the effects of this edition by generating the following graphs and comparing their corresponding edges in every evolutionary scenario:

- The True Orthology Graph $\Theta(T, \sigma)$.
- The Best Match Graph $G(T, \sigma)$.
- The Reciprocal Best Match Graph $\vec{G}(T, \sigma)$.
- The Corrected P_4 Graph $\vec{G}_{p4}(T, \sigma)$. This graph is the result of removing the (v, w) edges in the induced G_4 from $\vec{G}(T, \sigma)$.

By using a wide range of parameters : high Duplications, HGTs and Losses probabilities ranging from 0.5 to 1, and Species Trees with leaves ranging from 3 to 100 we generated the necessary Evolutionary Scenarios for these comparisons and computationally explore the effect of HGTs.

6.5.1 Duplication-Loss Scenarios

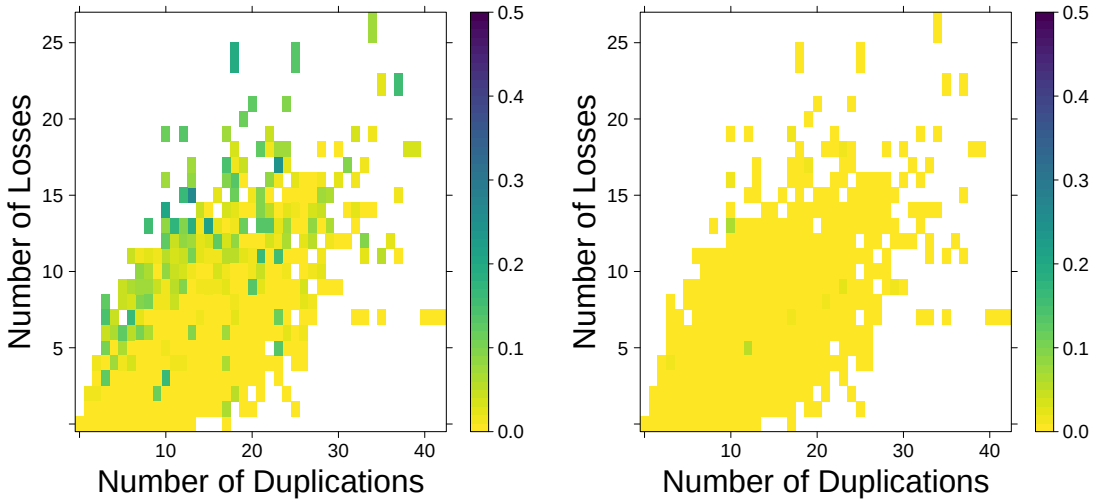


(a) $|E(\Theta) \setminus E(G(T,\sigma))|/|E(\Theta)|$ (b) $|E(\Theta) \setminus E(\vec{G}_{p4}(T,\sigma))|/|E(\Theta)|$

(c) $|E(\vec{G}(T,\sigma)) \setminus E(\Theta)|/|E(\vec{G}(T,\sigma))|$ (d) $|E(\vec{G}_{p4}(T,\sigma) \setminus E(\Theta))|/|E(\vec{G}_{p4}(T,\sigma))|$

Figure 6.7: True Orthology Graph Θ edge comparison against (a) Reciprocal Best Match $\vec{G}(T,\sigma)$ and (b) Corrected P_4 Graph $\vec{G}_{p4}(T,\sigma)$. (c) Reciprocal Best Match $\vec{G}(T,\sigma)$ comparison against the True Orthology Graph Θ. (d) Corrected P_4 Graph $\vec{G}_{p4}(T,\sigma)$ edge comparison against the True Orthology Graph Θ.

For scenarios were there are no HGTs, firstly in fig. 6.7(a) and 6.7(b) we confirm that $E(\Theta) \subseteq E(\vec{G}(T,\sigma))$ and therefore also $E(\Theta) \subseteq E(\vec{G}_{p4}(T,\sigma))$ in the absence of HGTs. Also following our expectations in fig. 6.7(c), the fraction of edges of the Reciprocal Best Match $\vec{G}(T,\sigma)$ which is not contained in the True Orthology Graph Θ, is small for moderate quantities of duplications and losses. Moreover in fig. 6.7(d), we can further see that in most cases the Corrected P_4 Graph $\vec{G}_{p4}(T,\sigma)$ is a better approximation to Θ.



(e) $E(G(T,\sigma)) \setminus E(\vec{G}(T,\sigma))|/|E(G(T,\sigma))|$ (f) $|E(\vec{G}(T,\sigma)) \setminus E(\vec{G}_{p4}(T,\sigma))|/|E(\vec{G}(T,\sigma))|$

Figure 6.8: (e) Best Match Graph $G(T,\sigma)$ edge comparison against the Reciprocal Best Match Graph $\vec{G}(T,\sigma)$. (f) Reciprocal Best Match Graph $\vec{G}(T,\sigma)$ comparison against Corrected P_4 Graph $\vec{G}_{p4}(T,\sigma)$.

When the Reciprocal Best Match $\vec{G}(T,\sigma)$ is a correct approximation of the True Orthology Graph Θ, the coloured Best Match Graph $G(T,\sigma)$ is dominated by reciprocal edges, as indicated by fig. 6.8(e). This result also shows that the non-reciprocal matches increase much faster than the edges in fig. 6.7(c). In fig. 6.8(f) we can see that there is a small number of edges that where edited from the Reciprocal Best Match $\vec{G}(T,\sigma)$. We summarize these results in the following figure:

Figure 6.9: Boxplot Summary of Duplication-Loss Scenarios. (a) $|E(\Theta) \setminus E(G(T,\sigma))|/|E(\Theta)|$. (b) $|E(\Theta) \setminus E(\vec{G}_{p4}(T,\sigma))|/|E(\Theta)|$. (c) $|E(\vec{G}(T,\sigma)) \setminus E(\Theta)|/|E(\vec{G}(T,\sigma))|$. (d) $|E(\vec{G}_{p4}(T,\sigma) \setminus E(\Theta))|/|E(\vec{G}_{p4}(T,\sigma))|$. (e) $E(G(T,\sigma)) \setminus E(\vec{G}(T,\sigma))|/|E(G(T,\sigma))|$. (f) $|E(\vec{G}(T,\sigma)) \setminus E(\vec{G}_{p4}(T,\sigma))|/|E(\vec{G}(T,\sigma))|$.

