Combining approaches for predicting genomic evolution

Bassam Alkindy

SPIM

Thèse de Doctorat

école doctorale **sciences pour l'ingénieur et microtechniques**

UFC UNIVERSITÉ DE FRANCHE-COMTÉ

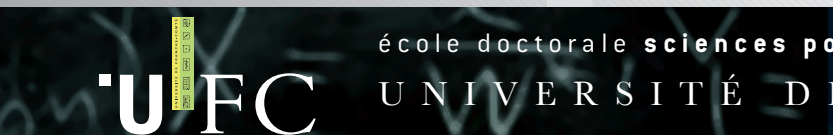
Combining Approaches for Predicting Genomic Evolution

Combinaison d'Approches pour Résoudre le Problème du Réarrangement de Génomes

BASSAM BASIM JAMIL ALKINDY

SPIM

Thèse de Doctorat



N° | X | X | X

Combining Approaches for Predicting Genomic Evolution

Combinaison d'Approches pour Résoudre le Problème du Réarrangement de Génomes

A dissertation presented by

BASSAM BASIM JAMIL ALKINDY

and submitted to the

University of Franche-Comté

in partial fulfillment of the Requirements for obtaining the degree

DOCTOR OF PHYLOSOPHY

in speciality of **Computer Science**

Research Unit :
Laboratory of Femto-ST (SPIM)

Defended in public on 17 December 2015 in front of the Jury composed from :

LHASSANE IDOUMGHAR	President of jury	Professor, University of Haute-Alsace
JEAN-PAUL COMET	Reviewer	Professor, University of Nice
STÉPHANE CHRÉTIEN	Reviewer	Senior Researcher (HDR), National Physical Laboratory Mathematics, Modelling, and Simulation, UK
CHRISTOPHE GUYEUX	Examiner	Professor, University of Franche-Comté
JACQUES M. BAHI	Supervisor	Professor, University of Franche-Comté
JEAN-FRANÇOIS COUCHOT	Co-Supervisor	MCF, University of Franche-Comté
MICHEL SALOMON	Co-Supervisor	MCF, University of Franche-Comté

Acknowledgement

Following this work, I want to express my gratitude to allow all people who have contributed, each has its method, at the completion of this thesis. I want to express my deepest thanks to my supervisor Prof. Jacques M. Bahi and to my co-supervisors Dr. Jean-Francois Couchot and Dr. Michel Salomon. Words are broken me to express my gratitude. Their skills, their scientific rigidity, and clairvoyance taught me a lot. Indeed, I thank them for their organizations, and expert advice provide me they knew throughout these three years and also for their warm quality human, and in particular for the confidence they have granted to me.

I would nevertheless like to thank more particularly proudly Prof. Christophe Guyeux, professor in the University of Franche-Comté, for his advisers and his scientific expert who helped and guided me a lot for the completion of this thesis.

I would like to extend my sincere thanks to Lhassane Idoumghar, Professor at the University of Haute-Alsace, for giving me the honor to president the jury. I also extend my sincere thanks to Jean-Paul Comet, Professor at the University of Nice and Stéphane Chrétien, MCF HDR, National Physical Laboratory Mathematics, Modelling, and Simulation, UK, for giving me the honor of accepting to be rapporteurs of this thesis. I would like to extend my sincere thank to the examiners: Christophe Guyeux, Professor at the University of Franche-Comté and Jean-Francois Couchot, MCF, University of Franche-Comté.

I extend my warmest thanks to the Minister of Higher Education and Scientific Research in Iraq represented by Campus France, the University of Mustansiriyah, and the University of Franche-Comté, for their co-operation to finance this thesis.

My gratitude and my thanks go to the crew members of AND for the friendly and warm atmosphere in which it supported me to work. Therefore thanks to Raphaël Couturier, Karine Deschinkel, Stephane Domas, Giersch Arnaud, Mourad Hakem, Ali Idrees Kadhum, Ahmed Al-Badri, David Laiyamani, Yousra Ahmed Fadil, Abdallah Makhoul, Roxane Mallouhy, Ahmed Mostefaoui, and to whom I missed their names.

My gratitude and my thanks go to the crew members of DISC for the friendly and warm atmosphere in which it supported me to work. Therefore, Thanks Olga Kouchnarenko, director of DISC (Informatique des systèmes complexes) department in Besançon, Pierre-Cyrille Héam, Dominique Menetrier, Jean-Michel Caricand, Laurent Steck and to all other people if I forget his name. For their unwavering support and encouragement.

I would also like express my strongly thanks to the crew of super-computer facilities (Mesocentre) for their generous advices and help in launching the calculations using supercomputer capabilities by installing the modules, creation the site Internet that make dreams come true. Therefore, thanks to Laurent Philippe, Kamel Mazouzi, Guillaume

Laville, and Cédric Clerget.

I would also like express my thanks to my friends in bioinformatics team of Christophe for their kindly friendship, Thanks, Huda AL-NAYYEF, Bashar Al-Nauimi, Panisa Treepong. Before closing, I want to thank my dear friends including Lilia Ziane Khodja, Abbas Abdulhameed, Hamida Bouaziz, Hana M'Hemdi, Lemia Louail, Kitsiri Kizzyy Chochiang, who shared my hopes and studies, which made me comfort in the difficult moments and with whom I shared unforgettable moments of events.

Dedication

To my wife Huda, with my love.

I am also addressing the strongest thanksgiving words to my parents, my wife's parents, my sisters and their families, my brothers and their families, and to my lovely family, for their support and encouragement during the thesis in long years of studies. Their affection and trust lead me and guide me every day. Thank you, Mom, Dad, for making me what I am today.

Abstract

Chloroplasts is one of many types of organelles in the plant cell. They are considered to have originated from cyanobacteria through endosymbiosis, when an eukaryotic cell engulfed a photosynthesizing cyanobacterium, which remained and became a permanent resident in the cell. The term of chloroplast comes from the combination of plastid and chloro, meaning that it is an organelle found in plant cells that contains the chlorophyll. Chloroplast has the ability to convert water, light energy, and carbon dioxide (CO2) in chemical energy by using carbon-fixation cycle (also called Calvin Cycle, the whole process being called photosynthesis). This key role explains why chloroplasts are at the basis of most trophic chains and are thus responsible for evolution and speciation. Moreover, as photosynthetic organisms release atmospheric oxygen when converting light energy in chemical one, and simultaneously produce organic molecules from carbon dioxide, they originated the breathable air and represent a mid to long term carbon storage medium.

Consequently, exploring the evolutionary history of chloroplasts is of great interest, and we propose to investigate it by the mean of ancestral genomes reconstruction. This reconstruction will be achieved in order to discover how the molecules have evolved over time, at which rate, and to determine whether evidences of their cyanobacteria origin can be presented by this way. This long-term objective necessitates numerous intermediate research advances. Among other things, it supposes to be able to apply the ancestral reconstruction on a well-supported phylogenetic tree of a representative collection of chloroplastic genomes. Indeed, sister relationship of two species must be clearly established before trying to reconstruct their ancestor. Additionally, it implies to be able to detect content evolution (modification of genomes like gene loss and gain) along this accurate tree. In other words, gene content evolution on the one hand, and accurate phylogenetic inference on the other hand, must be carefully regarded in the specific case of chloroplast sequences, as the two main prerequisites in our quest of the last universal common ancestor of these chloroplasts.

In detail, given a collection of genomes, it is possible to define their core genes as the common genes that are shared among all the species, while pan genome is all the genes that are present at least once (all the species have each core gene, while a pan gene is in at least one genome). The key idea behind identifying core and pan genes is to understand the evolutionary process among a given set of species: the common part (that is, the core genome) can be used when inferring the phylogenetic relationship, while accessory genes of pan genome explain to some extent each species specificity. In the case of chloroplasts, an important category of genome modification is indeed the loss of functional genes, either because they become ineffective or due to a transfer to the nucleus. Thereby a small number of gene loss among species may indicate that these species are close to each other and belong to a similar lineage, while a large loss means

distant lineages.

More precisely, a key idea concerning phylogenetic classification is that a given DNA mutation shared by at least two taxa has a larger probability to be inherited from a common ancestor than to have occurred independently. Thus shared changes in genomes allow to build relationships between species. In that case, homologous genes are genes derived from a single ancestral one. They are divided in two types, namely paralogous and orthologous genes. Paralogy arises from ancestral gene duplication while the orthologous genes are products of speciation. Being able to understand the way that paralogous and orthologous genes evolve over time should clarify certain aspects of both the chloroplast evolution and origin.

We thus wonder, given a large set of complete chloroplastic genomes, how to find their genes and to determine how they have been acquired or lost during Evolution. Such a knowledge will lead to the ability to reconstruct the ancestral sequences of two sister species, using an algorithm to develop. Applying such an algorithm on a well supported tree will help us to reach the last common universal ancestor of all existing chloroplasts, and finally to study how these genomes have evolved over time.

Table of Contents

CHAPTER 1

Introduction

1.1/ General Presentation

Chloroplasts are one of the main organelles in plant cell. They are considered to have originated from cyanobacteria through endosymbiosis, when an eukaryotic cell engulfed a photosynthesizing cyanobacterium, which remained and became a permanent resident in the cell. The term of chloroplast comes from the combination of plastid and chloro, meaning that it is an organelle found in plant cells that contains the chlorophyll. Chloroplast has the ability to convert water, light energy, and carbon dioxide (CO_2) in chemical energy by using carbon-fixation cycle [1] (also called *Calvin Cycle*, the whole process being called photosynthesis). This key role explains why chloroplasts are at the basis of most trophic chains and are thus responsible for evolution and speciation. Moreover, as photosynthetic organisms release atmospheric oxygen when converting light energy in chemical one, and simultaneously produce organic molecules from carbon dioxide, they originated the breathable air and represent a mid to long term carbon storage medium.

Consequently, exploring the evolutionary history of chloroplasts is of great interest, and *we propose to investigate it by the mean of ancestral genomes reconstruction*. This reconstruction will be achieved in order to discover how the molecules have evolved over time, at which rate, and to determine whether evidences of their cyanobacteria origin can be presented by this way. This long-term objective necessitates numerous intermediate research advances. Among other things, it supposes to be able to apply the ancestral reconstruction on a well-supported phylogenetic tree of a representative collection of chloroplastic genomes. Indeed, sister relationship of two species must be clearly established before trying to reconstruct their ancestor. Additionally, it implies to be able to detect content evolution (modification of genomes like gene loss and gain) along this accurate tree. In other words, *gene content evolution* on the one hand, and *accurate phylogenetic inference* on the other hand, must be carefully regarded *in the specific case of chloroplast sequences*, as the two main prerequisites in our quest of the last universal common ancestor of these chloroplasts.

In detail, given a collection of genomes, it is possible to define their core genes as the common genes that are shared among all the species, while pan genome is all the genes

that are present at least once (*all* the species have each core gene, while a pan gene is in *at least one* genome). The key idea behind identifying core and pan genes is to understand the evolutionary process among a given set of species: the common part (that is, the core genome) can be used when inferring the phylogenetic relationship, while accessory genes of pan genome explain to some extent each species specificity. In the case of chloroplasts, an important category of genome modification is indeed the loss of functional genes, either because they become ineffective or due to a transfer to the nucleus. Thereby a small number of gene loss among species may indicate that these species are close to each other and belong to a similar lineage, while a large loss means distant lineages.

More precisely, a key idea concerning phylogenetic classification is that a given DNA mutation shared by at least two taxa has a larger probability to be inherited from a common ancestor than to have occurred independently. Thus shared changes in genomes allow to build relationships between species. In that case, homologous genes are genes derived from a single ancestral one. They are divided in two types, namely paralogous and orthologous genes. Paralogy arises from ancestral gene duplication while the orthologous genes are products of speciation. Being able to understand the way that paralogous and orthologous genes evolve over time should clarify certain aspects of both the chloroplast evolution and origin.

We thus wonder, given a large set of complete chloroplastic genomes, how to find their genes and to determine how they have been acquired or lost during Evolution. Such a knowledge will lead to the ability to reconstruct the ancestral sequences of two sister species, using an algorithm to develop. Applying such an algorithm on a well supported tree will help us to reach the last common universal ancestor of all existing chloroplasts, and finally to study how these genomes have evolved over time.

1.2/ PRESENTATION OF THE PROBLEMS

Understanding the evolution of DNA molecules is an open and complex problem.

Algorithms have been proposed to tackle this problem, but either they are limited to the evolution of one given character (for instance, a specific nucleotide), or conversely they theoretically focus on large scale nuclear genomes (several billions of nucleotides) facing multiple recombination events. One-character methods cannot be extended to large scale genomic evolution, while it is well known that the problem is NP hard when considering the set of all possible recombination on large genomes. So no concrete solution exists at present regarding the evolution of large DNA sequences. However, in this thesis, we focus on genomes that have a reasonable size and who faced a reasonable number of recombination. This is why we argue that the problem may be tractable in the chloroplast case – but it requires the design of *ad hoc* solutions, and various difficulties still remain to circumvent when dealing with such a specificity.

First, the evolution history of chloroplasts can only be inferred on shared coding sequences, which are difficult to extract. Indeed, no tool is available to find the core genes, and so bioinformatics investigations using sequence annotation and comparison tools are required to be able to determine the core of chloroplast genomes for a given set of photosynthetic organisms. Additionally, the amount of completely sequenced chloroplast genomes increases rapidly, leading to the possibility to build large-scale phylogenies that

represent well the plant diversity. But the size of the core genome is dramatically reduced when we consider very divergent plant species, which explain why these phylogenies are usually done using a small number of chloroplastic genes. In that case, we can wonder if we deal with a gene tree or a species one, and the obtained phylogeny is probably not accurate enough to deploy an ancestral reconstruction on it.

It is true that, if we are able to automatically consider various subsets of close plants defined according to their chloroplasts, some phylogenetic trees may be inferred on larger sets of core genes. But these trees are not necessarily well supported, due to the possible occurrence of homoplasic genes that may blur phylogenetic signals: a trustworthy phylogenetic tree can still be obtained only if the number of homoplasic genes is low, the problem becoming to *determine the largest subset of core genes that produces the most supported tree*. Furthermore, the way to merge such a forest of phylogenetic trees into only one supertree is not obvious.

Finally, given an accurate phylogenetic tree whose leaves contain well annotated genomes, the way to reconstruct node by node each ancestor until the last common one still remains unclear.

1.3/ THESIS OBJECTIVE

The objective of this thesis is to explore the possibility to reconstruct the last universal common ancestor (LUCA) of all available chloroplastic genomes, and to compare it with the ancestor of current cyanobacterial genomes. It is not demanded to give a definitive answer to this ambitious question, but to investigate scientific and technical obstacles that may potentially appear when trying to reach such a difficult goal.

In other words, considering available a black box receiving as input a large set of complete chloroplast genomes, and which produces LUCA as output, the thesis objective is to detail the general functioning of such a magic box. We must not only emphasize all difficulties that can possibly occur when trying to reach such an objective, but also be able to provide intermediate scientific stages. Having such a knowledge or feeling that particular points may raise difficulties, first elements of response to such putative difficulties should be provided.

This ancestral reconstruction can be achieved in 3 stages. Firstly, after having obtained a large collection of complete chloroplastic genomes, we must be able to extract their coding sequences. Using the genes shared in common by these species, a well-supported phylogenetic tree must be obtained. In case where the core genome of the whole species is too much small, a strategy grouping subsets of sequences according to their similarity, inferring their phylogenies, and then merging all the forest of trees, must be investigated. Secondly, algorithms that study the evolution of gene content and ordering among the supertree must be provided, and it must be validated with naked eye on well chosen plant families. Finally, ancestral nucleotide sequence of each gene must be obtained, and intergenic regions must be filled using either state of the art or novel algorithms.

Again, it is not demanded to give a final response to this very ambitious question, but to emphasize scientific and technical problems, and to provide first proposals to solve them.

1.4/ CONTRIBUTIONS

As stated previously, the main subject of this thesis is to investigate the evolution dynamics of DNA sequences contained in chloroplastic organelles (plant cells), using the state of the art or new bioinformatics intelligent algorithms that must be developed.

We have investigated in particular the problem of chloroplast annotations and of core gene extraction. Given a large set of common genes, the way to find a core subset as large as possible leading to a phylogenetic tree as supported as possible has been investigated too, using genetic algorithm and particle swarm optimization. Effects of gene selection on topology and supports has been regarded too by the mean of up to date statistical tests.

These algorithms can be applied in a distributed pipeline that automatically extracts a subset of 10 up to 20 close genomes from a collection of approximately 500 chloroplasts, annotates them with accuracy, and produces a well supported tree using the largest possible subset of core genes. The way to merge such a forest in a supertree has been regarded too, but this problem is not currently fully resolved. Finally, a first gene content and order ancestral reconstruction has been proposed and compared with manual reconstruction on various families of plants.

1.5/ PUBLICATIONS

Our contributions has led to various communications in both conferences and journals, which are listed thereafter.

1.5.1/ ACTS OF SELECTIVE INTERNATIONAL CONFERENCES

1. **ICBBS'2014** Bassam Alkindy, Jean-François Couchot, Christophe Guyeux, Arnaud Mouly, Michel Salomon, and Jacques Bahi. *Finding the Core-Genes of Chloroplasts*. 3rd Int. Conf. on Bioinformatics and Biomedical Science, number 4(5) of IJBBB, Journal of Bioscience, Biochemistery, and Bioinformatics, Copenhagen, Denmark, pages 357–364, June 2014.

2. **BIBM'2014** Bassam Alkindy, Christophe Guyeux, Jean-François Couchot, Michel Salomon, and Jacques Bahi. *Gene Similarity-based Approaches for Determining Core-Genes of Chloroplasts*. IEEE International Conference on Bioinformatics and Biomedicine, pages 71–74, Belfast, United Kingdom, November 2014.

3. **IWBBIO'2015** Bassam Alkindy, Huda Al-Nayyef, Christophe Guyeux, Jean-François Couchot, Michel Salomon, and Jacques Bahi. *Improved Core Genes Prediction for Constructing well-supported Phylogenetic Trees in large sets of Plant Species*. 3^{rd} Int. Work-Conference on Bioinformatics and Biomedical Engineering, Springer, volume 9043 of LNCS, Granada, Spain, pages 379–390, April 2015

4. **AICoB'2015** Bassam Alkindy, Christophe Guyeux, Jean-François Couchot, Michel Salomon, Christian Parisod, and Jacques Bahi. *Hybrid Genetic Algorithm and Lasso Test Approach for Inferring Well Supported Phylogenetic Trees based on Subsets of Chloroplastic Core Genes*. 2^{nd} International Conference on Algorithms

for Computational Biology, volume 9199 of LNCS/LNBI, Mexico City, Mexico, August 2015. Springer. Note: To appear in the LNCS/LNBI series.

5. **CIBB'2015** Reem Alsrraj, Bassam AlKindy, Christophe Guyeux, Laurent Philippe, and Jean-François Couchot. *Well-supported phylogenies using largest subsets of core-genes by discrete particle swarm optimization*. Preceedings of 12^{th} International meeting on Computational Intelligence methods for Bioinformatics and Biostatistics (CIBB), Naples, Italy, vol. 2, p. 1–6, September 2015.

1.5.2/ PUBLICATIONS IN NATIONAL SEMINARS AND WORKSHOPS

1. **SeqBio'2013** Bassam Alkindy, Jean-François Couchot, Christophe Guyeux, and Michel Salomon. *Finding the core-genes of Chloroplast Species*. Workshop of SeqBio 2013, Montpellier, November 2013.