6.5.2 Horizontal Gene Transfer Scenarios

From all the resulting simulations with $HGTs > 0$, we grouped the scenarios by proportion of HGTs w.r.t. Duplications and Losses. This proportion is expressed as a percentage. For these scenarios, we used high duplication and loss probabilities > 0.6 and low HGT probabilities < 0.25.

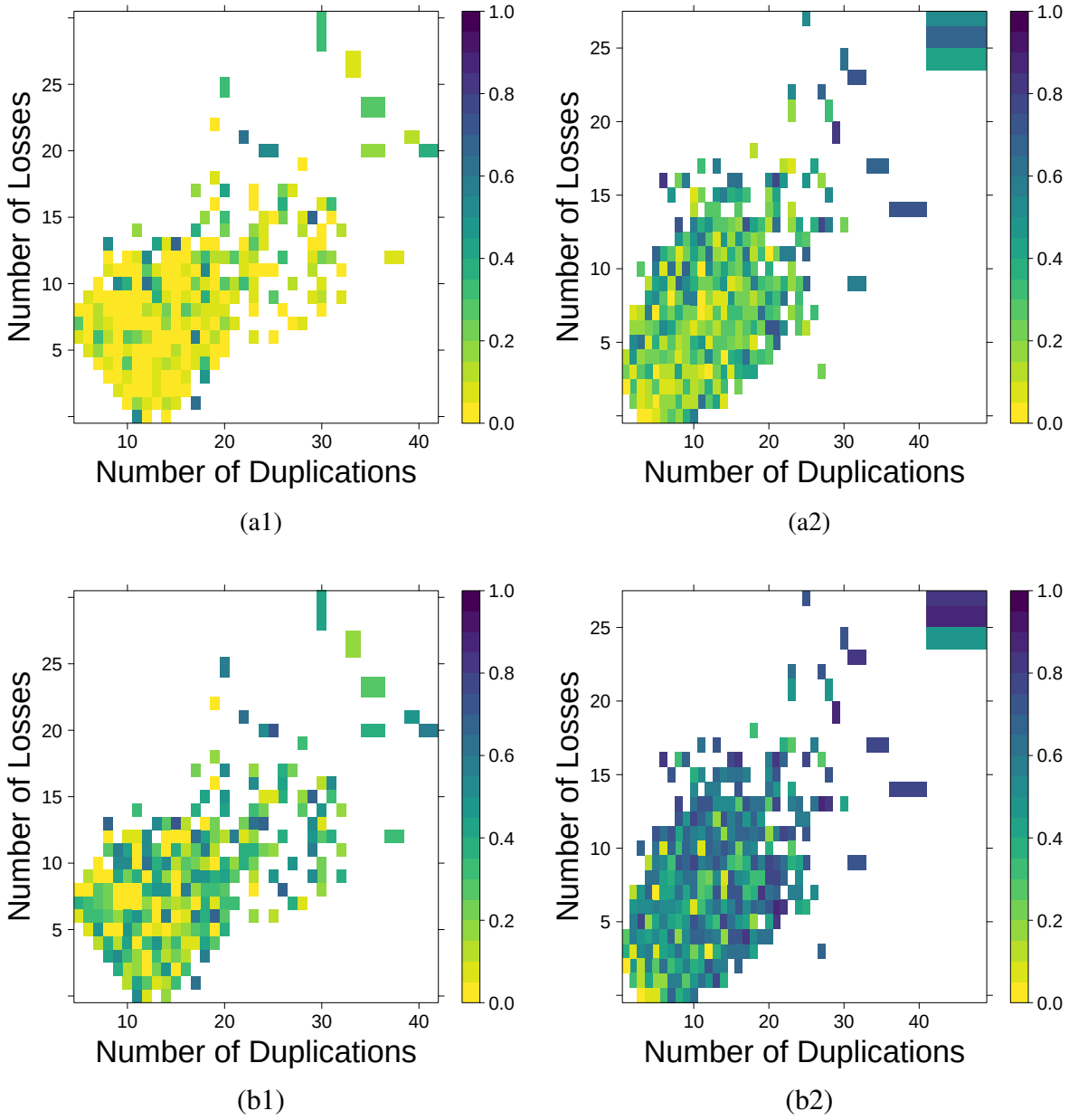


Figure 6.10: True Orthology Graph Θ edge comparison against Reciprocal Best Match $\vec{G}(T,\sigma)$ for scenarios with HGTs (a1) between 4% and 8% and (a2) between 16% and 32%. True Orthology Graph Θ edge comparison against Corrected P_4 Graph $\vec{G}_{p4}(T,\sigma)$ for scenarios with HGTs (b1) between 4% and 8% and (b2) between 16% and 32%

Contrary to our expectations, with HGTs the Reciprocal Best Match $\vec{G}(T,\sigma)$ becomes a slightly better approximation to the True Orthology Graph Θ than the Corrected P_4 Graph $\vec{G}_{p4}(T,\sigma)$ as shown in fig.6.10.

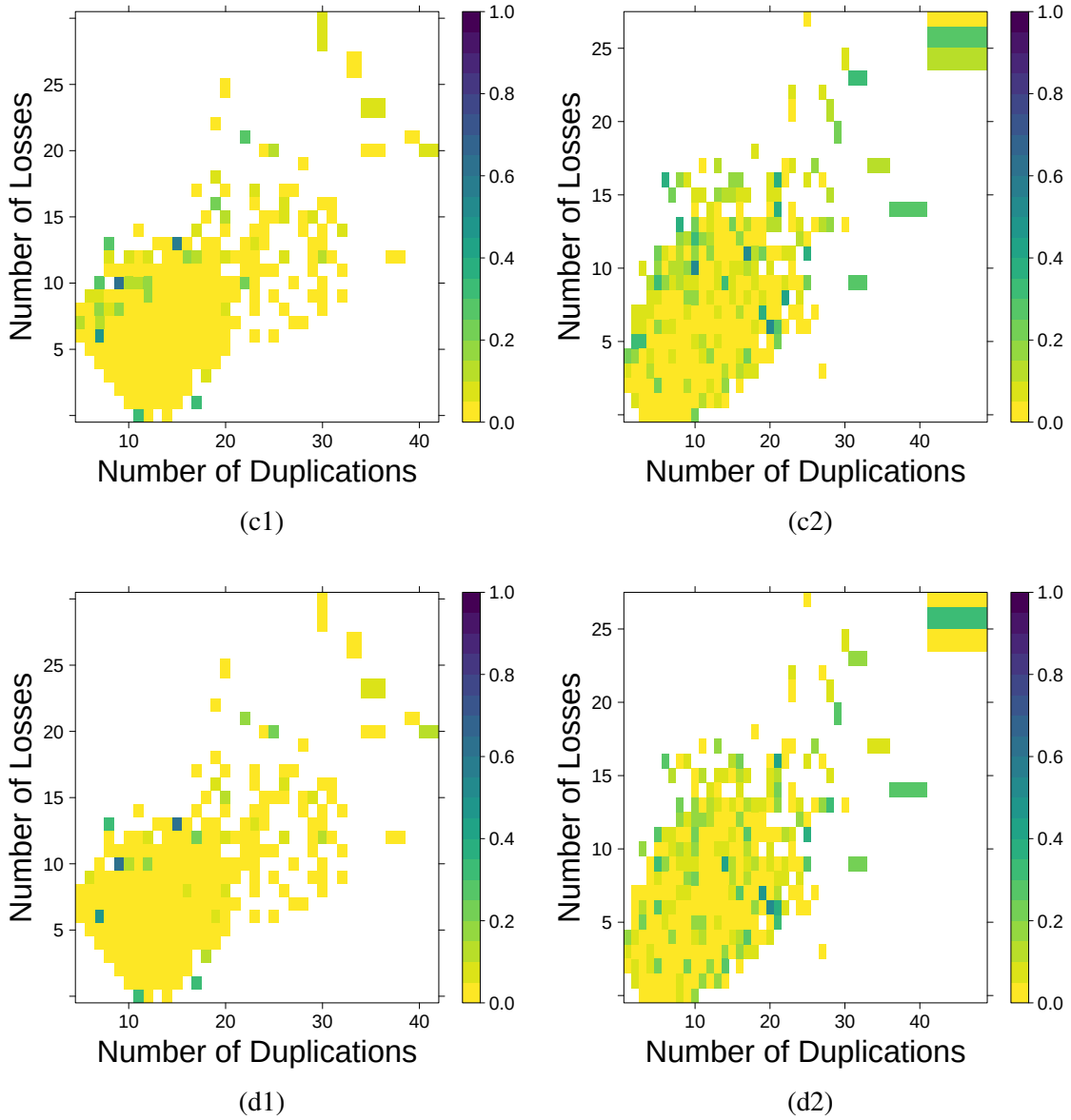


Figure 6.11: Reciprocal Best Match $\vec{G}(T,\sigma)$ edge comparison against the True Orthology Graph Θ for scenarios with HGTs (c1) between 4% and 8% and (c2) between 10 and 50. Corrected P_4 Graph $\vec{G}_{p4}(T,\sigma)$ edge comparison against the True Orthology Graph Θ for scenarios with HGTs (d1) between 4% and 8% and (d2) between 16% and 32%.

Since the removal is performed after all possible P_4 middle edges are identified, we might be removing some true positive edges as well. This can be improved by inverting the sequence of the steps, i.e. we could remove the middle edge of a P_4 inmmediately after being identified and then search

for the next one. This will prevent us from removing more than one edge in a subgraph where only one edge removal is enough to edit a series of P_4's. With this modification to our method, we would expect that the Corrected P_4 Graph $\vec{G}_{p4}(T,\sigma)$ would be more similar to the True Orthology Graph Θ.

On the other hand, as shown in fig.6.11 there is no significant difference between the Reciprocal Best Match $\vec{G}(T,\sigma)$ and the Corrected P_4 Graph $\vec{G}_{p4}(T,\sigma)$.