2. **Femto-st'2014** Bassam Alkindy, Huda Al'Nayyef, Jean-François Couchot, Christophe Guyeux, Michel Salomon, and Jacques Bahi. *Algorithmics Genomic Evolution: Insertion Sequences and Core Genomes*. Workshop of Femto-ST, June 2014, Besancon, France. Note: Poster.

3. **Femto-st'2015** Bassam Alkindy, Huda Al'Nayyef, Panisa Treepong, Bashar Al-Nuaimi, Christophe Guyeux, Jean-François Couchot, Michel Salomon, and Jacques Bahi. *Bioinformatics Approaches on Genomic Evolution in Femto-ST (Core Genome, Phylogenetic Analysis, Transposable Elements, and Ancestral Reconstruction)*. Workshop of Femto-ST, June 2015, Besancon, France. Note: Poster.

4. **MCEB'2015** Bassam Alkindy, Christophe Guyeux, Jean-François Couchot, Michel Salomon, and Jacques Bahi. *Using Genetic Algorithm for Optimizing Phylogenetic Tree Inference in Plant Species*. In MCEB15, Conference of Mathematical and Computational Evolutionary Biology, Porquerolles Island, France, June 2015. Note: Poster.

5. **SeqBio'2015** Bashar Al-Nuaimi, Roxane Mallouhi, Bassam AlKindy, Christophe Guyeux, Michel Salomon, and Jean-François Couchot. *Ancestral reconstruction and investigations of genomic recombination on Campanulides chloroplasts*. Workshop of SeqBio 2015, Orsay, November 2015.

1.6/ LIST OF ABBREVIATIONS

Abbreviation	Description
BLAST	Basic Local Alignment Search Tool.
BP	Bootstrap Probability.
CC	Connected Component.
CEGMA	Core Eukaryotic Genes Mapping Approach.
CpBase	The Chloroplast Genome Database.
CpGAVAS	Chloroplast Genome Annotation, Visualization, Analysis and GenBank Submission Tool.
DDBJ	DNA Data Bank of Japan.
DNA	Deoxyribonucleic Acid.
DOGMA	Dual Organellar GenoMe Annotator.
DBLT	Dummy Binary Logit Test.
DPSO	distributed Particle Swarm Optimization.
EMBL	European Molecular Biology Laboratory.
FPE	False Positive Error.
FNE	False Negative Error.
GA	Genetic Algorithm.
GSA	Global Sequence Alignment.
ICM	Intersection Core Matrix.
IS	Intersection Score.
LASSO	Least Absolute Shrinkage and Selection Operator Test.
LGI	Lowest Number of Ignored Genes.
LSA	Local Sequence Alignment.
MGI	Maximum Number of Ignored Genes.
MSA	Multiple Sequence Alignment.
MUSCLE	MUltiple Sequence Comparison by Log- Expectation.
ML	Maximum Likelihood.
NCBI	National Center of Biotechnology Information.
NW	Needle-man Wunsch Alignment.
Occ.	Number of Tree Occurences.
PSA	Pairwise Sequence Alignment.
PSO	Particle Swarm Optimization.
RAxML	Randomized Axelerated Maximum Likelihood.
RNA	Ribonucleic acid.
SH	Shimodaira-Hasegawa Algorithm.
SW	Smith-Waterman Alignment.
rRNA	ribosomal RNA.
T-COFFEE	Multiple sequence alignment that provides a dramatic improvement in accuracy with a modest sacrifice in speed as compared to the most commonly used alternatives.
Topo.	Topology Number.
tRNA	Transfer RNA.
WSH	Weighted Shimodaira-Hasegawa Algorithm.

1.7/ MATHEMATICAL NOTATIONS

Symbol	Description
A	is the nucleotide alphabet.
A^*	the set of finite words on A.
$\mathcal{R}$	equivalent relation.
$s_1, ..., s_k$	finite sequence of vertices (DNA sequences).
$d : N = A^* \times A^* \rightarrow [0, 1]$	A function of similarity measure on A^*.
w	the binary word.
w'_i	new word generated after specific event (ex., mutation).
s'	subset of core genome.
b	bootstrap value.
p	percentage of gene presents.
p'	the number of 1's in w.
p_{val}	p-value.
P	set of population.
P'	New population generated from P.
P_c	Population generated from crossover stage.
P_m	Population generated from mutation stage.
P_r	populaton having lessthan 10% of 0's.
T'	is the list of phylogenetic trees.
W'	the set of topologies.
lb	the lower bound threshold.
c	the set of core genes.
$\|c\|$	the length of core genome.
m'	is the size of T.
$N_{mutation}$	amount of mutations.
$N_{crossover}$	amount of crossover.
2^n	phylogenetic tree inferences for a core genome of size n.
$(X_1, X_2, \ldots, X_n)$	Positions of n particles vectors.
$(V_1, V_2, ..., V_n)$	particles associated velocities, which are N-dimensional vectors of real numbers between 0 and 1.

1.8/ ORGANIZATION OF THE THESIS MANUSCRIPT

The current chapter is devoted to a general introduction of the thesis, providing the problematics and a brief description of thesis subject and objectives. Then the thesis manuscript is organized in three parts.

In the state of the art, Part I, three chapters detail a small overview of main background aspects in bioinformatics domain employed in this manuscript, like sequence alignments and phylogenetic analysis, etc. Some available tools are provided too. In details, a state-of-the-art in core and pan gene extraction is outlined in Chapter 2. The concepts of local and global alignments are detailed in Chapter 3, by giving examples of most common alignment algorithms used in this field. Multiple alignment algorithms are detailed too, and we explain why small divergences in given sequences can lead to a hard alignment problem. To analyse aligned sequences, in Chapter 4, we will detail various phylogenetic concepts like rooted or unrooted trees. Methods for constructing phylogenetic trees are

also summarized (such as distance and character based methods), together with bootstrap analysis.

Part II starts with an introduction that explains the importance of discovering core and pan genes (Chapter 5). The way to distinguish the rooted and sub-rooted ancestor genomes, and to understand their impact on the genomic recombination in Eukaryotes is detailed too. Secondly, three pipelines for the discovery of core and pan genes of chloroplast sequences are presented in Chapter 6. The next chapter 7 details the use of an artificial intelligence algorithm for phylogenetic tree reconstruction. It is based on genetic algorithm while, in Chapter 8, a new pipeline for constructing phylogenetic trees with best subsets of core genes is presented. It uses a particle swarm optimization approach that is developed in both linear and parallel fashions, in order to reconstruct the phylogenetic tree. Then, in the following chapter, a comparison between genetic algorithm and particle swarm optimization is outlined in parallel version, by focusing on 12 groups of chloroplasts. In Chapter 9, a predefined ad-hoc algorithm for generating ancestor genomes is finally detailed, depending on all provided information obtained with previously detailed tools.

This manuscript ends with Part III, which contains a conclusion and some perspectives.

I

State of the Art

CHAPTER 2

A short history regarding core and pan genome extraction

Let us start by presenting some examples of core and pan gene extraction that can be found in the state of the art. Note that we oddly have found only a few articles dealing with such a problem, during our review of the literature.

An early study about finding the common genes in chloroplasts has been realized by *Stoebe et al.* in 1998 [2]. They established the distribution of 190 identified genes and 66 hypothetical protein-coding genes (*ysf*) in all nine photosynthetic algal plastid genomes available (excluding non-photosynthetic *Astasia tonga*) from the last update of plastid genes nomenclature and distribution. The distribution reveals a set of approximately 50 core protein-coding genes retained in all taxa. In 2003, *Grzebyk et al.* [3] have studied the core genes among 24 chloroplastic sequences extracted from public databases, 10 of them being algae plastid genomes. They broadly clustered the 50 genes from *Stoebe et al.* into three major functional domains: (1) genes encoded for ATP synthesis (*atp* genes); (2) genes encoded for photosynthetic processes (*psa* and *psb* genes); and (3) housekeeping genes that include the plastid ribosomal proteins (*rpl* and *rps* genes). The study shows that all plastid genomes were rich in housekeeping genes with one *rbcLg* gene involved in photosynthesis.

Another example of the extraction of core genome can be found in 2009 by *Sharon* [4], where he focused on photosynthetic productivity in *Synechococcus* and *Prochlorococcus* (*Cyanobacteria*) to extract the core genome. He successfully identified the core genes of photosystem II in *Cyanophage* as functional genes for photosynthesis process; then he increased the viral fitness by supplementing the host production of a specific type of proteins. The study also proposed an evidence of the presence of photosystem I genes in the genomes of viruses that affect *Cyanobacteria*.

In 2014, *De Chiara et al.* [5] aligned a collection of 97 sequenced genomes to a reference, the complete genome of the *Haemophilus influenza* strain 86-028NP, using the Nucmer alignment program [6]. They generated a list of polymorphic sites with these alignments. This list was then filtered to include only the polymorphic sites in the core genome of NTHi, *i.e.*, the regions of the reference strain that could be aligned against all other strains, yielding a set of 149,214 SNPs. A clustering algorithm has been finally used on these SNPs to achieve the core genes extraction.

Many studies have then realized the extraction of core and pan genomes for *bacteria* (such as *Cyanobacteria*) using NCBI annotations, which are mainly based on generic annotation tools like Glimmer, MuMmer, RATT, or RAST (see [7]). Then, NTHi strains selected for genome sequencing (dataset S1) were obtained from a collection of isolates archived in Oxford.

In all of these studies, considered genomes have been annotated with various different annotation algorithms, mixing human curated and automatic coding sequence prediction tools that are not specific to chloroplastic genes. This large variety of manners to detect coding sequences and their functionality leads to large variability in gene boundaries (start and stop codons), which obviously severely biases the core and pan genomes determination.

Let us now present, in the next chapter, various methods for aligning biological sequences by using local and global alignment techniques (the last chapter of this part will focus on phylogenetic reconstruction).

CHAPTER 3

Technical Aspects of Sequence Alignments

In this chapter, we will introduce different sequence alignment algorithms. We will adopt an evolutionary perspective in our description of how amino-acids (or nucleotides) in two sequences can be aligned and compared. We will then describe various local and global alignment algorithms and programs for single and multiple alignment manners.

3.1/ INTRODUCTION

In bioinformatics, sequence alignment (or Pairwise Sequence Alignment (PSA)) is an important stage for aligning and comparing DNA sequences. It can be seen as the fundamental procedure that can be implicitly or explicitly applied in any biological research that compares two or more sequences (DNA, RNA, or protein). It is the procedure by which one attempts to infer which positions (sites) within sequences are homologous, that is, which sites share a common evolutionary history [8].

We need first to give some definitions for some important keywords such as: homology, similarity, and identity. We recall the definition of these keywords from [9, 10]:

Definition 1: Homology

Two sequences are said to have a *homologous* relation, if they share a common evolutionary ancestor.

It is clear to say that there are no degree of homology, sequences are either homologous or not. Homologous protein sequences can be *Orthologus*: homologous sequences in different species that arose from a common ancestral gene during speciation. Orthologous genes have similar biological functions [10].

Definition 2: Similarity

Two sequences are said to be similar, if it is possible to transform the first one in the second one by using only a small number of edit operations (insertion, deletion, and substitution).

Definition 3: Sequence identity

Sequence identity between two different sequences is the amount of characters that match exactly when comparing them pairwise. This is a percentage.

It is important to notice that sequence identity is not transitive, in the meaning that sequences S_A and S_B on the one hand, S_B and S_C on the other hand, can have a high identity while it is not the case between S_A and S_C. For example:

Example 1: Sequence identity vs transitivity

Let S_A = AAGCCTT, S_B = AAGCC, and S_C = AAGCCTA respectively, and S_I be the function that produces the identity score between two sequences. This identity is computed by counting the number of matching characters between two sequences divided on the minimum length of given sequences, multiplied by 100:

$$S_I(S_A, S_B) = \frac{5}{min(7,5)} \times 100 = 100\%$$

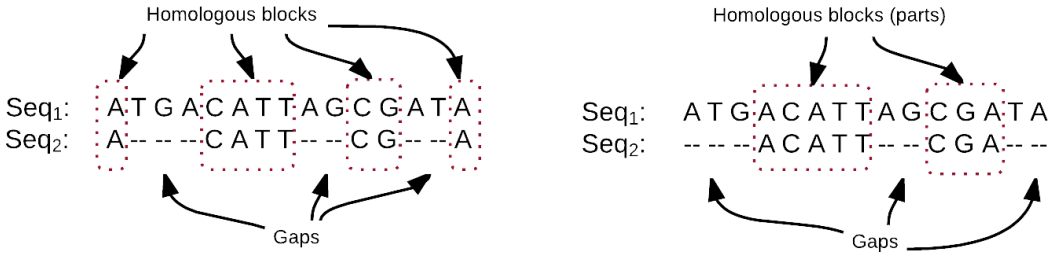
$$S_I(S_B, S_C) = \frac{5}{min(5,7)} \times 100 = 100\%$$

$$S_I(S_A, S_C) = \frac{6}{min(7,7)} \times 100 = 85.7\%$$

In a computer science perspective, PSA is simply a pattern matching problem. The goal is to find the minimum edit distance between two given strings. Some algorithms applied for this task achieved to align strings in non-linear time and/or memory consuming, specially for large strings. In 1973, for instance, Peter Weiner [11] proposed a linear algorithm to find the maximum pattern matching score between two strings in linear time. However he did not success to write a powerful matching algorithm running in less than $O(n^2)$ and some string operations (such as, insertion, deletion, etc.) were not taken into account. This is why, in 1985, Esko Ukkonen [12] presented a string matching algorithm by considering three string operations:

1. Deletion: remove symbol $a \in \sum$ from position i, where $\sum$ is a given alphabet.
2. Insertion: insert a symbol $b \in \sum$ in position i.
3. Substitution: replace a symbol a in position i by a symbol $b \in \sum$ in the same position.

The alphabet $\sum$ in above edit operations is constituted by strings (including alphabets, numbers, and/or special characters). But, in bioinformatics, it is composed by four nitrogenous base characters when considering DNA, that is: $\sum = \{\mathcal{A}, \mathcal{T}, \mathcal{C}, \mathcal{G}\}$. So the pattern matching algorithms developed for strings cannot directly be applied to DNA sequences, as both symbols and positions has biological meaning. Thus specific algorithms need to be developed by taking into account the particularity of DNA "edit operations". For instance, deleting a part of the molecule should have approximately the same cost for small or large part, as it corresponds to only one chemical operation. Considering edit distances in the case of DNA sequences leads to two kinds of alignment algorithms: either globally align two DNA sequences as shown in Figure 3.1(a), or find the best local alignment as depicted in Figure 3.1(b).



(a) Global alignment matching.

(b) Local alignment matching.

Figure 3.1: demonstration of sequence alignment approaches. (a) The process of global alignment. (b) The process of local alignment.

In other words, in global alignments, the entire (protein or nucleotide) sequences are aligned, while a local alignment concentrates the search on the regions of highest similarities within two sequences. In Figure 3.1, we can see in the two examples that DNA sequences have different sizes. It is remarkable that after applying global or local alignment algorithm, the two sequences have the same size, which is the largest one in the global alignment case, and the size of the best common subpattern in the local one. Each column in these two sequences is called a *site*. *Homologous sites* are columns in aligned sequences where characters are equal. For instance, in Figure 3.1(a) that represent the global alignment of sequence S_1 with sequence S_2, we have four homologous blocks of eight homologous sites distributed along S_1.

Gaps in both figures indicate the non-matching sites due to an insertion or deletion of k elements. The most accurate matching algorithms are those that consider an "opening gap" penalty in its scoring function, and all these alignment algorithms need to evaluate the cost of a substitution, either for nucleotides (A, T, C, G alphabet) in DNA alignment, or for amino acids (20 letters) in the protein case. This need to attribute a cost to a substitution leads to the introduction of the standard substitution matrices of both nucleotides and amino acids.

3.2/ STANDARD SUBSTITUTION MATRICES

3.2.1/ NUCLEOTIDE SUBSTITUTION MATRICES

Codons are not uniformly distributed in the genome. Over time, mutations have introduced some variations in their frequency of apparition. It can be attractive to study the genetic patterns (blocs of more than one nucleotide: dinucleotides, trinucleotides...) that appear and disappear depending on mutation parameters. Mathematical models allow the prediction of such an evolution, in such a way that statistical values observed in current genomes can be recovered from hypotheses on past DNA sequences. A first model for genome evolution was proposed in 1969 by Thomas Jukes and Charles Cantor [13]. This first model is very simple, as it supposes that each nucleotide A, C, G, T has the probability m to mutate to any other nucleotide, as described in the following mutation

matrix,

$$\begin{pmatrix} 1-3m & m & m & m \\ m & 1-3m & m & m \\ m & m & 1-3m & m \\ m & m & m & 1-3m \end{pmatrix}$$

In this matrix, the coefficient in row 3, column 2 represents the probability that the nucleotide G mutates in C during the next time interval, *i.e.*, $P(G \to C)$. This first attempt has been followed up by Motoo Kimura [14], who has reasonably considered that transitions ($A \longleftrightarrow G$ and $T \longleftrightarrow C$) should not have the same mutation rate as transversions ($A \longleftrightarrow T$, $A \longleftrightarrow C$, $T \longleftrightarrow G$, and $C \longleftrightarrow G$), leading to the following mutation matrix.

$$\begin{pmatrix} 1-a-2b & b & a & b \\ b & 1-a-2b & b & a \\ a & b & 1-a-2b & b \\ b & a & b & 1-a-2b \end{pmatrix}$$

This model was refined by Kimura in 1981 (three constant parameters, to make a distinction between natural $A \longleftrightarrow T$, $C \longleftrightarrow G$ and unnatural transversions), Joseph Felsenstein, Masami Hasegawa, Hirohisa Kishino, Taka-Aki Yano [15], and so on. Up to date mutation models encompass the General Time Reversible (GTR, [16], 1990), Tamura-Nei (TrN) in 1993 [17], or any model that describes rate variation among sites in a sequence such as gamma distribution (G) and proportion of invariable sites (I). For more information on the types of substitution matrices, reader is referred to [18].

In the next section, we will focus on amino acid substitution matrices: PAM and BLOSUM.

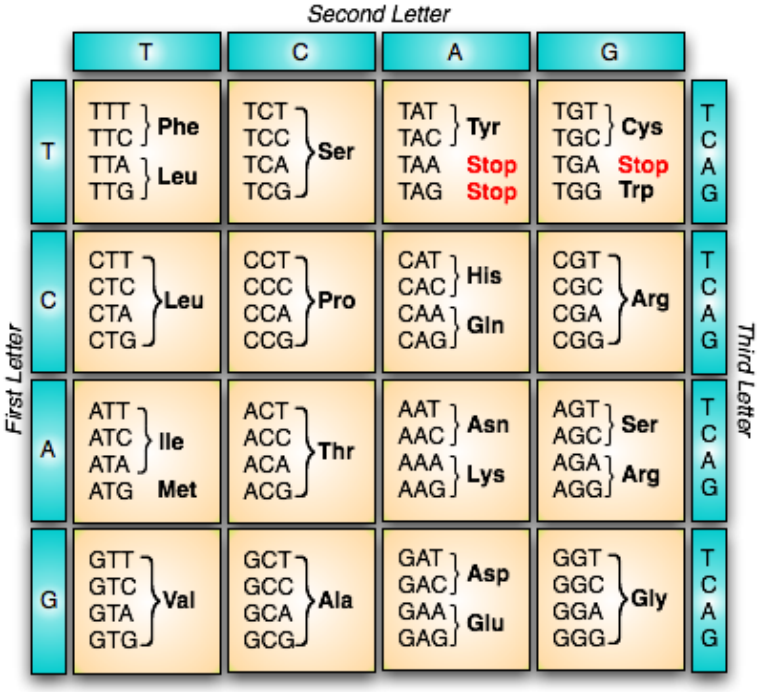


Figure 3.2: The standared genetic code for codon to amino acids translation. See [19]

3.2.2/ POINT ACCEPTED MUTATION (PAM) MATRIX

In pairwise alignment, the Point Accepted Mutation (PAM, sometimes called Percent Accepted Mutation) matrices are series of scoring matrices for amino acid[1] substitution costs, each reflecting a certain level of divergence between the acids. In 1978, The researches of Margaret Dayhoff have led to the constitution of such a matrix, by just observing the differences based on global alignment of closely related protein sequences with the identity score greater than 85% (see [20, 21]). The first version of this matrix is called PAM_1. This latter estimates of how much the rate of character substitution would be if only 1% of amino acids residue had exchanged to another amino acid type. Dayhoff starts by calculating the relative probability ratio (m_j) for each amino-acid according to the following formula:

$$m_j = \frac{number\ of\ changes\ of\ j}{number\ of\ occurrences\ of\ j} \tag{3.1}$$

The mutation probability matrix can be determined based on the following formulas:

- **For diagonal elements:**
$$M_{jj} = 1 - \lambda m_j$$
where λ is a proportionality constant, and m_j is the relative mutability ratio of j^{th} amino acid computed using Equation 3.1.

- **For non-diagonal elements:**
$$M_{ij} = \frac{\lambda m_j A_{ij}}{\sum_i A_{ij}}$$
where A_{ij} is a constant of accepted point mutation whose value can be found in [20], λ is a proportionality constant, and m_j is the relative mutability ratio of j^{th} amino acid computed using Equation 3.1

In further investigations, Dayhoff computed the Relatedness Odd matrix (R_{ij}) per amino acid as:

$$R_{ij} = \frac{M_{ij}}{f_i} \tag{3.2}$$

where, M_{ij} is the probability element of changing residue j to residue i in mutation probability matrix, and f_i represents the frequency of residue i that may occur by chance:

$$f_i = k \sum_b q_j^{(b)} N^{(b)}$$

where, the sum is taken over all alignment blocks b. $q_j^{(b)}$ is the observed frequency of amino acid j in block b, $N^{(b)}$ is the number of substitutions in a tree built for b and the coefficient k is chosen to ensure that the sum of the frequencies $f_j = 1$.

The PAM_1 matrix, shown in Figure 3.3(a), is at the basis of all the other PAM models like the log-odds matrix[2]. Matrices such as PAM_{100} and PAM_{250} are generated to reflect the

[1] Amino-acids are inferred from different nucleotide codons, see Figure 3.2

[2] This matrix is used by BLAST when scoring an alignment called *BLOSUM* (see Section 3.2.3). Indeed this latter is obtained by applying the following formula: $S_{i,j} = 10 * log(\frac{q_{i,j}}{p_i})$ on the PAM_1 matrix.

different types of amino-acid substitutions that may occurred in distantly proteins, based on the hypothesis that some repeated mutations would following the similar model conserved in PAM_1 matrix, and multiple substitutions may occur in the related site. However, other PAM matrices such as PAM_{30} and PAM_{70} are still used. An example of PAM_{250} matrix is given in Figure 3.3(b). For more information on this matrix, we recommend to read [20, 21, 22].

		Ala	Arg	Asn	Asp	Cys	Gln	Glu	Gly	His	Ile	Leu	Lys	Met	Phe	Pro	Ser	Thr	Trp	Tyr	Val
		A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
Ala	A	9867	2	9	10	3	8	17	21	2	6	4	2	6	2	22	35	32	0	2	18
Arg	R	1	9913	1	0	1	10	0	0	10	3	1	19	4	1	4	6	1	8	0	1
Asn	N	4	1	9822	36	0	4	6	6	21	3	1	13	0	1	2	20	9	1	4	1
Asp	D	6	0	42	9859	0	6	53	6	4	1	0	3	0	0	1	5	3	0	0	1
Cys	C	1	1	0	0	9973	0	0	0	1	1	0	0	0	0	1	5	1	0	3	2
Gln	Q	3	9	4	5	0	9876	27	1	23	1	3	6	4	0	6	2	2	0	0	1
Glu	E	10	0	7	56	0	35	9865	4	2	3	1	4	1	0	3	4	2	0	1	2
Gly	G	21	1	12	11	1	3	7	9935	1	0	1	2	1	1	3	21	3	0	0	5
His	H	1	8	18	3	1	20	1	0	9912	0	1	1	0	2	3	1	1	1	4	1
Ile	I	2	2	3	1	2	1	2	0	0	9872	9	2	12	7	0	1	7	0	1	33
Leu	L	3	1	3	0	0	6	1	1	4	22	9947	2	45	13	3	1	3	4	2	15
Lys	K	2	37	25	6	0	12	7	2	2	4	1	9926	20	0	3	8	11	0	1	1
Met	M	1	1	0	0	0	2	0	0	0	5	8	4	9874	1	0	1	2	0	0	4
Phe	F	1	1	1	0	0	0	0	1	2	8	6	0	4	9946	0	2	1	3	28	0
Pro	P	13	5	2	1	1	8	3	2	5	1	2	2	1	1	9926	12	4	0	0	2
Ser	S	28	11	34	7	11	4	6	16	2	2	1	7	4	3	17	9840	38	5	2	2
Thr	T	22	2	13	4	1	3	2	2	1	11	2	8	6	1	5	32	9871	0	2	9
Trp	W	0	2	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	9976	1	0
Tyr	Y	1	0	3	0	3	0	1	0	4	1	1	0	0	21	0	1	1	2	9945	1
Val	V	13	2	1	1	3	2	2	3	3	57	11	1	17	1	3	2	10	0	2	9901

(a) PAM_1 matrix, all its values are scaled by 10000.

		Ala	Arg	Asn	Asp	Cys	Gln	Glu	Gly	His	Ile	Leu	Lys	Met	Phe	Pro	Ser	Thr	Trp	Tyr	Val
		A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
Ala	A	13	6	9	9	5	8	9	12	6	8	6	7	7	4	11	11	11	2	4	9
Arg	R	3	17	4	3	2	5	3	2	6	3	2	9	4	1	4	4	3	7	2	2
Asn	N	4	4	6	7	2	5	6	4	6	3	2	5	3	2	4	5	4	2	3	3
Asp	D	5	4	8	11	1	7	10	5	6	3	2	5	3	1	4	5	5	1	2	3
Cys	C	2	1	1	1	52	1	1	2	2	2	1	1	1	1	2	3	2	1	4	2
Gln	Q	3	5	5	6	1	10	7	3	7	2	3	5	3	1	4	3	3	1	2	3
Glu	E	5	4	7	11	1	9	12	5	6	3	2	5	3	1	4	5	5	1	2	3
Gly	G	12	5	10	10	4	7	9	27	5	5	4	6	5	3	8	11	9	2	3	7
His	H	2	5	5	4	2	7	4	2	15	2	2	3	2	2	3	3	2	2	3	2
Ile	I	3	2	2	2	2	2	2	2	2	10	6	2	6	5	2	3	4	1	3	9
Leu	L	6	4	4	3	2	6	4	3	5	15	34	4	20	13	5	4	6	6	7	13
Lys	K	6	18	10	8	2	10	8	5	8	5	4	24	9	2	6	8	8	4	3	5
Met	M	1	1	1	1	0	1	1	1	1	2	3	2	6	2	1	1	1	1	1	2
Phe	F	2	1	2	1	1	1	1	1	3	5	6	1	4	32	1	2	2	4	20	3
Pro	P	7	5	5	4	3	5	4	5	5	3	3	4	3	2	20	6	5	1	2	4
Ser	S	9	6	8	7	7	6	7	9	6	5	4	7	5	3	9	10	9	4	4	6
Thr	T	8	5	6	6	4	5	5	6	4	6	4	6	5	3	6	8	11	2	33	6
Trp	W	0	2	0	0	0	0	0	0	1	0	1	0	0	1	0	1	0	55	1	0
Tyr	Y	1	1	2	1	3	1	1	1	3	2	2	1	2	15	1	2	2	3	31	2
Val	V	7	4	4	4	4	4	4	4	5	4	15	10	4	10	5	5	5	72	4	17

(b) A PAM_{250} matrix. The column summation adjusted to 100.

Figure 3.3: Examples of PAM_1 and PAM_{250} matrices presented in [20].

3.2.3/ BLOCKS SUBSTITUTION MATRIX (BLOSUM)

PAM matrices, introduced in the previous section, are obtained with the comparisons of closely related protein sequences, and so more divergent sequences cannot work with PAM. This is why, in 1992, Henikoff and Henikoff [23] introduced a new amino acid substitution matrix named *BLOcks SUbstitution Matrix* (BLOSUM). This latter is used to align protein sequences by scoring different alignments among evolutionary diverging sequences. To construct this model, a local alignment algorithm is applied on given protein sequences, then a database is scanned for highly similar block regions of protein

families (sequence alignment without gaps), in order to obtain the relevant frequencies of conserved amino acids with their substitution probabilities. After the exploration of amino-acids frequencies and their substitution probabilities, a computation of log-odds scores for each of the 210 possible substitution pairs of the 20 standard amino acids is applied. All BLOSUM matrices are based on observed alignments.

According to [23], BLOSUM matrices are obtained by using blocks of similar protein sequences as input data, then various statistical approaches are applied on the data to infer similarity scores. We recalled the following pipeline steps:

- **Procuring Frequency Table:** In this step, a local alignment algorithm is applied on the raw data of protein sequences to infer the set of conserved blocks of families, using an automatic tool named *PROTOMAT* [24], to acquire a set of scored blocks. The latter lead to construct a database of blocks. Conserved blocks are then clustered under a specific threshold to generate a set of clusters that contain a set of blocks based on identity score. In the same manner, if we want to add new sequence, then a set of matching/mismatching pairs of sequence compared with blocks should be computed. If we have a block of width w amino acids and a block depth of s sequences, it provides $\frac{ws(s-1)}{2}$ amino acid pairs. The result from this counting is a frequency table, the latter listing the number of times each of different amino acid pairs occurs among the blocks. A table is used to calculate a matrix representing the odds ratio between these observed frequencies and those expected by chance.

- **Generate a Logarithm of Odds (Lod) Matrix:** In this step, let the frequency table of total pairs of amino-acids be denoted by a function (f_{ij}). So, the function for observed probability of each given pair is:

$$q_{ij} = \frac{f_{ij}}{\sum_{i=1}^{20} \sum_{j=1}^{i} f_{ij}}. \tag{3.3}$$

  We estimate the expected probability of occurrence for each i, j pair based on i^th amino-acids by the following formula:

$$p_i = q_{ii} + \sum_{j \neq i} \frac{q_{ij}}{2}.$$

  The expected probability of occurrence e_{ij} for each i, j pair is:

$$e_{ij} = \begin{cases} p_i p_j = p_i^2 & \text{if } i = j, \\ p_i p_j + p_j p_i = 2 \times p_i p_j & \text{if } i \neq j. \end{cases} \tag{3.4}$$

  The odds ratio matrix is then calculated where each entry is q_{ij}/e_{ij}. A lod ratio is then calculated in bit units as:

$$s_{ij} = log_2(q_{ij}/e_{ij})$$

  where e_{ij} is computed from Equation 3.4, and q_{ij} is computed from Equation 3.3. Lod ratios are finally multiplied by a scaling factor of 2 and then rounded to the nearest integer value to produce the BLOSUM matrix in half-bit units, as shown in Figure 3.4.

	Ala	Arg	Asn	Asp	Cys	Gln	Glu	Gly	His	Ile	Leu	Lys	Met	Phe	Pro	Ser	Thr	Trp	Tyr	Val
Ala	4																			
Arg	-1	5																		
Asn	-2	0	6																	
Asp	-2	-2	1	6																
Cys	0	-3	-3	-3	9															
Gln	-1	1	0	0	-3	5														
Glu	-1	0	0	2	-4	2	5													
Gly	0	-2	0	-1	-3	-2	-2	6												
His	-2	0	1	-1	-3	0	0	-2	8											
Ile	-1	-3	-3	-3	-1	-3	-3	-4	-3	4										
Leu	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4									
Lys	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5								
Met	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5							
Phe	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6						
Pro	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7					
Ser	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4				
Thr	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5			
Trp	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11		
Tyr	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	
Val	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4

Figure 3.4: The standared BLOSUM62 matrix. See [23]

Remark 1: BLOSUM Number

The number attached with BLOSUM matrix represents the identity matching score in clustering step. In other words, if the conserved blocks are clustered based on an identity score of 75%, then the generated matrix is called BLOSUM75.

For more information on different types of BLOSUM matrix, see, *e.g.*, [23, 24]. Having the way to attribute a cost to a substitution in either DNA or protein sequences, we can now explain more deeply the alignment algorithms.

3.3/ Local Alignment Algorithms

In comparative biology, when we have a partial sequence of DNA and we need to provide some information about it, the first idea is to compare this sub-sequence (pattern) with a database of already identified sequences, seeking for relatively conserved subsequences [25] using local alignment algorithms (LSA). This process will find the conserved regions of this partial sequence in the database, providing thus information thanks to the reference sequence.

There are many algorithms developed for this kind of alignment. In next sub-sections, we will summarize some of the most popular ones.

3.3.1/ Basic local alignment search tool (BLAST)

In 1985, David J. Lipman and William R. Pearson [26] have developed a software package for protein-protein sequence similarity search called FASTP for proteins and FASTN for nucleotides. These software have been popularized under the name of FASTA, which is an abbreviation of “FAST-All”. This tool combines the ability to do DNA-DNA and translated protein-DNA searches.The FASTA file format is now widely used by other sequence

database search tools, such as BLAST Altschul [25], and sequence alignment programs like ClustalW [27], MUSCLE [28], T-COFFEE [29], etc.

In 1990, a more time-efficient algorithm than FASTA, called Basic local alignment search tool (BLAST), was developed by Altschul [25]. BLAST is a heuristic algorithm that gives a comparison approximation of the best local alignment between biological amino-acid sequences of protein, or nitrogen base sequences. It enables bioinformatic researchers to compare a desired query sequence with a library of sequence databases, in order to identify the target sequences that are the most similar with the desired query (given a certain threshold). Having the same sensitivity than FASTA, BLAST is more reliable as it only searches the most significant patterns in the sequence database. Note that various versions of BLAST have been developed by the National Center of Biotechnology Information NCBI.

There are various software versions of BLAST depending on the type of the queried sequence:

- *BLASTN*: Program that searches in nucleotide databases using a nucleotide query.
- *BLASTP*: Program that investigates protein databases using a protein query.
- *BLASTX*: Search in protein databases using a translated nucleotide query (*e.g.*, protein query).
- *TBLASTN*: Search in translated nucleotide databases using a protein query.
- *TBLASTX*: Search in translated nucleotide databases using a translated nucleotide query.

3.3.2/ SMITH–WATERMAN ALGORITHM

The Smith-Waterman is an algorithm based on dynamic programming developed by Smith in 1981. Its main purpose is to align locally two biological sequences in order to discover in minimal cost the optimal alignment path [30]. It is independent of any distance function (such as Euclidean, Manhattan, or Levenshtein that will be detailed in Section 3.5). The algorithm calculates the alignment that minimizes the costs provided by a certain distance function. It aims to align two sequences in a way that similar subsequences are aligned together. Local alignment is very useful when we want to align a partial portion of a sequence with a database of biological sequences. It can be applied in computer science in many applications, especially with those that need database search (such as data mining, information retrieval, pattern matching, image processing, *etc.*).

In this algorithm, a two dimensional scoring matrix T of size $(m+1)\times(n+1)$ is formed from the two provided biological sequences[3] of length n and m. One extra column and one row containing zeros are added to the matrix, for score computation. The score in each cell is computed based on the scoring function presented in Equation 3.5.

Remark 2: Zero state in SW matrix

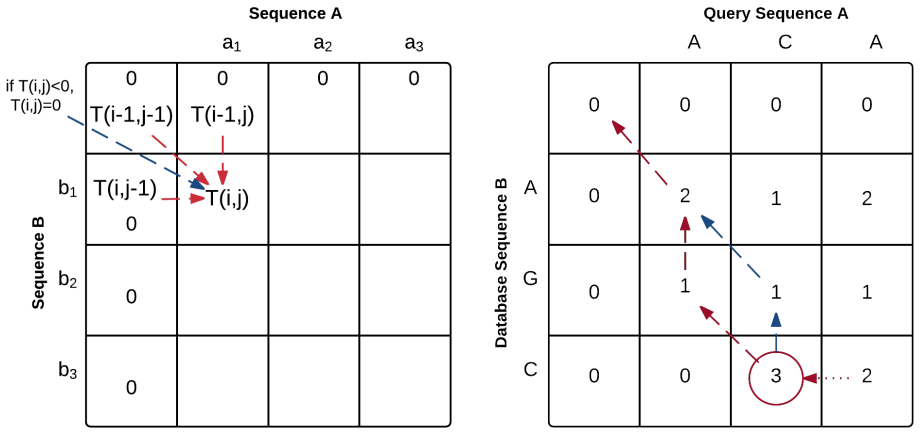
If the scoring numbers generated from the first three rules in Equation 3.5 are negative, then zero must be inserted in the cell $T(i, j)$ to ensure to have no negative value in the matrix.

[3] Biological sequences could be nucleotide, RNA, or protein sequences.

$$T(i,j) = \max \begin{cases} T(i-1,j-1) + \sigma(a_i, b_j), \\ T(i,j-1) - \text{gap penalty}, \\ T(i-1,j) - \text{gap penalty}, \\ 0. \end{cases} \tag{3.5}$$

where $T(i,j)$ is the value at line i and column j of the scoring matrix of a_i and b_j. The value $\sigma(a_i, b_j)$ is provided by a standard substitution matrix, like those detailed in Section 3.2 Note that some parameters can be optionally specified for the match, mismatch, and gap penalties in the scoring matrix.

Let us now consider that we have two nucleotide sequences A and B of different sizes, where $A = a_1a_2a_3...a_n$ and $B = b_1b_2b_3...b_m$, and let us explain how to compute the scoring matrix T. Figure 3.5(a) shows that SW uses an individual pairwise comparisons between characters to fill the scoring matrix. In this figure, to compute the value of $T(i,j)$, we need to take into consideration all the four scoring options and select the maximum one. When the matrix T is all computed, a new process starts by tracing back the matrix by selecting the position of maximum score. Then, from that position, we go up following the maximum score until reaching the first diagonal position. The selected positions in trace back process is considered as the optimal alignment path, as shown in Figure 3.5(b).



(a) Calculate the scoring values. (b) Tracing back alignments.

Figure 3.5: Example of Smith-Waterman local alignment algorithm of two given sequences (A, B), $A = a_1a_2a_3...a_n$ and $B = b_1b_2b_3...b_m$. (a) Calculate a new matching score depending heuristically on the previous around values. (b) Tracing back the alignment by starting from the maximum score in the generated matrix, then follow the maximum score on each step up.

For more details on Smith-Waterman algorithm, see [30] or [31] for an improved version.