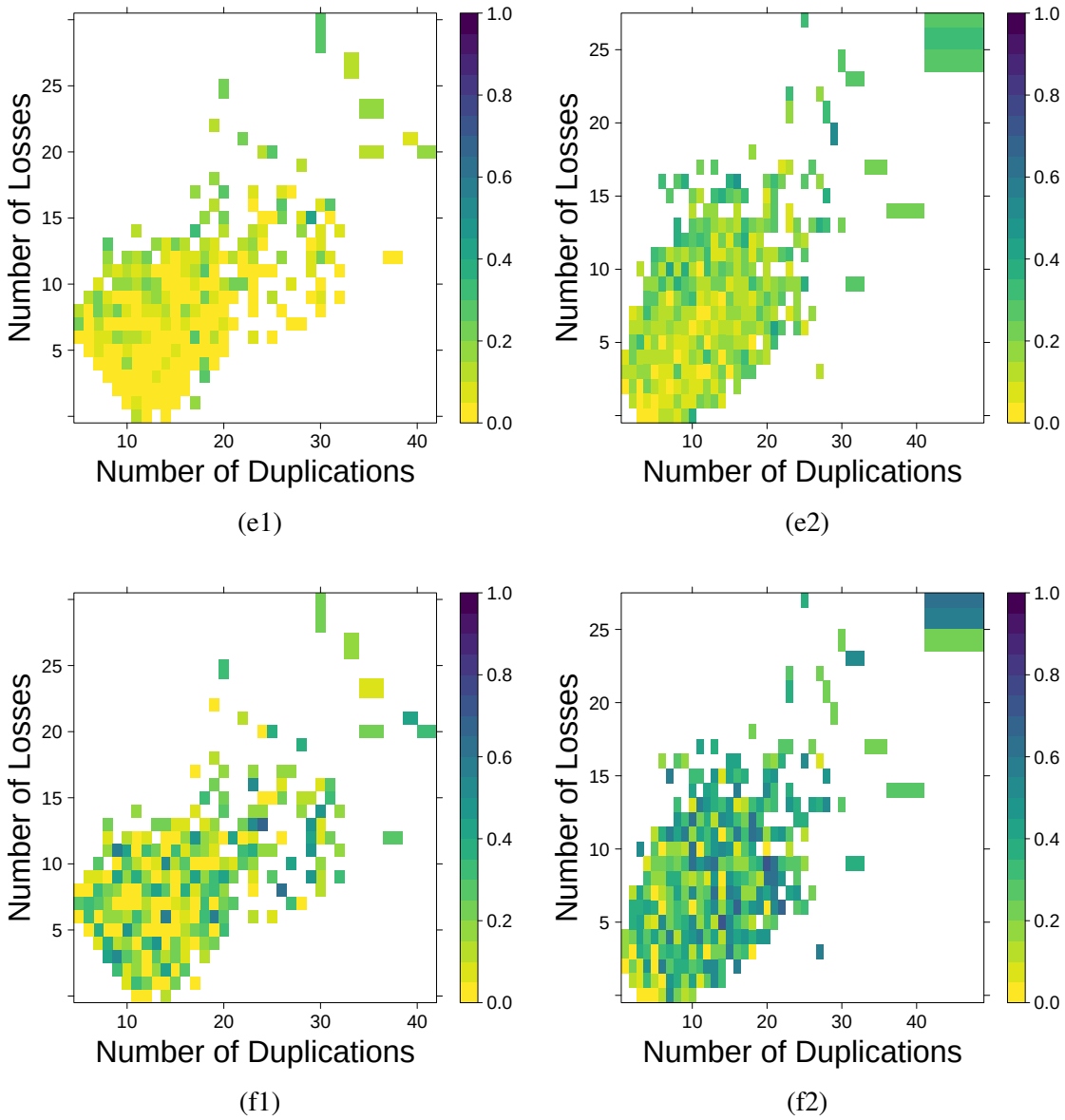


Figure 6.12: Best Match Graph $G(T,\sigma)$ edge comparison against the Reciprocal Best Match Graph $\vec{G}(T,\sigma)$ for scenarios with HGTs (e1) between 4% and 8% and (e2) between 16% and 32%. Reciprocal Best Match Graph $\vec{G}(T,\sigma)$ edge comparison against Corrected P_4 Graph $\vec{G}_{p4}(T,\sigma)$ for scenarios with HGTs (f1) between 4% and 8% and (f2) between 16% and 32%.

Finally, in fig.6.12(e1) and fig.6.12(e2) our results show that there is a predominance of non-symmetrical edges that grows with the number of duplications and losses as well as with the number of HGTs. It is interesting to see, that there is an increase in the number of edited edges from the Reciprocal Best Match $\vec{G}(T,\sigma)$ as shown in fig.6.12(f1) and fig.6.12(f2).

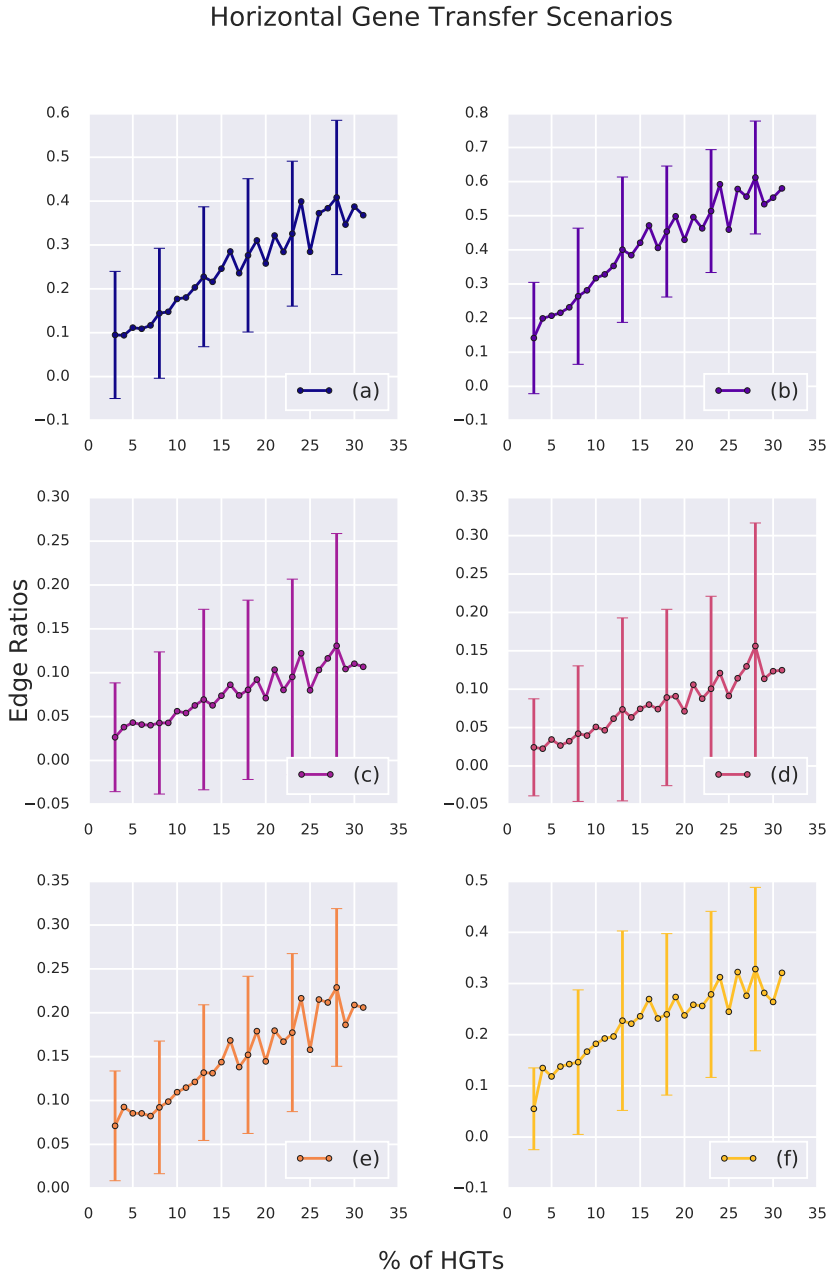


Figure 6.13: Edge Ratio Average and Error along all scenarios with HGTs from 1% to 35%. (a) $|E(\Theta) \setminus E(G(T,\sigma))|/|E(\Theta)|$. (b) $|E(\Theta) \setminus E(\vec{G}_{p4}(T,\sigma))|/|E(\Theta)|$. (c) $|E(\vec{G}(T,\sigma)) \setminus E(\Theta)|/|E(\vec{G}(T,\sigma))|$. (d) $|E(\vec{G}_{p4}(T,\sigma) \setminus E(\Theta))|/|E(\vec{G}_{p4}(T,\sigma))|$. (e) $E(G(T,\sigma)) \setminus E(\vec{G}(T,\sigma))|/|E(G(T,\sigma))|$. (f) $|E(\vec{G}(T,\sigma)) \setminus E(\vec{G}_{p4}(T,\sigma))|/|E(\vec{G}(T,\sigma))$

To further highlight the observed effects of HGTs over the constructed graphs, in fig.6.13 we summarize the obtained scenarios with HGTs > 0. For each percentage of HGTs, we grouped the corresponding scenarios and display the average ratio. Error bars are plotted every fifth value, i.e. the error accounts for the average error of every fifth value of HGT percentge.

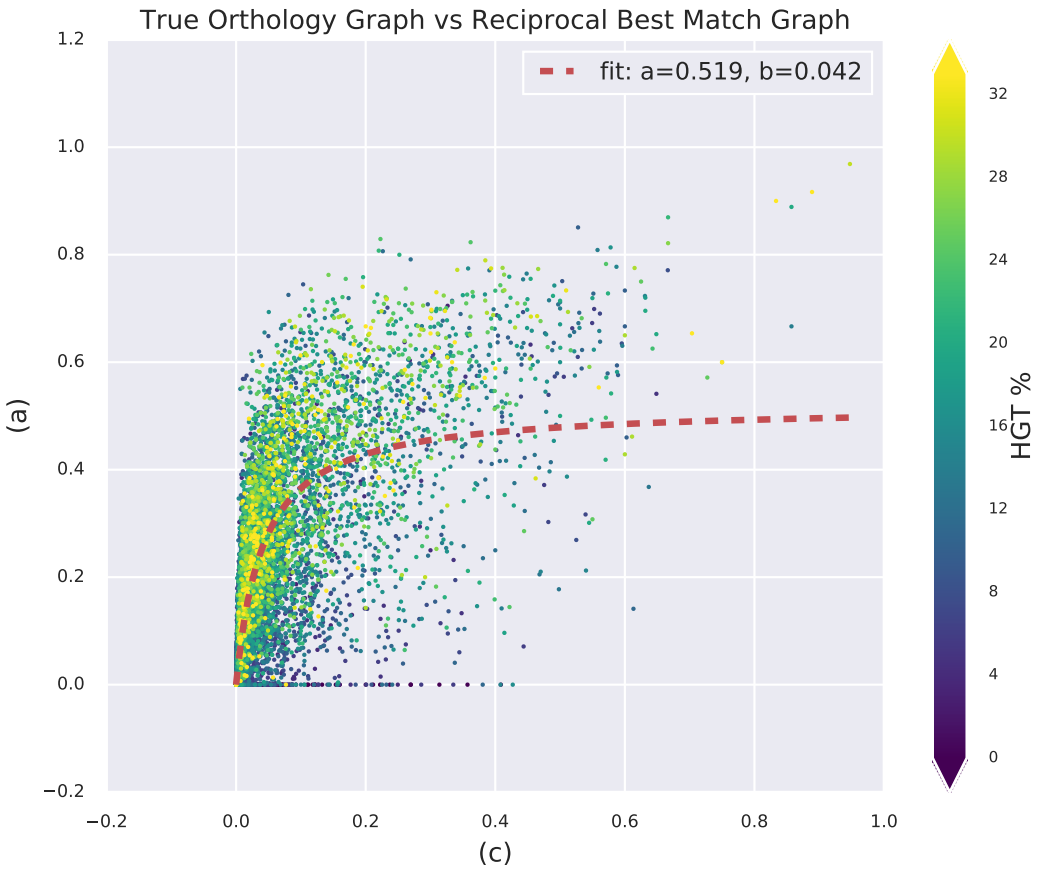


Figure 6.14: Exploratory data analysis of (a) $|E(\Theta) \setminus E(G(T,\sigma))|/|E(\Theta)|$ and (c) $|E(\vec{G}(T,\sigma)) \setminus E(\Theta)|/|E(\vec{G}(T,\sigma))|$. The approximated dashed curve corresponds to the function $f(x) = \frac{ax}{b+x}$ with parameters $a = 0.519$ and $b = 0.042$.

Finally, in figure 6.14 we explore the possible correlation between (a) $|E(\Theta) \setminus E(G(T,\sigma))|/|E(\Theta)|$ and (c) $|E(\vec{G}(T,\sigma)) \setminus E(\Theta)|/|E(\vec{G}(T,\sigma))|$ for all generated scenarios with HGT events ranging from 0% to 32% and visualize the effect of HGTs. We used `scipy.optimize` function `curve_fit` to plot the red approximated dashed curve, which corresponds to:

$$f(x) = \frac{ax}{b+x}$$

with $a = 0.519$ and $b = 0.042$. We see that this shows an increasing non-linear correlation.

6.6 Technical Details

This procedure was implemented in `Python version 3.5` using the following libraries:

- `NetworkX` version 2.2 [38], which is a module that is widely used for studying graphs and networks.
- `Pyvolve` version 0.8.8 [39]. This module allows us to simulate sequence evolution along gene trees continuous-time Markov models of sequence evolution.
- `Matplotlib` version 1.5.3 and `Seaborn` version 0.9.0. These libraries where used to generate most of the plots of the applied case section.

Additionally, the level plots where created using `R version 3.4.4`
This simulation was implemented as separate Python modules and is organized as follows:

Module 1: Species Tree Generation. A species Tree $\hat{S}$ with a dating map is generated.

Module 2: Event Generation. This module distributes the number of events along the edges of the species tree. It also defines the time consistency required for HGT events to happen according to the branch lengths of the Species Tree.

Module 3: True Gene Tree Generator.

Module 4: Observable Gene Tree Generator.

Module 5: Graphs Generator. In this module, the necessary functions to generate the corresponding Orthology Graph, Best Match Graph, Reciprocal Best Match Graph are stored.

Module 6: Sequence Generator.

Module 7: Statistical Analysis Tools.

To ensure an efficient use of space, every graph is stored as an adjacency list representation.

For the computational development of these programs, we used servers provided by the Laboratorio Nacional de Visualización Científica (LAVIS) UNAM.

Chapter 7

Discussion

Through this project, we explored the theoretical and experimental possibility of extracting information regarding the gene and species tree of an evolutionary scenario by just looking at the relations between genes, with the formalized concept of "closely related genes". The study of Best Match Graphs, Reciprocal Best Matches and their relation to the different flavours of homology, remains as an open question, with promising new interdisciplinary challenges to face.

With this project, we successfully fulfilled our goal of creating a computational tool that will help in the exploration of new methods for the graph-based inference of orthology relations, particularly regarding the Best Match heuristic.

The graph generation algorithms presented in this work, though simple, provide a powerful basis for further benchmarking of orthology. Moreover, by using the `NetworkX` library any further graph analysis algorithms can be implemented and added to this pipeline easily.

For this framework, the following enhancements should be considered:

A main drawback of the *innovation model* used for the species tree generation part is that it does not include species extinction events. Thus, the inclusion of a model that contemplates this kind of species event might lead to a more realistic simulation.

In the gene tree generation part, a sequential algorithm was used for the generation of HGT events along the edges of the species tree. Nevertheless, this method fails to efficiently generate this type of event. A parallel strategy can be further implemented to generate HGTs more efficiently.