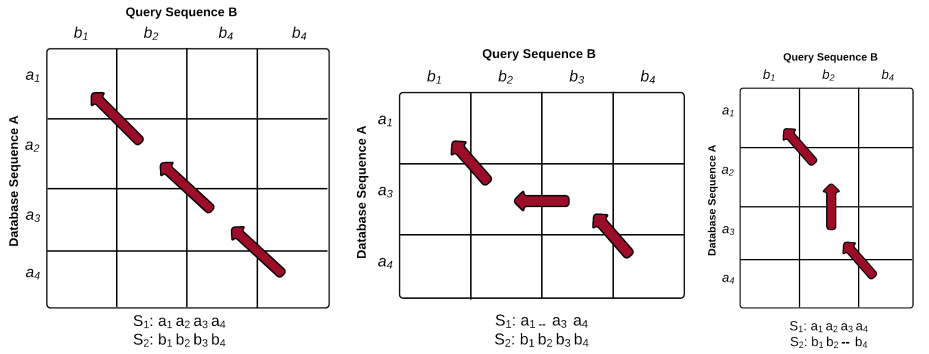
3.4/ GLOBAL SEQUENCE ALIGNMENT: THE NEEDLEMAN WUNSCH EXAMPLE

In global alignments, we compare the entire sequences by counting the amount of identical residues along the alignment. We explain in what follows how one of these algorithms works, namely the Needleman Wunsch algorithm.

The Needleman–Wunsch algorithm has been firstly developed by Saul B. Needleman and Christian D. Wunsch in 1970 [NW70]. This algorithm follows the concepts of dynamic programming: it divides a large problem (*e.g.*, the full sequence) in a series of smaller ones more tractable. Then, it solves the smaller problems in order to finally provide a solution for the larger one. The Needleman-Wunsch algorithm is still widely applied for optimal global alignment, especially when the quality of the global alignment is of high importance.

This algorithm is constituted by the following steps:

- **Setting up the matrix:** let $A = a_1a_2a_3 \dots a_n$ and $B = b_1b_2b_3 \dots b_m$ be two sequences of different sizes that we want to compare. A two-dimensional matrix T should be computed. In this matrix, the row vector represents sequence A while the column one corresponds to sequence B. A perfect correspondence or a mismatch alignment between these two sequences is represented by a diagonal line as shown in Figure 3.6(a). A gap in the first sequence leads to a horizontal line (Figure 3.6(b)), while a gap in the second sequence is drawn as a vertical line, as shown in Figure 3.6(c).



(a) Identical sequence matching. (b) Gaps in horizontal lines. (c) Gaps in vertical lines.

Figure 3.6: Example of Needleman Wunsch global alignment algorithm of two given sequences $A = a_1a_2a_3...a_n$ and $B = b_1b_2b_3...b_m$. (a) A diagonal line is when the two characters are equal, or when there is a substitution of characters. (b) Gaps in the first sequence are expressed from horizontal line. (c) Gaps in the second sequence correspond to vertical lines.

- **Scoring the Matrix:** In Needleman-Wunsch algorithm, we fill the matrix T in the same manner than in Smith-Waterman, as shown in Figure 3.7(b):

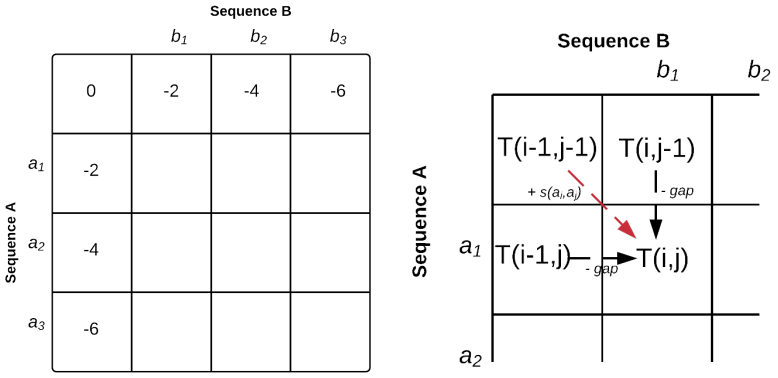
$$T(i,j) = \max \begin{cases} T(i-1, j-1) + \sigma(a_i, b_j) \\ T(i-1, j) - \text{gap penalty} \\ T(i, j-1) - \text{gap penalty} \end{cases} \tag{3.6}$$

Remark 3: Needleman-Wunsch vs Smith-Waterman

The main differences between Needleman-Wunsch and Smith-Waterman algorithms are:

- The zero condition: in SW algorithm, we insert a 0 in the cell i, j if $T_{i,j}$ is negative, which is not the case in the NW one.
- Sequences in scoring matrix are ordered in an opposite direction.

A computation example of scoring matrix is given in Figure 3.7.



(a) Matrix initialization. (b) Computing the scoring values.

Figure 3.7: Example of Needleman-Wunsch global alignment algorithm of two given sequences $A = a_1a_2a_3...a_n$ and $B = b_1b_2b_3...b_m$. (a) The initialization of the scoring matrix. (b) How to calculate the next score: (+1) for matching, (-2) for mismatching, and (-2) for gap penalty.

- **Identify the optimal path:** In Figure 3.8, the tracing back process starts from the lowest right position in the scoring matrix, following the maximum scores until reaching the upper left position. The path drawn by this matrix is considered as the optimal alignment path given for aligning the two sequences.

3.5/ EDIT DISTANCES

In computer science and information retrieval, edit distance is a way of clarifying how different two strings are. This latter can be achieved by counting the minimum number of events that are required to convert one word into another one. Edit distances are used in various application domains, for example in natural language processing where the automatic spell corrections are determined according to the closest word in a given

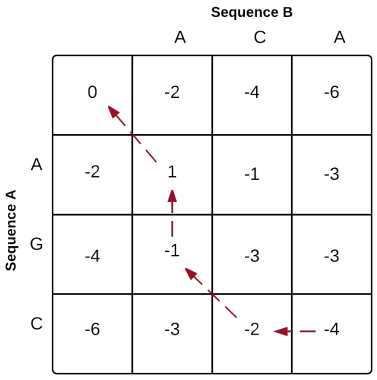


Figure 3.8: Tracing back the alignment by starting from the lowest right corner and following the maximum score on each step up.

dictionary. In bioinformatics, such distances are used to evaluate the similarity of DNA or amino acid strings.

Needleman-Wunsch alignment algorithm can be used to provide an edit distance with gaps, as the lowest right column of the scoring matrix contains the scoring cost. If the distinction between gap opening and extension is not required, and if we only need to consider insertion, deletion, and substitution of characters, then the Levenshtein edit distance can be used. This latter corresponds to usual spelling errors like in gene names, while the former is more adequate when considering usual chemical modifications of biomolecules (this fact will be used in our first contribution). Let us bring more details about the Levenshtein distance.

The string metric proposed by Vladimir Levenshtein in 1965 [32, 33], is defined formally as the minimum number of insertion, deletion, or substitution operations required to change one word into the other one. Mathematically speaking, the Levenshtein distance between A, a string of length n, and B, another string of length m, can be computed using the same dynamic programing canvas than in Needleman-Wunch, except that T matrix is filled as follows:

$$T(i,j) = \begin{cases} \max(\mathrm{i,j}) & \text{if } \min(\mathrm{i,j}) = 0, \\ \min \begin{cases} T(i-1,j)+1 \\ T(i,j-1)+1 \\ T(i-1,j-1)+1_{(a_i \neq b_j)} \end{cases} & \textit{Otherwise.} \end{cases}$$

where $1_{(a_i \neq b_j)}$ is 1 if and only if $a_i \neq b_j$ and 0 otherwise.

3.6/ MULTIPLE SEQUENCE ALIGNMENT (MSA)

Dynamic programming as described by Needleman-Wunsch for pairwise alignment is guaranteed to identify the optimal global alignment. Exact methods for multiple sequence alignment employ dynamic programming too.

The goal here is to maximize the summed alignment score of each pair of sequences. Exact methods generate optimal alignments but are not feasible in time or space for more than a few sequences. MSA are easy to generate for a group of very closely related protein (or DNA) sequences, as shown in Figure 3.9, as soon as the sequences exhibit

some divergence, the problem of multiple alignment becomes extraordinary difficult to solve. The Multiple Sequence Alignment (MSA), is a collection of three or more nucleic acid (or protein) sequences that are partially or completely aligned. Homologous residues are aligned in columns across the length of the sequences. These aligned residues are homologous in a structural sense or even in an evolutionary sense: they are presumably derived from a common ancestor.

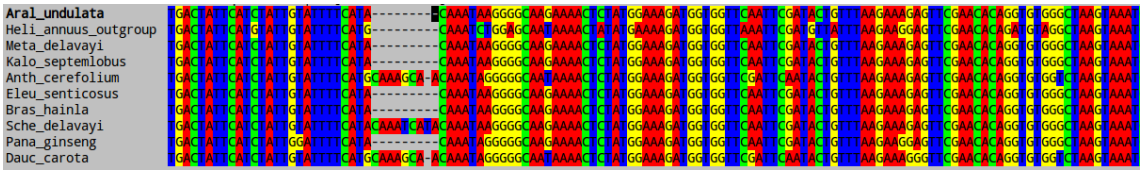


Figure 3.9: Multiple sequence alignment editing of different sequences of *Apiales* order.

The MUltiple Sequence Comparison by Log-Expectation (MUSCLE) measures the distance between given sequences by iteratively refining multiple sequence alignment by deleting the edge of the guide trees to form a bi-partition, and then extracting pair of profiles and realigning then. Several functions are applied to align pairs of columns optimally. MUSCLE uses the sum-of-pairs (PSP) profile in the scoring function:

$$PSP^{xy} = \sum_i \sum_j f_i^x f_j^y S_{ij}$$

where PSP^{xy} is a sequence-weighted sum of substitution matrix scores for each pair of latters. S_{ij} is the log expectation $S_{ij} = \log(p_{ij}/p_i p_j)$. MUSCLE applies two PAM matrices and new log-expectation score for its PSP function:

$$LE^{xy} = (1 - f_G^x)(1 - f_G^y)\log \sum_i \sum_j f_i^x f_j^y \frac{p_{ij}}{p_i p_j}$$

where the factor $(1 - f_G)$ is the occupancy of a column. For more information, see [28].

3.7/ CONCLUSION

In this chapter, we recall various algorithms of sequence alignments based on computing the edit distance. Computing the edit distance means that we considered the minimum edit operations that change one sequence into other one. In bioinformatics, sequence alignment algorithms lie in two types: local and global alignment algorithms.

In local alignment algorithms, a query sequence is aligned with a database of well-known protein or nucleotide sequences, where there are some regions with highest similarity score. Well-known algorithms for Local alignment are BLAST and Smith-Waterman. For global alignment, two sequences are aligned based on the computation of optimal alignment path. This latter is computed from a scoring function by tracing back the scoring matrix from the lower right cell following the maximum scores until reaching the upper left cell. Distance measures such as Levenshtein, Euclidean, and Manhattan distances are also detailed. Levenshtein measure is not an alignment algorithm, but it takes into account some edit operations such as insertion and deletion.

Finally, we detailed MUSCLE algorithm of multiple sequence alignment tools. We explained that this algorithm use the sum-of-pairs (PSP) profile with two PAM matrices and novel log-expectation formula.

CHAPTER 4

Concept of Phylogenetic Tree Construction

In computational and molecular biology, phylogenetic tree reconstruction is an attempt to focus on the ancestral relationship among a set of biological sequences. It involves the construction of a tree, where the nodes indicate separate evolutionary paths, and the lengths of the branches give an approximation of how distantly related the sequences represented by those branches are. This chapter gives a brief knowledge on how a phylogenetic tree can be generate from a set of DNA sequences, and how we can evaluate the predicted one. Finally, some concepts regarding phylogenetic analysis will be defined, and algorithms used for phylogenetic reconstruction will be detailed.

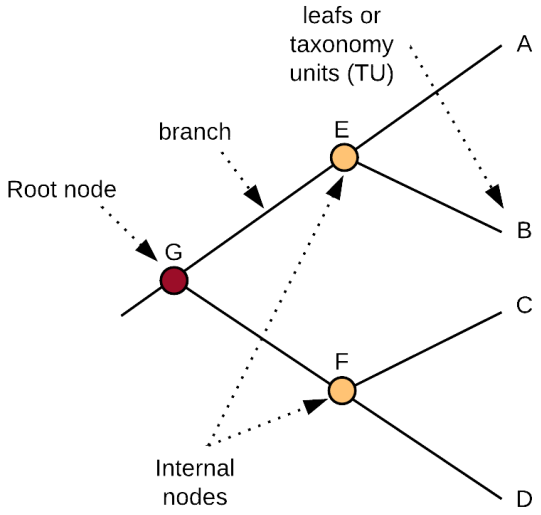
4.1/ Various Types of Phylogenetic Trees

A phylogenetic tree is a graph composed of edges (or branches) and nodes as shown in Figure 4.1(a). In this figure, edges connect exactly two nodes. A node can be either an internal (an ancestry node) or a terminal one (a leaf). Terminal nodes are sometimes called taxonomic units (TU) or simply *taxa* (plurial of *taxon*). These taxa can be organisms, coding sequences, proteins, genes, etc. Internal nodes in the tree represents the ancestor of the given TUs. A phylogenetic tree can be either rooted or not. Finally, the edge that connects one leaf with an internal node is called an *external branch*, while an edge between two internal nodes is called an *internal branch* or an *inner* one.

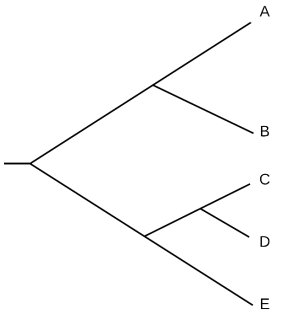
Branches define how nodes are connected in the tree, or in other words its *topology*. The latter highlights the relationship among TUs and their ancestors. Each branch has a value (or weight) which is called branch length. This value represents, for example, the number of changes (in amino-acids or nucleotide) that have possibly occurred between sequences in this branch. More precisely, depending on the existence of branch lengths, the tree can be either a *cladogram* or a *phylogram*.

In cladograms, branch lengths are not meaningful in the tree, which means that they are not related to the number of changes that have occurred between sequences. This tree is useful to align large TUs and to infer the time scale if a date of divergence is assumed precisely. An example of cladogram tree is shown in Figure 4.1(b). Conversely, in phylograms, branch lengths are scaled in the tree: they depend on the number of

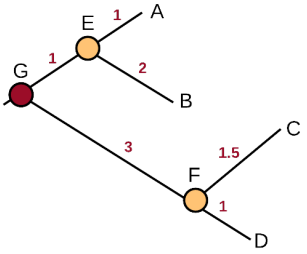
changes between sequences. An example of such trees is depicted in Figure 4.1(c).



(a) An overview of phylogenetic tree structure.



(b) An overview of cladogram tree.



(c) An overview of phylogram trees.

Figure 4.1: Types of phylogenetic trees. (a) An overview of phylogenetic tree structure. (b) Example of cladogram tree. (c) Example of phylogram trees.

As stated previously, a phylogenetic tree can be either rooted or unrooted. Let us now detail these two tree structures (for further information on phylogenetic tree construction, see, *e.g.*, [34]).

- **Unrooted Phylogenetic Trees:** This type of trees specifies the relationships among the given TUs. However, they did not provide any information to infer completely the evolution from the last common ancestors. The number of possible unrooted trees can be inferred according to the number of TUs (*c.f.* Cavalli-Sforza and Edwards [35]). It is indeed well-known that the number of trees increases rapidly with the number of TUs. More precisely, the number T_U of possible unrooted trees can be computed according to the following formula:

$$T_U = \frac{(2m-5)!}{[2^{m-3}(m-3)!]},$$

where $m \geq 3$ is the number of TUs.

Example 2: Number of unrooted trees with 6 species

Suppose that $m = 6$, then the number of generable unrooted trees is:

$$T_U = \frac{(2 \times 6 - 5)!}{[2^{6-3}(6-3)!]} = \frac{7!}{[8 \times 3!]} = \frac{5040}{48} = 105.$$

An example of unrooted tree is provided in Figure 4.2. For further information regarding unrooted trees, the reader is referred to [35, 34].

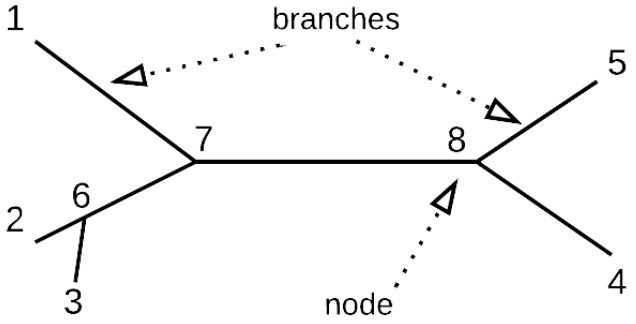


Figure 4.2: An overview of unrooted tree.

- **Rooted Phylogenetic Trees:** This type of phylogenetic trees includes a root that represents the last common ancestor of all TUs in the tree. In Figure 4.3 for instance, the internal nodes, represented by yellow circles, have an ancestor depicted in red. The main way to root a tree is to specify an *outgroup*, which is a TU known to be outside the group of TUs under consideration. This latter can be a species known to have diverged before the divergence of the considered TUs. For instance, if the leaves correspond to chloroplast genomes, then an outgroup node could be a *Cyanobacteria*, which is probably the bacteria at the origin of the chloroplasts. We have represented an outgroup (the node F) in Figure 4.3.

  The number of rooted trees can also be computed, see Cavalli-Sforza and Edwards for instance [35]. This number T_R of possible bifurcating rooted trees for m TUs is:

$$T_R = \frac{(2m-3)!}{[2^{n-2}(n-2)!]}$$

  where $m \geq 2$.

Example 3: Number of rooted trees with 6 leaves

Suppose that $m = 6$, then the number of rooted trees is equal to:

$$T_R = \frac{(2 \times 6 - 3)!}{[2^{6-2}(6-2)!]} = \frac{(12-3)!}{[(2^4) \times 4!]} = \frac{362880}{384} = 945$$

$T_R(m)$ is too the size of the searching space when inferring a rooted phylogenetic tree with m TUs.

The dotted lines in Figure 4.3 represent the delay between two bifurcations, while the red circle represents the last common ancestor of given TUs. So the time of evolution can be computed from each sub-ancestor to the last common one when either the date of divergence or the divergence rate are known. Until now, however, this problem is still a challenging task. For further information, see, *e.g.*, [36, 10, 37]

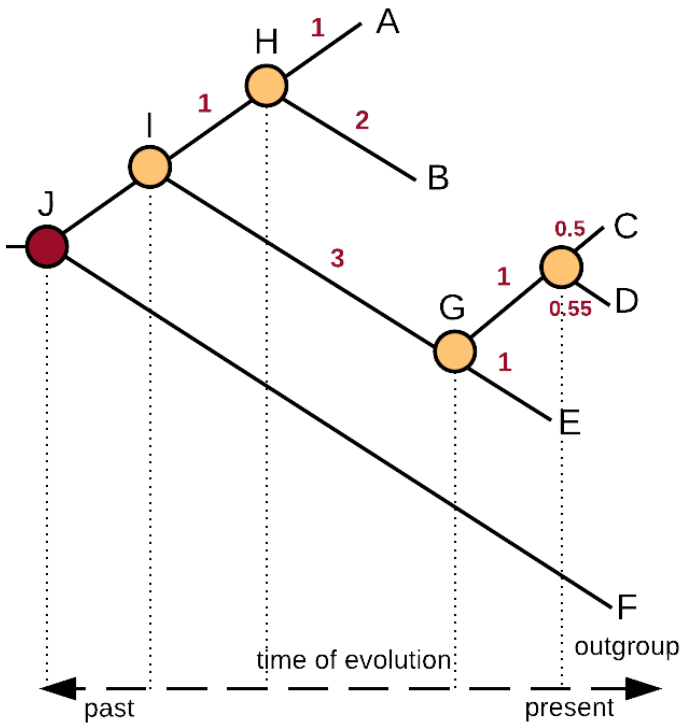


Figure 4.3: An overview on rooted phylogenetic tree.