As for the sequence generation part, `Pyvolve` has not included (yet) the possibility of generating nucleotide insertions and deletions, which are a more realistic representation of sequences, and thus it limits the application of our sequence generation pipeline to its application on testing algorithms where sequence length remains equal for all generated genes.

Although the generated scenarios fulfil our theoretical framework, we must note that in the generated scenarios for our applied study case, smaller number of events are generated more frequently,

which follow a distribution according to the Poisson Distribution with $\lambda = 1$, regardless of the number of leaves of the species tree, as shown in the following figures:

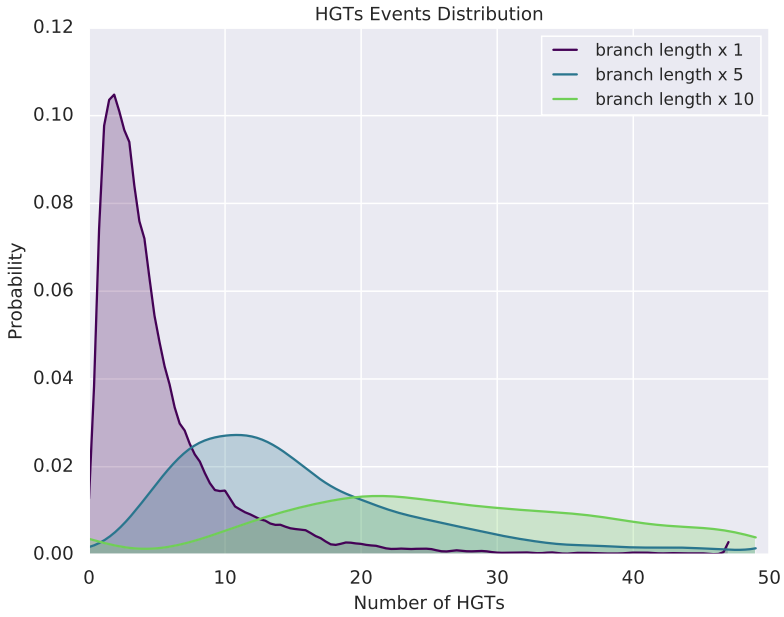


Figure 7.1: HGT events distribution along simulated scenarios with species trees ranging between 3 and 10 species. Branch lengths where multiplied by 1, 5 and 10.

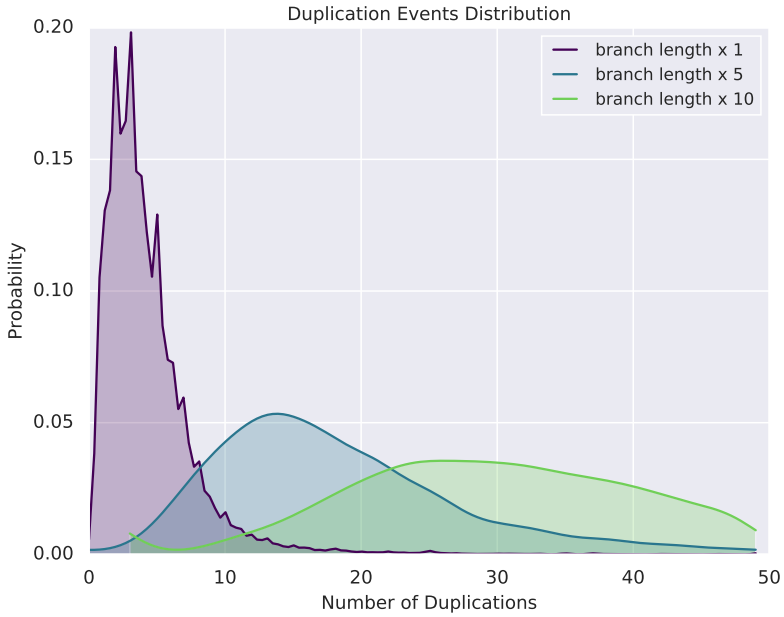


Figure 7.2: Duplication event distribution along simulated scenarios with species trees ranging between 3 and 10. Branch lengths where multiplied by 1, 5 and 10.

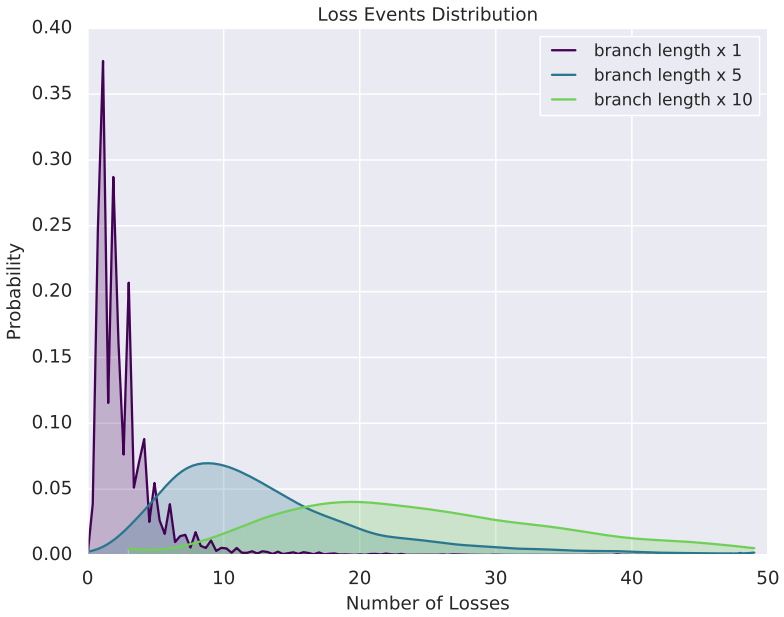


Figure 7.3: Loss events distribution along simulated scenarios with species trees ranging between 3 and 10 species. Branch lengths where multiplied by 1, 5 and 10.

These results could be improved either by multiplying the branch lengths associated to the species tree by a scalar to get higher number of events the implementation of another model for the event assignation of events, such a Gillespie algorithm, which was originally intended for Numerically Simulating the Evolution of Chemical reactions through time. Here our single ancestral gene ("particle" in the Guillespie Algorithm) would evolve. Each event could represent a reaction channel, and with this we would also ensure an exactly dated set of events.

Chapter 8

Concluding Remarks

The aim of this dissertation has been to provide a controlled environment to explore the evolutionary relationships from a graph-based point of view. Graph-based inference of orthology is a promising field that will certainly bring new theoretical basis to aid in the experimental inference of the different flavours of homology.