4.2/ Methods for Phylogenetic Construction

4.2.1/ Introduction

There is a lot of methods for constructing a phylogenetic tree, which can be roughly separated in two categories: the *distance-based* and the *character-based* methods.

In distance-based methods, a multiple alignment algorithm is applied on given sequences and pairwise distances are computed on each couple of aligned sequences. This computation leads to a two-dimensional distance matrix, on which a distance-based algorithm is applied to infer the desired phylogenetic tree. These algorithms encompass *UPGMA* and *Neighbor-Joining*, this latter being detailed below.

In character-based methods, an outgroup is compared to a set of sequences. A multiple alignment algorithm is then launched to align all sequences of characters against the outgroup. The multiple character-based alignment is then exploited using *Maximum Likelihood*, *Maximum Parsimony*, or *Bayesian* methods, in order to find the best tree according to the characters. For the sake of illustrations, we will detail the maximum likelihood method in what follows.

4.2.2/ A Distance-Based Method: the Neighbor-Joining Algorithm

Neighbor-joining consists of building unrooted phylogenetic trees using distance methods [38]. It produces both topology and branch lengths by defining iteratively (based on a distance matrix) a neighbor as a pair of TUs that are connected in a single internal node X

in an unrooted bifurcating tree. Depending on the distance matrix previously computed, the method steps are:

1. Generate a full tree with all TUs in a starlike structure with no hierarchy, see Figure 4.4(a).
2. A pairwise comparison using the distance matrix is done, in order to recognize the two most related sequences (TUs). To check the selection, the sum of the branch lengths of selected TUs should be smaller than all the other ones.
3. The identified TUs are connected to an internal node X, and they are treated now as one TU, see Figure 4.4(b).
4. Select the base pair that has the smallest sum-of-branch-lengths.
5. The process continues until the topology of the tree is completed.

The neighbor-joining method produces an unrooted tree. According to Saitou and Nei (1987), The sum of the branch lengths of N TUs in the tree 4.4(a) is computed as follows: Let us define D_{ij} and L_{ab} as the distance between TUs i and j and the branch length between nodes a and b respectively. The sum of branch lengths of the tree is defined based on the following formula:

$$S = \sum_{i=1}^{N} L_{iX} = \frac{1}{N-1} \sum_{i<j} D_{ij}$$

where D_{ij} is equal to the distance between TUs i and j. Note that in Figure 4.4, we suppose that a means TU number 1, b is TU number 2, and so on. Furthermore, to compute the distance between nodes X and Y, we proceed as follows:

$$L_{XY} = \frac{1}{2(N-2)} \left[\sum_{k=3}^{N} (D_{1k} + D_{2k}) - (N-2)(L_{1X} + L_{2X}) - 2 \sum_{i=3}^{N} L_{iY} \right]$$

In this equation, the term inside the brackets is the sum of all distances including L_{XY}, and the outer term $\frac{1}{2(N-2)}$ is to exclude irrelevant branch lengths. For more details, see [38, 39].

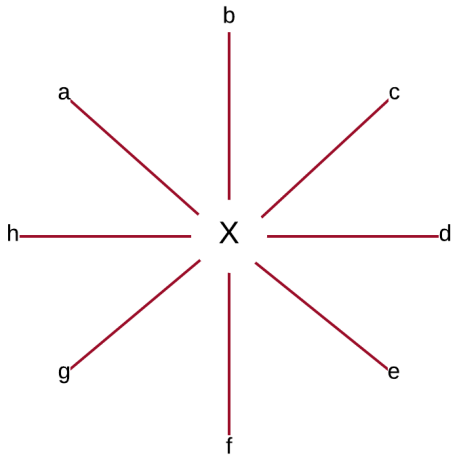
William *et al.* have presented in 2002 an improved version of neighbor-joining method called *weighted neighbor joining*, or simply *Weighbor* [39]. The Weighbor criteria for determining a pair of TUs measures the errors in the distance which can be exponentially large for higher distances. The former includes a likelihood function for computing the distances, while the latter are modeled as correlated Gaussian random variables with various means and variances, estimated under a probabilistic model for sequence evolution.

In this model, the cost function is:

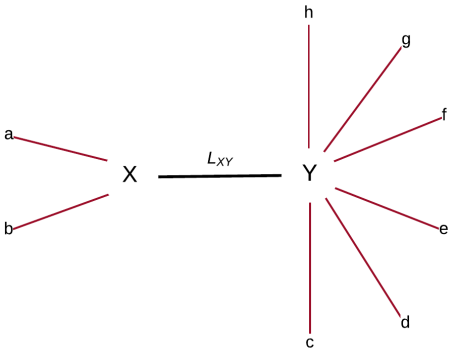
$$S(i,j) = g\mathsf{Add}(i,j) + \mathsf{Pos}(i,j)$$

where g is used to address that the tree may not be at all starlike by correcting for potential correlations among different terms in $\mathsf{Add}(i,j)$. $\mathsf{Add}(i,j)$ is defined as:

$$\mathsf{Add}(i,j) = \frac{1}{2} \sum_{k \notin \{i,j\}} \frac{[d_{ik} - d_{jk} - (\overline{d_{iP} - d_{jP}})]^2}{\sigma^2_{\text{noadd}}(d_{iP}, d_{Pk}) + \sigma^2_{\text{noadd}}(d_{jP}, d_{Pk})}$$



(a) A starlike tree.



(b) The identification of lowest sum of branch lengths TUs.

Figure 4.4: Simulation of Neighbor-Joining method. (a) All TUs are organized in starlike tree. (b) Two nodes are connected to internal node if they have lowest sum of branch lengths value.

This equation can be translated as weighted least-squares χ^2 function. σ^2_{noadd} is called a "no addition" and computed as follows:

$$\sigma^2_{\text{noadd}}(d_{iP}, d_{Pk}) = \sigma^2(d_{ik} - \sigma^2(d_{iP}) - \sigma^2(d_{Pk})),$$

where d_{iP} and d_{Pk} are simple estimation values. Finally, the "evaluating positively function" Pos(i, j) is computed as [39]:

$$\text{Pos}(i, j) = -ln\left(\frac{1}{2}erfc\left(\frac{-d_{PQ}}{\sqrt{2}\sigma_{PQ}}\right)\right).$$

4.3/ CHARACTER-BASED METHODS

We will give in what follows brief details on the three most known character-based methods, namely the *maximum likelihood*, the *maximum parsimony*, and the *Bayesian inference* method. We will then explain how to launch a RAxML maximum likelihood analysis on a given multifasta.

4.3.1/ MAXIMUM PARSIMONY

In a maximum parsimony MP method, the best tree is defined as the tree with the lowest branch lengths. More precisely, for given sequences, a multiple alignment algorithm is used to align the sequences, and to identify the informative[1] and non-informative[2] sites. The next step is to count the number of changes and assign this cost to each generated

[1] Informative sites: a column in multiple sequence alignment with no gap and at least two characters.

[2] Non-Informative sites: a column with a gap (missing character represented by a minus -) or with only one character.

phylogenetic tree. The method then computes the total length L for each tree and selects the minimum one. The L value is calculated according to the following formula:

$$L = \sum_{j=1}^{C} w_j l_j$$

where l_j is the sum of the lengths of a full tree, C is the total number of characters, and w_i is the assigned weight for each character, which is set to 1 in most cases. For further information about maximum parsimony model, see [40].

4.3.2/ Bayesian Method

For the sake of completeness, we evoke here the well-known and frequently used Bayesian methods, which estimate the phylogeny by calculating the conditional probability given the model, based on the following formula:

$$Pr[Tree|Data] = \frac{\Pr[Data|Tree] \times \Pr[Tree]}{\Pr[Data]}$$

where $Pr[Tree|Data]$ is called a posterior probability distribution[3].

Being not familiar with probability and statistics, and due to the fact that we do not have used such methods during our thesis, we will not enter more deeply in Bayesian inference. For more information about this method, see [41].

4.3.3/ Maximum Likelihood

4.3.3.1/ General presentation

The maximum likelihood ML method is commonly used for determining the topology and branch lengths that have the greatest likelihood to produce the aligned data, providing the substitution model and the tree.

Given a set of sequences on which a multiple alignment procedure has been applied, the phylogenetic tree that optimizes the above likelihood must be found. To do so, the search space (the set of all rooted trees having the good number of leaves) is visited until reaching the tree that optimizes the likelihood score (this is an optimization problem in which any optimization technique can be used). To compute this score, we must first have chosen a substitution model (see Section 3.2). Then the likelihood to have the residue (column) of the alignment, given the model and the visited tree, is computed, and all per site likelihoods are finally summed.

4.3.3.2/ Bootstrap values

Additionally to branch length values, a rooted tree can have another value attached with internal nodes, which is called a *bootstrap* value. This value is mainly used to evaluate the robustness of a given phylogenetic tree topology. This robustness evaluation can be achieved in the following way.

[3] A posterior probability is the probability that the tree is considered to be correct, if it has the maximum probability.

After obtaining the phylogenetic tree with branch lengths values, a bootstrap analysis is then involved by creating a simulated dataset of the similar size as the original one. The process starts by randomly picking columns from the multiple sequence alignment sequences; this is usually performed with replacement, where any individual column may appear multiple times or not at all. Novel trees are generated by considering a large number of bootstrap replicates (from 50 to 1000). The trees generated from bootstrap replications are then compared with the original inferred one, and the proportion of trees that present the same branch is set as bootstrap value on the associated node in the best tree. By doing so, we can observe the frequency of each clade topology in the original one.

4.4/ Stages for Phylogenetic Analysis

In what follows, we summarize the four stages required to construct a phylogenetic tree with bootstraps using maximum likelihood method.

1. **Acquiring Gene Sequences:** In this stage, corresponding sequences of each given core gene are collected from both the outgroup and the genomes under consideration (methods for acquiring gene sequences are presented in Chapter 6). The gene file, having the form depicted in Figure 4.5 in fasta format, is generated for each core gene. Such multifasta files will be the input of next stages aiming at constructing the phylogenetic tree.

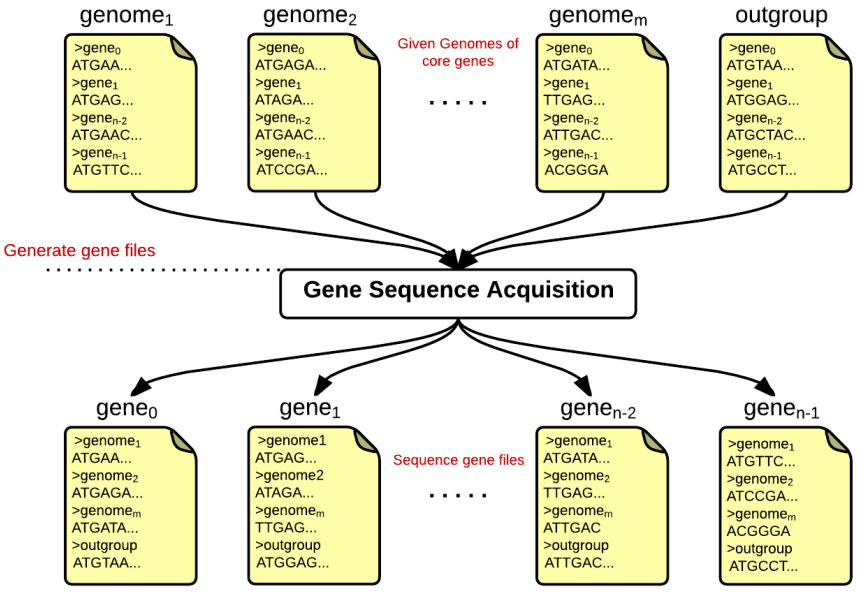


Figure 4.5: Generating individual sequence gene files. Each gene in the core genome is treated by acquiring its sequences from outgroup and given genomes.

2. **Multiple Sequence Alignment and Concatenation Stage:** A multiple sequence alignment tool, as previously explained, receives the generated fasta file. It aligns globally all including sequences of given gene as shown in Figure 4.6. Various multiple sequence alignment algorithms, like *MUSCLE* [28] or *T-COFFEE* [29], can be used for aligning separated fasta gene files.

- MUSCLE (briefly detailed in Chapter 3) on the one hand, has been used during this thesis with its default parameters. It accepts the fasta file described above, for each gene, as an input. So it produces a multiple alignment file as output. MUSCLE is a fast and semi-accurate alignment tool working with either small or large amount of sequences, while its accuracy decreases accordingly to the increasing of sequence lengths.
- On the other hand, T-COFFEE is slower than MUSCLE but more robust and accurate. It generates more accurate alignments than MUSCLE, and it works with large amount of DNA sequences. This advantage gives an extra point to use T-COFFEE instead of MUSCLE. T-COFFEE is also considered in special cases under *mcoffee* mode.Wallace *et al.* (2006) have developed a meta version of T-COFFEE called M-Coffee. This latter makes it possible to combine the output of at least eight packages (*MUSCLE, probcons [42], dialignT [43], mafft [44], clustalw [27], PCMA [45] and T-COFFEE [29]*). T-COFFEE will generate two files: .aln and .dnd. The former is the multiple alignment file of input sequences, while the latter is the guided newick format tree.

A concatenation is required to have one sequence per TU. The result of this assembly is provided as an input file for the phylogenetic tree reconstruction stage.

3. **Tree Building Stage:** This stage is concerned with the construction of phylogenetic tree. In this stage, we consider to use RAxML as a default phylogenetic tree reconstruction toolkit. If you are more interested in RAxML, we adviced you to see [46]. In this stage, the procedure of building phylogenetic tree by RAxML is divided into the following steps:

   - **Generating RAxML input file:** As shown in Figure 4.6, the generated files from sequence alignment stage are used to formulate the desired RAxML file. Based on binary pattern of given individual, gene sequences of presented genes in the binary pattern are assembled (*e.g.,* concatenating) together for each given genome. The predicted fasta file is then saved, with the amount of given genomes and the length of assembled sequences at the top of the file.
   - **Generating random tree:** In this step, a random tree of target taxa genomes is created based on the following RAxML command:
     raxmlHPC -d -f o -p 12345 -m GTRGAMMA -q Resultats/'+texte+'/modele.txt -n '+texte+'1 -o '+outgroup+' -s Resultats/'+texte+'/alignementsRAxML.fasta.
     The description of the used RAxML options are presented in Table 4.1. The GAMMA substitution model is used on the input sequence alignment file in $-s$ symbol. The assignment of models to the alignment partitions are stored in *modele.txt* file. Note that, in this step neither branch lengths, nor bootstraps values are computed yet.
   - **Invoking bootstrap analysis:** The parameters for initializing the bipartition analysis are given based on the following RAxML command: *raxmlHPC -d -f o -p 12345 -m GTRGAMMA -n '+texte+'2 -o '+outgroup+r' -b 0123 -N autoMRE -s Resultats/'+texte+'/alignementsRAxML.fasta*. In this command, *-b 0123* and *-N auto MRE* are two new options for invoking the multiple bootstrapping analysis. According to Table 4.1, *-b 0123* is the random bootstrap seed that will be considered across runs, while *-N auto MRE* specifies the number of alternative runs on given starting trees. Branch length values are estimated in this step and added into generated best tree.

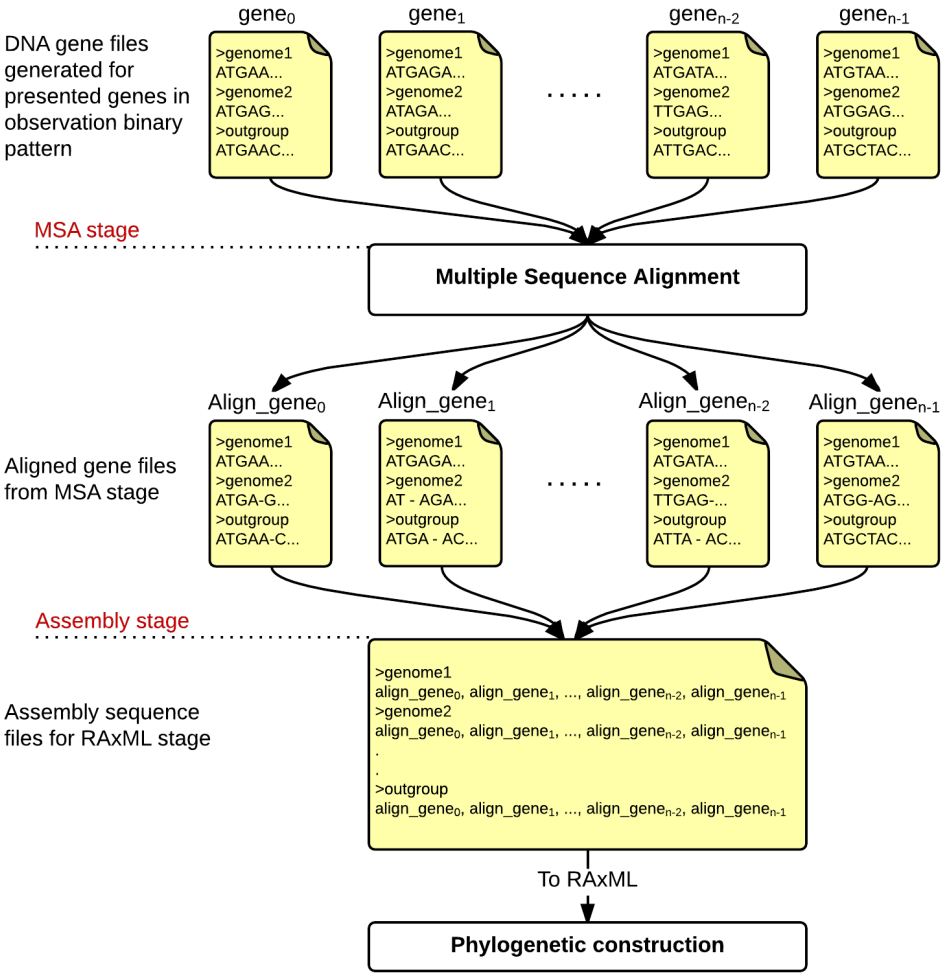


Figure 4.6: Multiple sequence alignment of genes files. In this figure, gene files with correspondent gene sequences are inputted during the multiple alignment stage. In concatenation stage, all gene sequences are concatenated based on given genomes with the outgroup. This assembly file will be used in the phylogenetic construction stage using RAxML.

- **Applying bootstrap analysis:** In this step, depending on the given trees from previous steps, the bootstrap analysis is employed by generating a bipartition file of different tree topologies based on various bootstrap replications as stated in the following command: *raxmlHPC -f o -m GTRGAMMA -q Resultats/'+texte+'/modele.txt -n '+texte+'3 -o '+outgroup+' -f b -t RAxML_bestTree.'+texte+'1 -z RAxML_bootstrap.'+texte+'2'*. The most supported tree is then generated into a newick file format.

4. **Tree Verification:** To verify the given trees, all generated *.newick* phylogenetic trees from given analysis are verified based on two factors: lowest bootstrap value and the amount of genes in given tree. A bootstrap function is applied on each tree generated from the last RAxML command in previous subsection. $-NautoMRE$ is used to specify the number of alternative runs on distinct starting tree. Using $-N$ with $-b$, this will invoke a multiple bootstrap analysis. Bootstrap information are drawn using -f b option over the best selected bootstrap tree specified by $-t$ option.