As we have seen, even though Reciprocal Best Matches are the main basis for graph-based orthology inference methods, their theoretical basis remained unexplored until very recently with the mathematical characterization of Best Match Graphs. This work has been a first approach to a further exploration of the role of Best Match Graphs in an evolutionary context.

Nevertheless, there is still much to learn about Best Match Graphs and Reciprocal Best Match Graphs in an experimental context, with conditions more akin to what we can find in the biological reality.

Chapter 9

Future Work

Besides the usual maintenance, library updates and the inclusion of new models with this stepwise procedure to simulate evolutionary scenarios, the following problems remain as direct follow-ups of the Best Match Graphs project and potential applied study cases for this contribution:

9.1 Best Match Graphs as an ILP Problem

An Integer Linear Programming Model or Integer Optimization Model (ILP-model) is a linear programming model where the values of the decision variables are restricted to be integers. The reason of investigating integer models is that many practical problems require integer solutions. On the other hand, solving practical integer models is usually very complicated and the solution methods require in general excesive high calculation times.

The most widely used solution technique for ILP is the so-called Branch-and-Bound Method: a solution procedure that creates a "tree" of (noninteger) LP models whose solutions "grow" to an optimal integer solution of the initial model.

Some important concepts regarding ILPs are:

- *Decision Variables* Real decision variables, denoted by $x_1, x_2, \cdots x_n$ where n is a finite positive integer.
- *Objective Function* $c_1x_1 + \cdot + sc_nx_n$ is a linear function in the n decision variables with $c_1, \cdots, c_n$ real numbers called *objective coefficients*. Depending on whether the objective function has to be maximized or minimized the *objective* of the model is written as:

$$max(c_1x_1 + \cdots + sc_nx_n) \tag{9.1}$$

  or

$$min(c_1x_1 + \cdots + sc_nx_n) \tag{9.2}$$

- *Constraints* of an LP model is either a $\leq, \geq$ or an $=$ expression of the form:

$$a_{i1}x_1 + \cdots + a_{n1}x_n(\leq, \geq, =)b_i \tag{9.3}$$

where $(\leq, \geq, =)$ means that either the sign $\leq, \geq$ or $=$ holds. The entry a_{ij} is the coefficient of the j-th decision variable x_j in the i-th constraint. Let m be the number of constraints. All the (left-hand sides of the) constraints are linear in $x_1, x_2, \cdots x_n$. For $i \in 1, \cdots, m$ and $i \in 1, \cdots, n$ the entries a_{ij}, b_i and c_j are real numbers and are called *parameters* of the model.

- The *nonnegativities* which is an inequality of the form:

$$x_i \geq 0; \tag{9.4}$$

- The *nonpositivities* which is an inequality of the form:

$$x_i \leq 0; \tag{9.5}$$

The constraints, nonnegativities and nonpositivities are called the *restrictions* of the model.

- The set of vectors satisfying all restrictions of the problem are called *feasible region*.

A two coloured Best Match Graph, as seen in [3] can be characterized in terms of:

- Necessary Conditions (N0-N1). As seen in *theorem 3* of [3].
- Informative triples, which come from the Induced Subgraphs X_1, X_2, X_3, X_4 *theorem 5* of [3].

With this,the goal of this project will be to establish an appropriate ILP model using each identification method, such that we can minimize the symmetric difference between a coloured Digraph Y and Best Match graph G, i.e. an ILP whose objective function is:

$$\sum_{xy}(G_{xy}(1 - Y_{xy}) + (1 - G_{xy})Y_{xy}) \rightarrow \min \tag{9.6}$$

Where:

$$G_{xy} = \begin{cases} 1 & \text{if } (x,y) \in E(G) \\ 0 & \text{otherwise} \end{cases}$$

$$Y_{xy} = \begin{cases} 1 & \text{if } (x,y) \in E(Y) \\ 0 & \text{otherwise} \end{cases}$$

And validate the obtained model with the stepwise procedure presented in this work.

9.2 Inference of Best Matches from Sequence Data

The connection between Best Match Graphs and Orthology relations is not direct. One step towards this connection is to translate the definition of Best Matches in terms of divergence times (the sequence version of evolutionary relatedness.)
In this project, we ask to what extent an additive distance d can be used to infer the best match relation $\rightarrow$ by comparing algorithms and methods to infer best matches from generated sequence data versus the actual best matches.

References

[1] W. M. Fitch, "Distinguishing homologous from analogous proteins.," *Systematic Biology*, vol. 19, no. 2, pp. 99–113, 1970.

[2] D. C. Krakauer, J. P. Collins, D. Erwin, J. C. Flack, W. Fontana, M. D. Laubichler, S. J. Prohaska, G. B. West, and P. F. Stadler, "The challenges and scope of theoretical biology," *Journal of Theoretical Biology*, vol. 276, no. 1, pp. 269–276, 2011.

[3] M. Geiß, E. Chávez, M. González, A. López, B. M. R. Stadler, D. I. Valdivia, M. Hellmuth, M. H. Rosales, and P. F. Stadler, "Best Match Graphs." Available at `https://arxiv.org/abs/1803.10989.`, 2018.

[4] R. L. Kruse, *Data Structures and Program Design in C++*. Prentice Hall, 2000.

[5] M. Geiß, A. López, M. González, E. Chávez, D. I. Valdivia, M. Hernandez-Rosales, M. Hellmuth, and P. F. Stadler, "Best Match Graphs and Reconciliation of Gene Trees with Species Trees." *In preparation.*, 2019.

[6] P. Górecki and J. Tiuryn, "DLS-trees: A model of evolutionary scenarios," *Theoretical Computer Science*, vol. 359, no. 1-3, pp. 378–399, 2006.

[7] A. Brandstadt, *Graph Classes : A Survey*. Philadelphia: Philadelphia : Society for Industrial and Applied Mathematics, siam monog ed., 1999.

[8] E. W. Weisstein, "Cograph." From MathWorld–A Wolfram Web Resource. `http://mathworld.wolfram.com/Cograph.html`.

[9] S. Jia, L. Gao, Y. Gao, J. Nastos, Y. Wang, X. Zhang, and H. Wang, "Defining and identifying cograph communities in complex networks," *New Journal of Physics*, vol. 17, 2015.

[10] W. M. Fitch, "Homology a personal view," *Trends in Genetics*, vol. 16, no. 5, pp. 227–231, 2000.