Symbol	Description
-b 0123	Specify the random bootstrap number seed that will be consistent across runs.
-d	Used to start maximum likelihood optimization from random starting tree.
-f o	Slower rapid hill climbing algorithm without the heuristic cutoff but this algorithm typically get slightly better likelihood scores.
-f b	Draw bipartition information on a best knowing Bootstrapping tree provided with -t, based on multiple bootstrap trees in a file specified by -z.
-m GTRGAMMA	Specify the substitution DNA model where the *ALPHA* values estimated.
-n	Specify the name of output file.
-N autoMRE	Specifies the number of alternative runs on distinct starting trees, with -b this will invoke a multiple bootstrap analysis.
-o	Specify the name of single outgroup genome.
-p 12345	Specify a random number seed for the parsimony inferences.
-q	Specify the file name which contains the assignment of models to alignment partitions for multiple models of substitution.
-s	Specify the name of the alignment data file in PHYLIP or FASTA format.
-z	Specify the file name of a file containing multiple trees e.g. from a bootstrap that shall be used to draw bipartition values onto a tree provided with $-t$.

Table 4.1: Optional parameters of RAxML commands.

The number of genes (or gene rate) in the other hand indicates how many gene are conserved to generate the target phylogenetic tree. The largely presented genes are the highly stable tree.

4.5/ CONCLUSION

In this chapter, we gave a small background on phylogenetic tree reconstruction from biological sequences. The types of the phylogenetic tree presented as rooted and unrooted trees, and we showed how unrooted tree can represent the natural relations among applied Taxonomy units (TUs). The unrooted tree based on some related works can be inferred based on the number of TUs. Indeed, this latter did not provide useful information. On the contrary, rooted trees provide more useful information on how the given tree are growth over time. We also showed how many rooted trees can be infer based on the number of TUs.

Various models can be applied for constructing the phylogenetic tree of desired sequences. Two branches have been realized in this domain: distance-based and character-based methods. In distance-based methods, a phylogenetic tree can be constructed by calculating the distances between desired sequences for a distance matrix. Two models are available in this kind of models: UPGMA and Neighbor-joining methods. The two algorithms are closed in their techniques so that we focused on neighbor-joining

algorithm as the fast, reliable, and most know algorithm for constructing phylogenetic tree based on distance matrix.

In character-based methods, three methods are available for the construction of tree depending on providing a reference sequence (outgroup): maximum likelihood, maximum parsimony, and Bayesian methods. Each of these algorithms has its own technique. In this manuscript, we only focus on the maximum likelihood as the main method for constructing phylogenetic trees. A bootstrap analysis is applied with each generated tree to compute the fitness value. We conclude that producing phylogenetic trees supported by bootstrap values can give to us a confident tree, so that, different TUs laying in the same clade are biologically related.

II

CONTRIBUTIONS

CHAPTER 5

General Introduction

We now enter in the main contribution part of this manuscript.

The first chapter of this part, Chapter 6, investigates the problem to find the core and pan genomes of a given set of chloroplastic sequences. Various approaches are evaluated, based either on NCBI database or on DOGMA annotation tool.

We describe in chapter 7 an optimization pipeline using the genetic algorithm that can efficiently optimize the searching space for well supported phylogenetic inference. In other words, we deal with discovering homoplasy in phylogenetic reconstruction. This is considered as a difficult computational problem, because the number of situations to investigate dramatically increases with the number of core genes and taxa.

More precisely, the objective is to obtain a well-supported phylogenetic tree by using the largest possible subset of core genes obtained previously. Indeed, if a well-supported tree cannot be reached by taking all core genes, the first thing to investigate is to test whether one or two particular genes are not responsible for this problem (by blurring the phylogenetic signals). In order to find such a supported tree that uses the largest possible subset of core genes, a genetic algorithm coupled with a lasso test is applied to identify (and remove) blurring genes

This work is then extended in chapter 8 by integrating a discrete particle swarm optimization method to provide the largest subset of sequences in order to obtain the most supported species tree.

Our proposed pipeline has been applied to various families of plant species. More than 65% of phylogenetic trees produced by this pipeline have presented bootstrap values larger than 95.

Finally, the last chapter 9 of this part proposes a first ancestral reconstruction algorithm. It receives a well supported phylogenetic tree based on a large set of core genes, and it puts gene contents at its leaves. Then all internal nodes until the root receive their (ancestral) gene contents.

CHAPTER 6

Core-Genes Prediction Approaches

Due to the recent evolution of sequencing techniques, the number of available genomes are rising steadily, leading to the possibility to make a large-scale genomic comparison between sets of close species. An interesting question to answer is: what is the common functionality genes of a collection of species. Or, conversely, to determine what is specific to a given species when compared to other ones belonging to the same genus or family. Investigating such problem means to find both core and pan genomes of a collection of species, *i.e.*, genes in common to all the species versus the set of all genes in all species under consideration. This chapter presents some general and heuristic methods for inferring such core and pan genomes, it summarizes three articles published in international conferences [47, 48, 49].

6.1/ INTRODUCTION

In bioinformatics, identifying core genes may be of importance, for instance to understand the shared functionality and specificity of a given set of species, or to construct their phylogeny using curated sequences. Therefore, in this chapter we present methods to determine both core and pan genomes of a large set of DNA sequences. However, obtaining trustworthy core and pan genomes is not an easy task, leading to a significant amount of computation, and requiring a rigorous methodology. This chapter is the basis of our work, which is progressively presented in the next chapters.

More precisely, we provide three distinct methods in order to obtain the set of desire core genome. A general overview of the entire proposed pipeline for core and pan genomes production and exploitation is presented in Figure 6.1, which consists of three principle stages: *Genomes Annotation*, *Core Extraction*, and *Features Visualization*.

As a starting point, the pipeline uses a DNA sequence database like NCBI's GenBank [50], the European *EMBL* database [51], or the Japanese *DDBJ* one [52] for acquiring target genomes. It is possible to obtain annotated genomes (DNA coding sequences with gene names and locations) by interacting with these databases, either by directly downloading annotated genomes delivered by these web sites, or by launching an annotation tool on complete downloaded genomes. Obviously, this annotation stage must be of

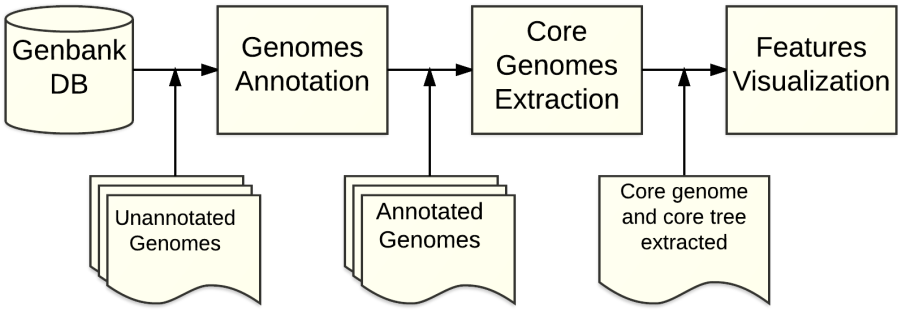


Figure 6.1: A general overview of the annotation-based approach

quality if we want to obtain acceptable core and pan genomes. Various cost-effective annotation tools [53] that produce genomic annotations have been designed recently, some reputed ones being: DOGMA [54], CpBase [55], CpGAVAS [56], and CEGMA [57]. Such tools usually use one out of the three following methods for finding gene locations in large DNA sequences: *alignment-based*, *composition-based*, or a combination of both [57]. An alignment-based method is used when trying to predict a protein coding sequence by aligning a genomic DNA sequence with a cDNA sequence coding an already known homologous protein [57]. This approach is applied, for instance, in GeneWise [58]. The alternative method, the composition-based one (also known as *ab initio*) is based on probabilistic models of gene structure [59].

These tools will be used in our pipeline in order do find, in a second stage, the genes that are commonly shared through the considered set of annotated genomes. Then, a final step is to take advantage of the information produced during this core and pan genome search. The feature visualization stage encompasses phylogenetic tree construction using core genes, genes content evolution illustrated by core trees, functionality investigations, and so on.

At the end of this chapter, a running example is proposed to demonstrate the relevance of the suggested approaches.

6.2/ Core genome extraction Approaches

We will detail three general approaches, published in various international conferences [47, 48, 49], for eliciting core genome, which serve as the second stage of the offered pipeline. The first approach uses similarities computed on predicted coding sequences, while the second one uses all the information provided during the annotation stage. The third method takes the advantages from the first two methods by considering gene names and sequences in order to find the target core genome. Indeed, such annotations can be used in various manners (based on gene names, gene sequences, and protein sequences) to extract the core and pan genomes.

6.2.1/ Similarity-based Approach

The first method, described below, considers a distance-based similarity measure on gene' coding sequences. Such an approach requires annotated genomes, like the ones

provided by the NCBI website.

6.2.1.1/ THEORETICAL PRESENTATION

We start with the following preliminary definition:

Definition 4: Similarity Matrix

Let $A = \{\mathcal{A}, \mathcal{T}, C, \mathcal{G}\}$ be the nucleotides alphabet, and A^* be the set of finite words on A (*i.e.*, of DNA sequences). Let $d : N = A^* \times A^* \rightarrow [0, 1]$ be a function called similarity measure on A^*. Consider a given value $T' \in [0, 1]$ called a threshold. For all $x, y \in A^*$, we will say that $x \sim_{d,T'} y$ if $d(x, y) \leqslant T'$.

Let be given a *similarity* threshold T' and a *similarity measure* d. The method begins by building an undirected graph between all the DNA sequences g of the set of genomes as follows: there is an edge between g_i and g_j if $g_i \sim_{d,T'} g_j$ is established. In other words, vertices are DNA sequences and two sequences are connected with an edge if their similarity is larger than a predefined threshold.

Remark 4: Graph connection limitation

This graph is not connected for sufficiently large threshold values.

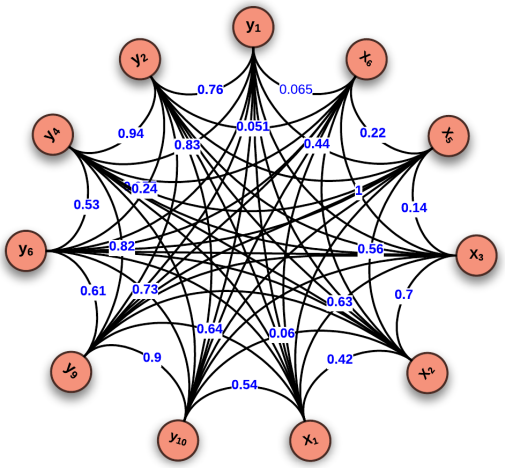
An example of such a graph denoted as the "similarity" graph, is shown in Figure 6.2(a). We thus say that two coding sequences g_i, g_j are equivalent with respect to the relation $\mathcal{R}$ if both g_i and g_j belong to the same Connected Component (CC) of this similarity graph, *i.e.*, if there is a path between g_i and g_j in the graph. To say this in another way, if there is a finite sequence $s_1, ..., s_k$ of vertices (DNA sequences) such that g_i is similar to s_1, which is similar to s_2, etc., and s_k is similar to g_j (as shown in Figure 6.2(b)).

It is not hard to see that this relation is an equivalence relation whereas $\sim$ is not. Any class for this relation is called a "gene" in this chapter, where its representatives (DNA sequences) are the "alleles" of this gene, such abuse of language being proposed to set our ideas down. Thus this first method produces for each genome G, which is a set $\left\{g_1^G, ..., g_{m_G}^G\right\}$ of m_G DNA coding sequences, the projection of each sequence according to π, where π maps each sequence into its gene (class) according to $\mathcal{R}$. In other words, a genome G is mapped into $\left\{\pi(g_1^G), ..., \pi(g_{m_G}^G)\right\}$. Note that a projected genome has no duplicated gene since it is a set.

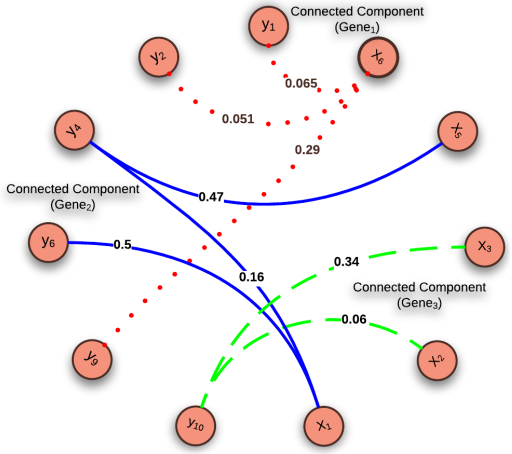
Consequently, the core genome (resp., the pan-genome) of two genomes G_1 and G_2 is defined as the intersection (resp., as the union) of their projected genomes. We finally consider the intersection of all the projected genomes, which is the set of all the genes $\dot{x}$ such that each genome has at least one allele in $\dot{x}$. This set will constitute the core genome of the whole species under consideration. The pan-genome is computed similarly as the union of all the projected genomes.

Remark 5: Major issue of gene prediction method

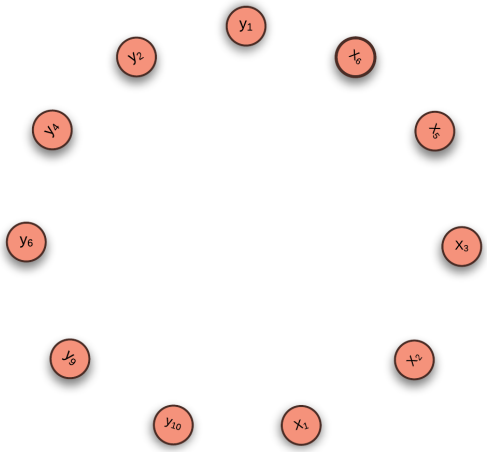
This first method requires the computation of all similarities among all allele sequences in all species under consideration. According to the number of organisms and even with a focus on a specific family or function, this is a computationally heavy operation.



(a) Complete (fully connected) graph for given coding sequences for $T = 1$.



(b) Amounts of generated connected components when $T = 0.5$.



(c) No connected components when $T = 0$.

Figure 6.2: Example of similarity-based approach of two given genomes (G_1, G_2), G_1 has five coding sequences $(\{x_1, x_2, x_3, x_5, x_6\})$ and G_2 has six coding sequences $(\{y_1, y_2, y_4, y_6, y_9, y_{10}\})$. (a) The similarity graph. On each connected edge, there is a similarity score between g_i and g_j. (b) Connected components obtained when $T = 0.5$. (c) No connected components when $T = 0$.

6.2.1.2/ A FIRST CASE STUDY

For illustration purposes, we have considered in this chapter 99 genomes of chloroplasts downloaded from GenBank database [50] as shown in Table 6.1. These genomes lie in the eleven type of chloroplast families as shown in Figure 6.3. Two kinds of annotations has been used, namely the ones provided by NCBI on the one hand, and the ones by DOGMA on the other hand. DOGMA[1], which stands for *Dual Organellar GenoMe Annotator*, has already been evoked in this chapter. The choice of DOGMA is natural, as this

[1]DOGMA has been developed in 2004 at the University of Texas for annotating plant chloroplast and animal mitochondrial genomes. This tool translates a genome in all six reading frames and then queries its amino acid sequence database using BLAST (blastx [25]) with various ad hoc parameters.

F.	#	Acc. No	Scientific Name
Brown Algae	11	NC_001713.1	*Odontella sinensis*
		NC_008588.1	*Phaeodactylum tricornutum*
		NC_010772.1	*Heterosigma akashiwo*
		NC_011600.1	*Vaucheria litorea*
		NC_012903.1	*Aureoumbra lagunensis*
		NC_014808.1	*Thalassiosira oceanica*
		NC_015403.1	*Fistulifera sp*
		NC_016731.1	*Synedra acus*
		NC_016735.1	*Fucus vesiculosus*
		NC_018523.1	*Saccharina japonica*
		NC_020014.1	*Nannochloropsis gadtina*
F1	3	NC_000925.1	*Porphyra purpurea*
		NC_001840.1	*Cyanidium caldarium*
		NC_006137.1	*Gracilaria tenuistipitata*
Green Algae	17	NC_000927.1	*Nephroselmis olivacea*
		NC_002186.1	*Mesotigma viride*
		NC_005353.1	*Chlamydomonas reinhardtii*
		NC_008097.1	*Chara vulgaris*
		NC_008099.1	*Oltmannsiellopsis viridis*
		NC_008114.1	*Pseudoclonium akinetum*
		NC_008289.1	*Ostreococcus tauri*
		NC_008372.1	*Stigeoclonium helveticum*
		NC_008822.1	*Chlorokybus atmophyticus*
		NC_011031.1	*Oedogonium cardiacum*
		NC_012097.1	*Pycnococcus provaseolii*
		NC_012099.1	*Pyramimonas parkeae*
		NC_012568.1	*Micromonas pusilla*
		NC_014346.1	*Floydiella terrestris*
		NC_015645.1	*Schizomeris leibleinii*
		NC_016732.1	*Dunaliella salina*
		NC_016733.1	*Pedinomonas minor*
F2	3	NC_001319.1	*Marchantia polymorpha*
		NC_004543.1	*Anthoceros formosae*
		NC_005087.1	*Physcomitrella patens*
F3	2	NC_014267.1	*Kryptoperidinium foliaceum*
		NC_014287.1	*Durinskia baltica*
F4	2	NC_001603.2	*Euglena gracilis*
		NC_020018.1	*Monomorphina aenigmatica*
Ferns	5	NC_003386.1	*Psilotum nudum*
		NC_008829.1	*Angiopteris evecta*
		NC_014348.1	*Pteridium aquilinum*
		NC_014699.1	*Equisetum arvense*
		NC_017006.1	*Mankyua chejuensis*
F5	1	NC_007288.1	*Emiliana huxleyi*
F6	2	NC_014675.1	*Isoetes flaccida*
		NC_006861.1	*Huperzia lucidula*

F.	#	Acc. No	Scientific Name
Angiosperms	45	NC_007898.3	*Solanum lyopersicum*
		NC_001568.1	*Epifagus virginiana*
		NC_001666.2	*Zea Mays*
		NC_005086.1	*Amborella trichopoda*
		NC_006050.1	*Nymphaea alba*
		NC_006290.1	*Panax ginseng*
		NC_007578.1	*Lactuca sativa*
		NC_007957.1	*Vitis vinifera*
		NC_007977.1	*Helianthus annuus*
		NC_008325.1	*Daucus carota*
		NC_008336.1	*Nandina domestica*
		NC_008359.1	*Morus indica*
		NC_008407.1	*Jasminum nudiflorum*
		NC_008456.1	*Drimys granadensis*
		NC_008457.1	*Piper cenocladum*
		NC_009601.1	*Dioscorea elephantipes*
		NC_009765.1	*Cuscuta gronovii*
		NC_009808.1	*Ipomea purpurea*
		NC_010361.1	*Oenothera biennis*
		NC_010433.1	*Manihot esculenta*
		NC_010442.1	*Trachelium caeruleum*
		NC_013707.2	*Olea europea*
		NC_013823.1	*Typha latifolia*
		NC_014570.1	*Eucalyptus*
		NC_014674.1	*Castanea mollissima*
		NC_014676.2	*Theobroma cacao*
		NC_015830.1	*Bambusa emeiensis*
		NC_015899.1	*Wolffia australiana*
		NC_016433.2	*Sesamum indicum*
		NC_016468.1	*Boea hygrometrica*
		NC_016670.1	*Gossypium darwinii*
		NC_016727.1	*Silene vulgaris*
		NC_016734.1	*Brassica napus*
		NC_016736.1	*Ricinus communis*
		NC_016753.1	*Colocasia esculenta*
		NC_017609.1	*Phalaenopsis equestris*
		NC_018357.1	*Magnolia denudata*
		NC_019601.1	*Fragaria chiloensis*
		NC_008796.1	*Ranunculus macranthus*
		NC_013991.2	*Phoenix dactylifera*
		NC_016068.1	*Nicotiana undulata*
Gymnosperms	7	NC_009618.1	*Cycas taitungensis*
		NC_011942.1	*Gnetum parvifolium*
		NC_016058.1	*Larix decidua*
		NC_016063.1	*Cephalotaxus wilsoniana*
		NC_016065.1	*Taiwania cryptomerioides*
		NC_016069.1	*Picea morrisonicola*
		NC_016986.1	*Gingko biloba*

where lineages F1, F2, F3, F4, F5, and F6 are *Red Algae*, *Bryophytes*, *Dinoflagellates*, *Euglena*, *Haptophytes*, and *Lycophytes* respectively.

Table 6.1: List of chloroplast genomes of photosynthetic Eucaryotes lineages from NCBI

annotation tool is reputed and specific to chloroplasts.

Each genome is thus transformed in a list of coding sequences, which depends on the chosen annotation tool. We have firstly evaluated the similarity score between each couple of sequences by using a Needleman-Wunch global alignment. The number of genes in the core genome and the pan-genome has then been computed according to the graph method detailed in the previous section. Obtained results with various threshold values are represented in Table 6.2.

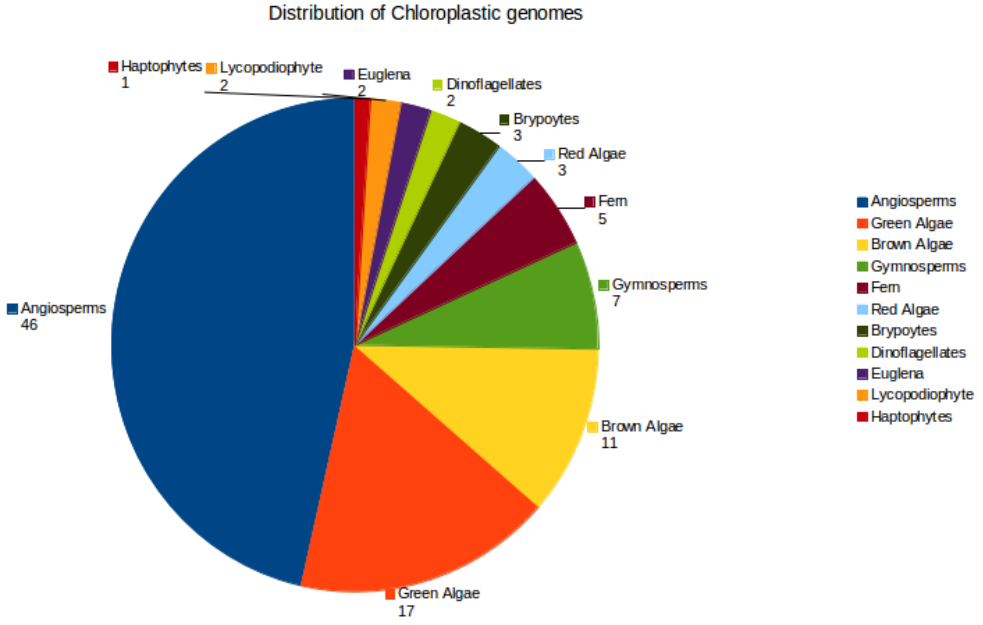
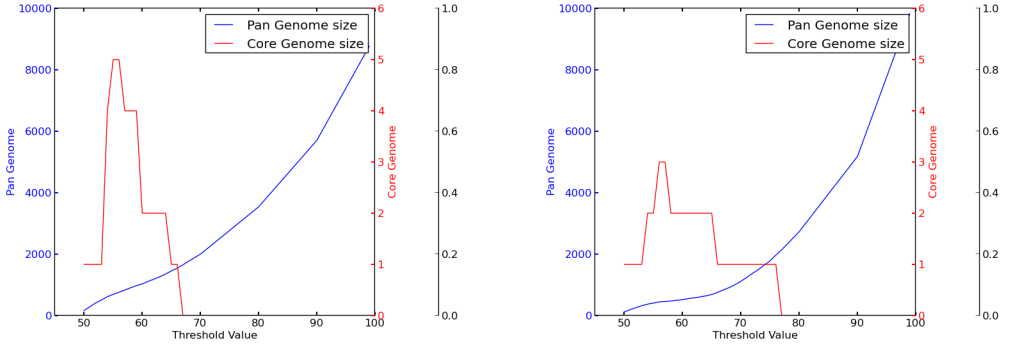


Figure 6.3: Distribution of 99 chloroplast genomes.



(a) Results based on NCBI annotation.

(b) Results based on DOGMA annotation.

Figure 6.4: Results obtained from genomes annotated based on (a) NCBI and (b) DOGMA

Computations regarding this first method, using both data and measure described previously, were done on the supercomputing facilities of the Mésocentre of calculations of Franche-Comté. Obtained results are presented in Figures 6.4(a) and 6.4(b) with respect to various threshold values on Needleman-Wunsch similarity scores.

Remark 6: Threshold status

When the threshold is large, we obtain more connected components, but with smaller sizes: a large number of genes, with a few numbers of alleles for each of them. In other words, when the threshold is high, the pan-genome is large too whereas the core-genome becomes either small or empty.

Similarity-Based Approach						
	NCBI		DOGMA			
Threshold(%)	core	pan	core	pan	genes NCBI	genes DOGMA
50	1	163	1	118	1	1
55	**5**	**692**	2	409	3, 4, 19, 61, 69	1, 45
60	2	1032	2	519	4, 88	1, 57
65	1	1454	2	685	4	1, 66
70	0	2000	1	1116		10
75	0	2667	1	1781		19
80	0	3541	0	2730		
85	0	4620	0	3945		
90	0	5703	0	5181		
95	0	7307	0	7302		
100	0	8911	0	10132		

Table 6.2: Size of core and pan genomes w.r.t. the similarity threshold

No matter the chosen annotation tool, this first approach suffers from producing too small core genomes, for any chosen similarity threshold, compared to what is usually expected by biologists. For NCBI, it is certainly due to a wrong determination of start and stop codons in some annotated genomes. Indeed, due to the large variety of annotation tools used during genomes submission on the NCBI server, some of them being old or deficient, some genes may be truncated. And such truncated genes will not produce a significant similarity score with their orthologuous genes found in other genomes. The case of DOGMA, for its part, is more difficult to explain as. According to our experiments and the state of the art, this gene prediction tool produces normally good results in average.

The best explanation of such an under-performance is that a few genomes are very specific and far from the remaining ones, in terms of gene contents, which leads to a small number of genes in the global core genome. However, this first approach cannot help us to determine which genomes we must remove from our data. To do so, we need to introduce a second approach based on gene names: from the problematic gene names, we will be able to trace back to the problematic genomes.

6.2.2/ Annotation-based Approach

6.2.2.1/ Using genes names provided by annotation tools

Instead of using the sequences predicted by annotation tools, we can try to use the names associated with these sequences, when available. The basic idea is thus to annotate all the sequences using a given software, and to consider as a core gene each sequence whose name can be found in all the genomes.

It is true that the NCBI annotations are of varying qualities, and sometimes such annotations are totally erroneous. However this database contains human-curated annotations of very good quality, and we wonder in this chapter if it is possible to detect and only use such well-annotated and curated genes. To summarize the approach detailed in this section: automatic DOGMA annotations are useful due to the automatic mechanism used for identifying genes and associating names without mistakes, while NCBI contains very good human-based annotations (together with errors). Our idea here is then to try to take

the best of both automatic and humanly curated approaches. Let us finally remark that DOGMA also predicts locations of *ribosomal RNA (rRNA)*, while they are not in the gene features file downloaded from NCBI.

We now investigate core and pan genome discovery using each of the two tools separately, which will constitute the second approach detailed in this chapter. From now on we will only consider annotated genomes: either "gene features" downloaded from the NCBI or the result of DOGMA.

6.2.2.2/ NAMES PROCESSING

As DOGMA is a deterministic annotation tool, when a given gene is detected twice in two genomes, the same name will be attached to the two coding sequences: DOGMA spells exactly in the same manner the two gene names. So each genome is replaced by a list of gene names, and finding the common core genes between two genomes simply consists in intersecting the two lists of genes. The sole problem we have detected using DOGMA on our chloroplasts is the case of the RPS12 gene: some genomes contain RPS12_3end or RPS12_5end in DOGMA result. We have manually considered that all these representatives belong to the same gene, namely to RPS12.

Dealing with NCBI names is more complicated, as various automatic annotation tools have been used together with human annotations, and because there is no spelling rule for gene names. For instance, NAD6 mitochondrial gene is sometimes written as ND6 while we can find RPOC1, RPOC1A, and RPOC1B in our chloroplasts. So if we simply consider NCBI data without treatment, intersecting two genomes provided as a list of gene names often leads to duplication of misspelled genes. Automatic names homogenization is thus required on NCBI annotations, the question being where to draw the line on correcting errors in the spelling of genes? In this second approach, we propose to automate only obvious modifications like putting all names in capital letters and removing useless symbols as "_", "(", and ")". Remark that such simple renaming process cannot tackle with the situations of NAD6 or RPOC1 evoked above. To go further in automatic corrections requires using edit distances like Levenshtein. However, such use will raise false positives (different genes with close names will be homogenized). The use of edit distances on gene names, together with a DNA sequence validation stage, will be investigated in a second methodology chapter.

In this section, we now consider that each genome is mapped to a list of gene names, where names have been homogenized in the NCBI case.

6.2.2.3/ CORE GENES EXTRACTION

To extract core genes, we iteratively collect the maximum number of common genes among genomes. Therefore, during this stage, an *Intersection Core Matrix* (ICM) is built. ICM is a two-dimensional symmetric matrix where each row and each column corresponds to one genome. Hence, an element of the matrix stores the *Intersection Score* (IS): the cardinality of the core genes obtained by intersecting the two genomes. Mathematically speaking, if we have n genomes in local database, the ICM is a $n \times n$ matrix whose each element $score_{ij}$ satisfies:

$$score_{ij} = |g_i \cap g_j| \tag{6.1}$$

where $1 \leq i \leq n$, $1 \leq j \leq n$, and g_i, g_j are genomes. The generation of a new core genome obviously depends on the value of the intersection scores $score_{ij}$. More precisely, the idea is to consider a pair of genomes such that their score is the largest element in the ICM. These two genomes are then removed from the matrix and the resulting new core genome is added for the next iteration. The ICM is then updated to take into account the new core genome: new IS values are computed for it. This process is repeated until no new core genome can be obtained. Figure 6.5 demonstrates the construction of ICM matrix.

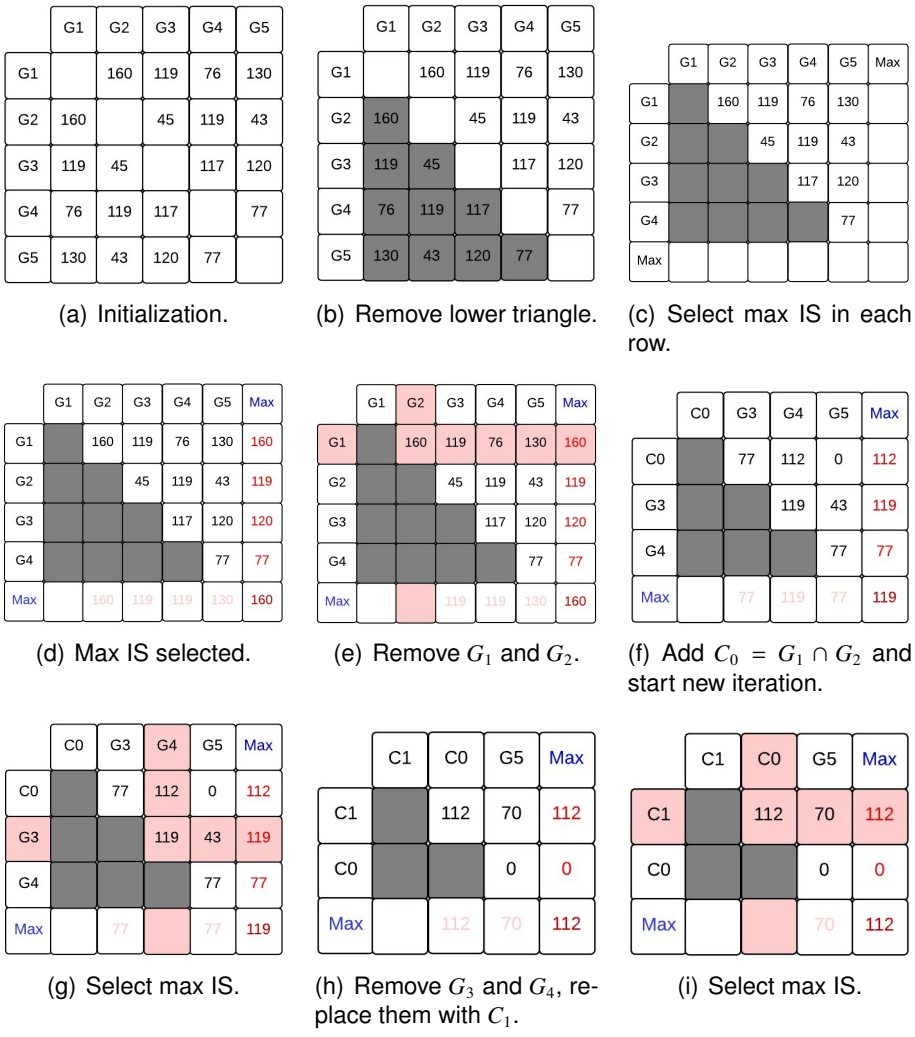


Figure 6.5: Evolution of the Intersection core matrix.

We can observe that the ICM is relatively large due to the amount of species. As a consequence, the computation of the intersection scores is both time and memory consuming. However, since ICM is obviously a symmetric matrix we can reduce the computation overhead by considering only its upper triangular part. The time complexity for this process is: $O(\frac{n.(n-1)}{2})$, where n is the number of genes. Algorithm 1 illustrates the construction of the ICM matrix and the extraction of the core genomes, where *GenomesDB* represents the database storing all genome data. At each iteration, this algorithm computes the maximum core genome from a set of genomes.

Algorithm 1: Retrieve Maximum Intersection Score

Require: $L \leftarrow$ GenomesDB
Ensure: Scores $\leftarrow$ Max Core genome
  for $i \leftarrow 1 : len(L) - 1$ **do**
    $G_i \leftarrow$ genome L_i
    $score \leftarrow 0$
    $core_1 \leftarrow$ gene set of G_i
    for $j \leftarrow i + 1 : len(L)$ **do**
      $G_j \leftarrow$ genome L_j
      $core_2 \leftarrow$ gene set of G_j
      $core \leftarrow core_1 \cap core_2$
      if $|core| > score$ **then**
        $score \leftarrow |core|$
        $G_{best} \leftarrow G_j$
      end if
    end for
    Scores$[score] \leftarrow (G_i, G_{best})$
  end for
  return max(Scores)

6.2.3/ Quality Test Approach

Let us now present the last approach. We start by the following definition:

Definition 5: Quality genome

Let G_i be a set of n gene names in genome i annotated by NCBI, and G'_i be a set of m gene names in genome i annotated by DOGMA.
A core gene g from the set of core genes $c_i = G_i \cap G'_i$, is called a quality gene if and only if $d(S_g^{G_i}, S_g^{G'_i}) \geqslant 90\%$, where d is the similarity score obtained after a global alignment algorithm between sequence $S_g^{G_i}$ of gene g in G_i and sequence $S_g^{G'_i}$ of gene g in G'_i.
The quality genome G_i^{quality} contains the sequences of generated quality genes from c_i.

In Definition 5, we propose to take the best from NCBI and DOGMA annotations. This latter is done by integrate a similarity distance on gene names in the pipeline (see Figure 6.6). Each similarity is computed between a name from DOGMA and a name from NCBI, as shown in the *Gene* column in Figure 6.7.

The proposed distance is the Levenshtein one, which is close to the Needleman-Wunsch, except that gap opening and extension penalties are equal. The same name is then set to sequences whose NCBI names are close according to this edit distance. The risk is now to merge genes that are different but whose names are similar (for instance, ND4 and ND4L are two different mitochondrial genes, but with similar names). To fix such a flaw, the *sequence* similarity of intersected genes in a genome is also compared in a second stage, using a Needleman-Wunsch global alignment algorithm. The genes correspondence is simply ignored if this similarity is below a predefined threshold. We call this operation, which will result in a set of quality genes, a “quality test”. A result from

this quality test process is a set of quality genes. These genes will then constitute the quality genomes as given in Definition 5. A list of generated quality genomes based on a specific threshold will construct the intersection core matrix to generate the core genes, core tree, and the phylogenetic tree after choosing an appropriate outgroup.

It is important to note that DNA sequence annotation raises a problem in the case of DOGMA: contrary to what happens with gene features in NCBI, genes predicted by DOGMA annotation might be fragmented in several parts. Such genes are stored in the gene-vision file format produced by DOGMA, as each fragment is in this file with the same gene name. A gene whose name is present at least twice in the file is either a duplicated gene or a fragmented one. Obviously, fragmented genes must be defragmented before the DNA similarity computation stage (remark that such a defragmentation has been already realized on NCBI website). The defragmentation consists in concatenating all possible permutations (in the case of duplication), and keeping only the permutation with the best similarity score in comparison with other sequences having the same gene name, if this score is larger than the given threshold.

6.2.3.1/ CONSTRUCTION OF QUALITY GENOMES

The first step in producing annotated genomes is to find the set of common genes, that is, genes sharing similar names and sequences, by using various annotation tools and following the method described previously. Figure 6.9(a) presents the original amount

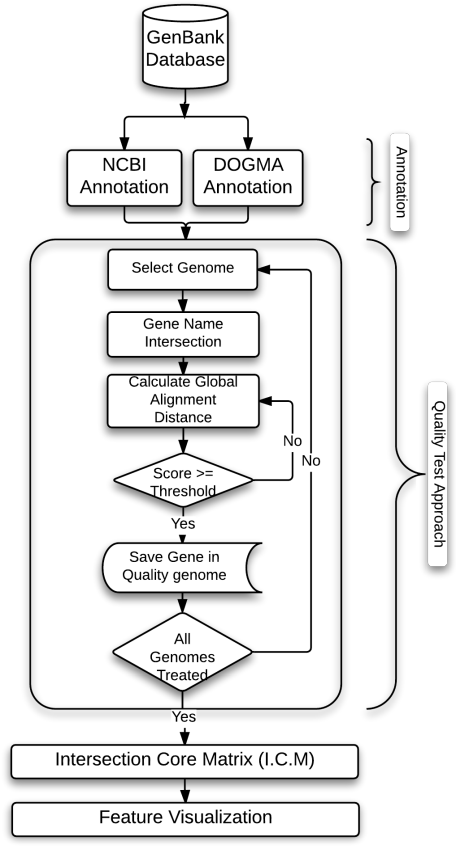


Figure 6.6: An overview of the pipeline.