[11] S. Choudhuri, *Bioinformatics for beginners : genes, genomes, molecular evolution, databases and analytical tools.* London : Academic Press, 2014, 2014.

[12] A. Force, M. Lynch, F. B. Pickett, A. Amores, Y.-l. Yan, and J. Postlethwait, "Preservation of Duplicate Genes by Complementary, Degenerative Mutations," *Genetics*, vol. 151, no. 4, pp. 1531–45, 1999.

[13] M. Ravenhall, N. Škunca, F. Lassalle, and C. Dessimoz, "Inferring Horizontal Gene Transfer," *PLoS Computational Biology*, vol. 11, no. 5, pp. 1–16, 2015.

[14] M. Hellmuth and N. Wieseke, "Construction of Gene and Species Trees from Sequence Data incl. Orthologs, Paralogs, and Xenologs," *arXiv*, vol. 1602, no. 08268v1, pp. 1–19, 2016.

[15] A. Kuzniar, R. C. H. J. van Ham, S. Pongor, and J. A. M. Leunissen, "The quest for orthologs: finding the corresponding gene across genomes," 2008.

[16] R. Dondi, N. El-Mabrouk, and K. M. Swenson, "Gene tree correction for reconciliation and species tree inference: Complexity and algorithms," *Journal of Discrete Algorithms*, vol. 25, pp. 51–65, 2014.

[17] D. A. Dalquen and C. Dessimoz, "Bidirectional best hits miss many orthologs in duplication-rich clades such as plants and animals," *Genome Biology and Evolution*, vol. 5, no. 10, pp. 1800–1806, 2013.

[18] D. G. Kendall, "On the Generalized "Birth-and-Death" Process," *The Annals of Mathematical Statistics*, vol. 19, no. 1, pp. 1–15, 1948.

[19] T. Gernhard, "The conditioned reconstructed process," *Journal of Theoretical Biology*, vol. 253, no. 4, pp. 769–778, 2008.

[20] S. Keller-Schmidt and K. Klemm, "A Model of Macroevolution as a Branching Process Based on Innovations," *Advances in Complex Systems*, vol. 15, no. 07, p. 1250043, 2012.

[21] G. H. Gonnet, M. T. Hallett, C. Korostensky, and L. Bernardin, "Darwin v. 2.0: An interpreted computer language for the biosciences," *Bioinformatics*, vol. 16, no. 2, pp. 101–103, 2000.

[22] D. A. Dalquen, M. Anisimova, G. H. Gonnet, and C. Dessimoz, "ALF-A simulation framework for genome evolution," *Molecular Biology and Evolution*, vol. 29, no. 4, pp. 1115–1123, 2012.

[23] R. G. Beiko and R. L. Charlebois, "A simulation test bed for hypotheses of genome evolution," *Bioinformatics*, vol. 23, no. 7, pp. 825–831, 2007.

[24] Z. Yang, *Computational Molecular Evolution*. Oxford University Press, 2006.

[25] J. Setubal, Joao C. Stoye and P. F. Stadler, *Comparative Genomics*, vol. 2. New York, NY : Springer New York : Imprint: Humana Press, 2018., 2018.

[26] M. Hellmuth, N. Wieseke, M. Lechner, H.-P. Lenhof, M. Middendorf, and P. F. Stadler, "Phylogenomics with Paralogs," *Proceedings of the National Academy of Sciences*, vol. 112, no. 7, 2017.

[27] D. M. Kristensen, Y. I. Wolf, A. R. Mushegian, and E. V. Koonin, "Computational methods for Gene Orthology inference," *Briefings in Bioinformatics*, vol. 12, no. 5, pp. 379–391, 2011.

[28] I. Lobo, "Basic Local Alignment Search Tool (BLAST)," 2008. Nature Education.1(1):215. Available at: https://www.nature.com/scitable/topicpage/basic-local-alignment-search-tool-blast-29096.

[29] D. P. Wall and T. DeLuca, *Ortholog Detection Using the Reciprocal Smallest Distance Algorithm*, pp. 95–110. Totowa, NJ: Humana Press, 2007.

[30] L. Li, C. J. S. Jr, and D. S. Roos, "OrthoMCL : Identification of Ortholog Groups for Eukaryotic Genomes," *Genome Research*, pp. 2178–2189, 2003.

[31] M. Lechner, S. Findeiß, L. Steiner, M. Marz, P. F. Stadler, and S. J. Prohaska, "Proteinortho: Detection of (Co-)orthologs in large-scale analysis," *BMC Bioinformatics*, vol. 12, no. 1, p. 124, 2011.

[32] A. Eyre-Walker, "Evolutionary genomics," *Trends in Ecology and Evolution*, vol. 14, no. 5, p. 176, 1999.

[33] A. M. Altenhoff, N. M. Glover, C. M. Train, K. Kaleb, A. Warwick Vesztrocy, D. Dylus, T. M. De Farias, K. Zile, C. Stevenson, J. Long, H. Redestig, G. H. Gonnet, and C. Dessimoz, "The OMA orthology database in 2018: Retrieving evolutionary relationships among all domains of life through richer web and programmatic interfaces," *Nucleic Acids Research*, vol. 46, no. D1, pp. D477–D485, 2018.

[34] B. T. Nichio, J. N. Marchaukoski, and R. T. Raittz, "New tools in orthology analysis: A brief review of promising perspectives," *Frontiers in Genetics*, vol. 8, no. OCT, pp. 1–12, 2017.

[35] M. Hernandez-Rosales, N. Wieseke, M. Hellmuth, and P. F. Stadler, "Simulation of gene family histories," *BMC Bioinformatics*, vol. 15, no. Suppl 3, p. A8, 2014.

[36] "PHYLIP multiple sequence alignment format." Available at http://scikit-bio.org/docs/0.2.3/generated/skbio.io.phylip.html. [Accessed: 30-Dec-2018].

[37] M. Geiß, M. Hellmuth, and P. F. Stadler, "Reciprocal Best Match Graphs." *Preliminary Draft.*, 2019.

[38] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring Network Structure, Dynamics, and Function using NetworkX," in *Proceedings of the 7th Python in Science Conference* (G. Varoquaux, T. Vaught, and J. Millman, eds.), (Pasadena, CA USA), pp. 11–15, 2008.

[39] S. J. Spielman and C. O. Wilke, "Pyvolve: A flexible python module for simulating sequences along phylogenies," *PLoS ONE*, vol. 10, no. 9, pp. 1–7, 2015.