Genome: NC_001713.1.fasta Threshold= 60 Mode:osneedle
Genes in NCBI: 138 Genes in Dogma: 155
Common Genes: 119 NCBI: 86.23% Dogma: 76.77%

No	Gene	Len_NC	Len_Do	N.Start	N.Stop	D.Start	D.Stop	Score
1.	RPOC2	4446	4442	ATG	TGA	TGA	AAT	99.90
2.	YCF4	546	542	ATG	TAA	TGC	TTA	99.30
3.	YCF3	540	536	ATG	TAA	ATG	TTT	99.30
4.	RPOB	4140	4136	ATG	TAA	TGA	TTT	99.90
5.	RPOA	939	935	ATG	TAA	ATG	TAA	99.60
6.	ATPH	249	245	ATG	TAA	TGG	GGT	98.40
7.	YCF47	225	221	ATG	TGA	TGC	GGA	98.20
8.	YCF46	1494	1490	ATG	TAA	ATG	TAA	99.70
9.	YCF45	1368	1364	ATG	TAA	ATG	TAA	99.70
10.	ATPG	471	467	ATG	TAA	TGG	ATT	99.20
11.	ATPF	540	536	ATG	TGA	TGG	TTA	99.30
12.	YCF41	342	338	ATG	TAA	ATG	TAA	98.80
13.	ATPD	564	560	ATG	TAA	TGA	ATT	99.30
14.	YCF24	1461	1457	ATG	TAA	TGA	GGT	99.70
15.	RPL14	366	362	ATG	TAA	ATG	TCT	98.90
16.	RPL16	414	410	ATG	TAA	ATG	AAT	99.00
17.	RPL11	426	422	ATG	TAA	ATG	AGA	99.10
18.	RPL12	384	380	ATG	TAA	ATG	AAA	99.00
19.	RPL13	420	416	ATG	TAA	ATG	TAT	99.00
20.	RPL18	408	404	ATG	TAA	ATG	ATT	99.00
21.	RPL19	363	359	ATG	TAA	TGT	CGA	98.90
22.	THIG	786	782	ATG	TAG	TGA	CAT	99.50
23.	RPS2	690	686	ATG	TAA	TGG	AAA	99.40
24.	YCF39	960	956	ATG	TAA	ATG	AAT	99.60
25.	RPL6	540	536	ATG	TAA	ATG	AAA	99.30
26.	RPL4	648	644	ATG	TGA	ATG	TGG	99.40
27.	RPL5	669	713	TTG	TAA	ATG	AGA	92.70
28.	RPL2	828	824	ATG	TAA	ATG	TTC	99.50
29.	RPL3	624	587	GTG	TAA	ATG	AAA	94.10
30.	RPL1	693	689	ATG	TAA	ATG	TCT	99.40
31.	FTSH	1935	1931	ATG	TAA	TGA	AAC	99.80

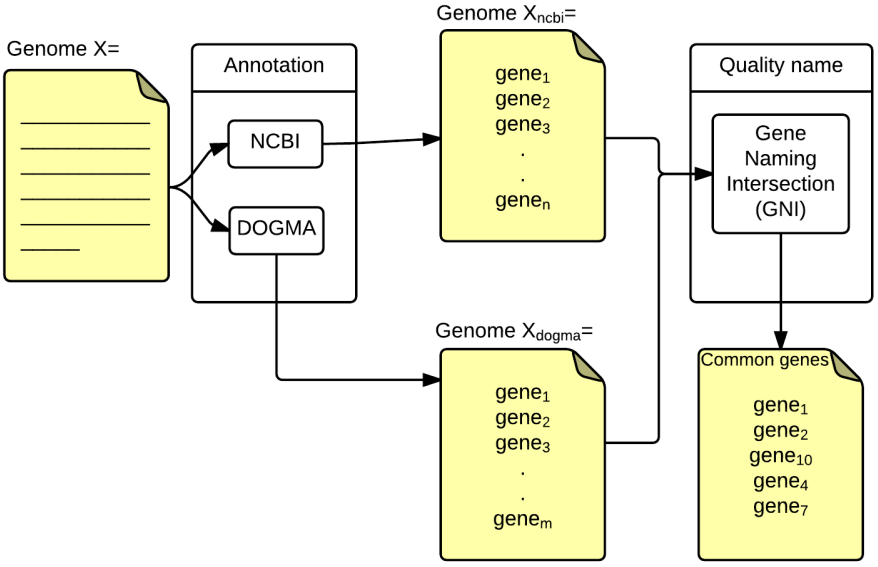
.

Figure 6.7: Part of the implementation of the third method, sequence comparison of the common genes from NCBI and DOGMA. In this figure, each record have information of selected common gene such as *gene name, sequence length from NCBI* and *DOGMA annotations, start* and *stop codons for both annotations*, and *the sequence matching score value*. Note that the gene column comes from producing common genes process (see Figure 6.8(a))

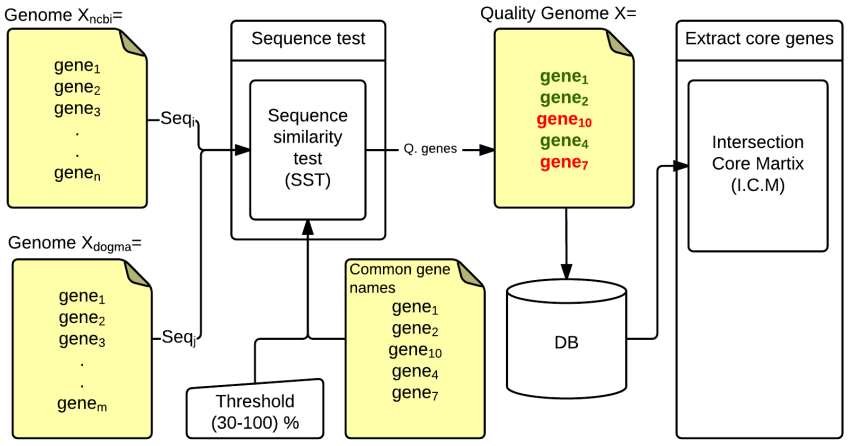
of genes based on NCBI and DOGMA annotations. Two quality test routines then take place to produce “quality genomes” as shown in Figure 6.8 by: (1) selecting all common genes based on gene names (see Figure 6.8(a)) and (2) checking the similarity of sequences (see Figure 6.8(b)), which must be larger than or equal to a predefined threshold (see Figure 6.9(a)).

Remark 7: Threshold usage

The predefined threshold is not used to determine the orthologuous genes, but to to ensure that core genes from NCBI and DOGMA annotations are identical.



(a) Producing sharing genes based on gene names.



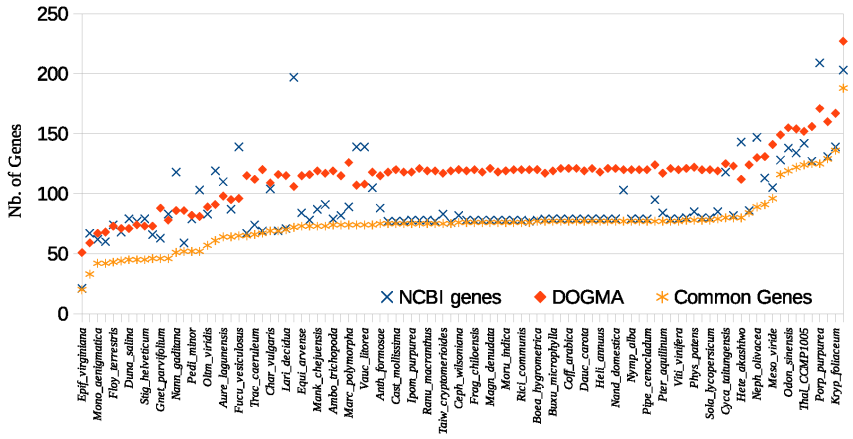
(b) Producing quality genomes.

Figure 6.8: demonstration of quality test approach pipeline. (a) The process of extracting quality genes based on gene features (*e.g.*, gene names). (b) The process of predicting the quality genomes based on quality genes from previous step.

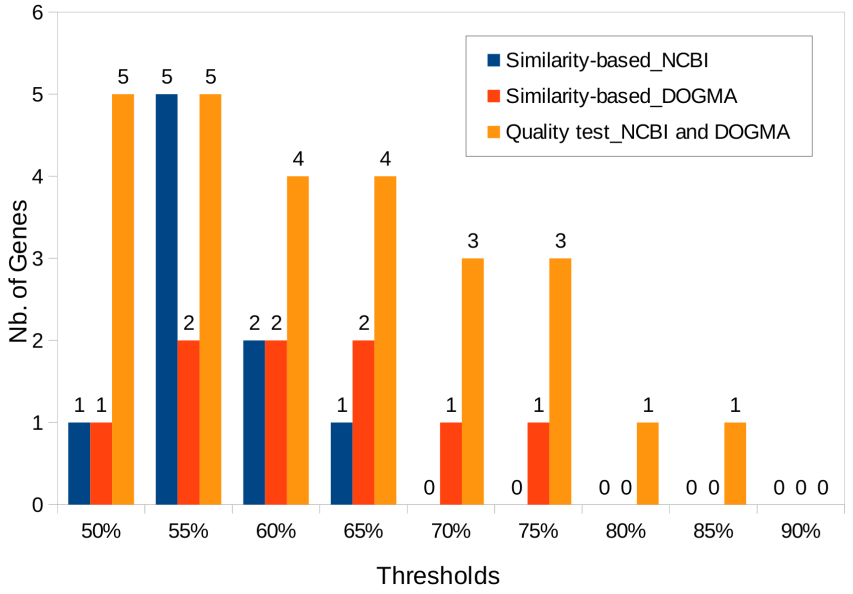
6.2.3.2/ CORE AND PAN GENOMES

To produce core genomes based on quality control approach[2], we need to know what are the common genes that share almost the same name and sequence from different annotation tools. Figure 6.11(a) shows the original amount of genes based on two annotation

[2]see http://members.femto-st.fr/christophe-guyeux/en/chloroplasts



(a) Amount of genes based on NCBI and DOGMA w.r.t quality common genes. DOGMA gives the larger number of genes.



(b) Core genomes sizes w.r.t. threshold. A maximal number of core genes does not mean a good core genomes: we are looking for genes meeting biological requirements.

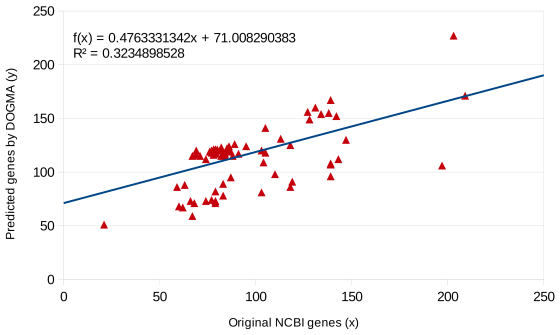
Figure 6.9: (a) Genes coverage for a threshold of 60% and (b) core genomes sizes.

tools. We also calculate the correlation coefficient (r) by applying the usual formula:

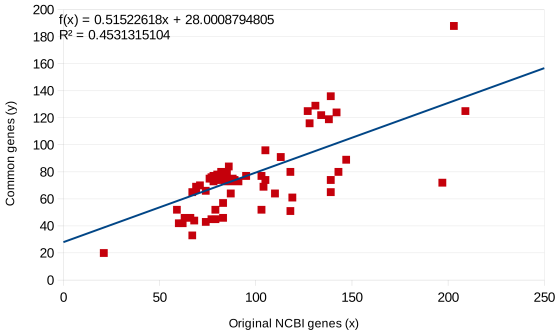
$$r_{xy} = \frac{\sum_{i=0}^{n-1}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=0}^{n-1}(x_i - \bar{x})^2 \sum_{i=0}^{n-1}(y_i - \bar{y})^2}} \tag{6.2}$$

where x, y are sample data (nb. of genes from two annotation algorithms), $\bar{x}, \bar{y}$ are the sample mean for x, y, and n is the number of genomes. We found that the correlation value based on the number of genes produced by the two annotation algorithms is $r = 0.57$ (see Figure 6.10(a)), which means that the two ways to obtain annotations are really different.

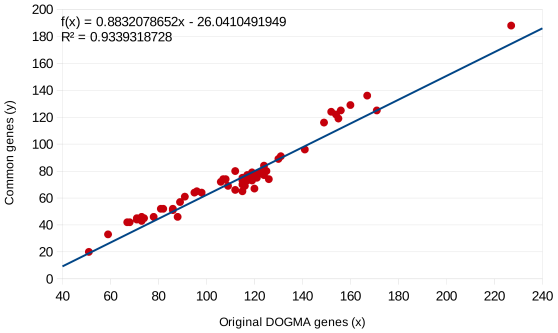
Two steps quality test routines have then been launched to produce “quality genomes” and to enlarge the correlation: (1) select all common genes based on gene names, (2)



(a) Correlation coefficient between predicted NCBI and DOGMA genes.



(b) Correlation coefficient between predicted NCBI and common genes.



(c) Correlation coefficient between predicted DOGMA and common genes.

Figure 6.10: Correlation coefficient between predicted NCBI and DOGMA annotations and predicted common genes

check the similarity of sequences, which must be larger than a predefined threshold. Table 6.3 summarize the results of annotating 98 chloroplast genomes. In this table, X and Y represents the number of genes obtained from NCBI and DOGMA annotations from a given genome. $X \cap Y$ specifies the number of common genes (quality genes) between genome X_i and genome Y_i. The last two columns give the covering percentage of common genes to the current ones.

Based on the values in Table 6.3, Figure 6.11(b) presents the genes coverage percentage between NCBI and DOGMA. The correlation value based on the number of genes between the produced quality genomes and NCBI genomes is $r = 0.6731$ (see Figure 6.10(b)), and $r = 0.9664$ between produced quality genomes and DOGMA ones (see Figure 6.10(c)). Such correlation coefficients illustrate the large variability in the quality of NCBI annotations, and the average stability in the DOGMA ones. Obviously, these differences between the annotation tools can affect the final core genome.

Remark 8: Possible origins of differences between NCBI and DOGMA

The number of *tRNAs* and *rRNAs* genes is very large in the case of DOGMA while they are very low in the NCBI case. Additionally, unnamed or misspelled genes are frequent in the NCBI annotations.

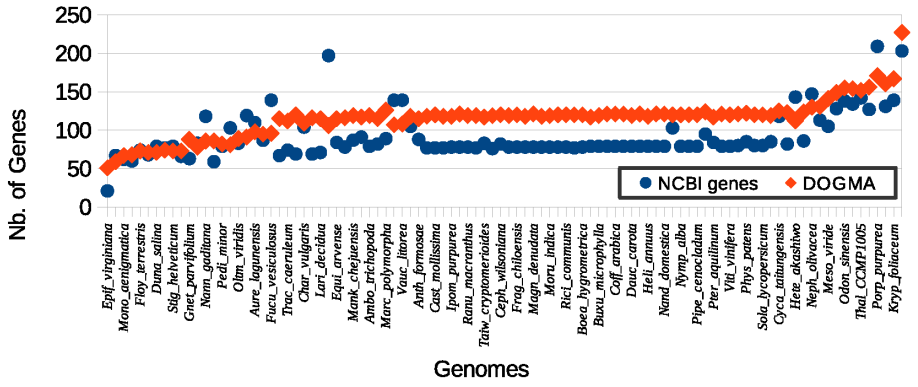
Figure 6.9(b) represents the amount of genes in the computed core genome of 98

Genome	X	Y	$X \cap Y$	NCBI (%)	DOGMA (%)
Epif_virginiana	21	51	20	95.24	39.22
Eugl_gracilis	67	59	33	49.25	55.93
Mono_aenigmatica	62	67	42	67.74	62.69
Ostr_tauri	60	68	42	70	61.76
Floy_terrestris	74	73	43	58.11	58.9
Pycn_provasolii	68	71	44	64.71	61.97
Duna_salina	79	71	45	56.96	63.38
Schi_leibleinii	77	74	45	58.44	60.81
Stig_helveticum	79	73	45	56.96	61.64
Chla_reinhardtii	66	73	46	69.7	63.01
Gnet_parvifolium	63	88	46	73.02	52.27
Oedo_cardiacum	83	78	46	55.42	58.97
Nann_gaditana	118	86	51	43.22	59.3
Cusc_gronovii	59	86	52	88.14	60.47
Pedi_minor	79	82	52	65.82	63.41
Pseu_akinetum	103	81	52	50.49	64.2
Oltm_viridis	83	89	57	68.67	64.04
Emil_huxleyi	119	91	61	51.26	67.03
Aure_lagunensis	110	98	64	58.18	65.31
Pyra_parkeae	87	95	64	73.56	67.37
Fucu_vesiculosus	139	96	65	46.76	67.71
Phal_equestris	67	115	65	97.01	56.52
Trac_caeruleum	74	112	66	89.19	58.93
Euca_grandis	69	120	67	97.1	55.83
Char_vulgaris	104	109	69	66.35	63.3
Pice_morrisonicola	69	116	69	100	59.48
Lari_decidua	71	115	70	98.59	60.87
Cyan_caldarium	197	106	72	36.55	67.92
Equi_arvense	84	115	73	86.9	63.48
Jasm_nudiflorum	78	116	73	93.59	62.93
Mank_chejuensis	87	119	73	83.91	61.34
Popu_trichocarpa	91	117	73	80.22	62.39
Ambo_trichopoda	79	119	74	93.67	62.18
Isoe_flaccida	82	115	74	90.24	64.35
Marc_polymorpha	89	126	74	83.15	58.73
Sacc_japonica	139	107	74	53.24	69.16
Vauc_litorea	139	108	74	53.24	68.52
Zea_mays	105	118	74	70.48	62.71
Anth_formosae	88	115	75	85.23	65.22
Bamb_emeiensis	77	118	75	97.4	63.56
Cast_mollissima	77	120	75	97.4	62.5
Goss_darwinii	77	118	75	97.4	63.56
Ipom_purpurea	78	118	75	96.15	63.56
Olea_europaea	78	121	75	96.15	61.98
Ranu_macranthus	78	119	75	96.15	63.03
Sile_vulgaris	77	119	75	97.4	63.03
Taiw_cryptomerioides	83	117	75	90.36	64.1
Theo_cacao	76	119	75	98.68	63.03
Ceph_wilsoniana	82	120	76	92.68	63.33
Dios_elephantipes	78	119	76	97.44	63.87
Frag_chiloensis	78	120	76	97.44	63.33
Lact_sativa	78	118	76	97.44	64.41
Magn_denudata	78	121	76	97.44	62.81
Mani_esculenta	78	118	76	97.44	64.41
Moru_indica	78	119	76	97.44	63.87
Oeno_biennis	78	120	76	97.44	63.33
Rici_communis	78	120	76	97.44	63.33
Wolf_australiana	77	120	76	98.7	63.33
Boea_hygrometrica	78	120	77	98.72	64.17
Bras_napus	79	117	77	97.47	65.81
Buxu_microphylla	79	119	77	97.47	64.71
Chlo_spicatus	79	121	77	97.47	63.64
Coff_arabica	79	121	77	97.47	63.64
Colo_esculenta	79	121	77	97.47	63.64
Dauc_carota	79	119	77	97.47	64.71
Drim_granadensis	79	121	77	97.47	63.64
Heli_annuus	79	118	77	97.47	65.25
Illi_oligandrum	79	121	77	97.47	63.64
Nand_domestica	79	121	77	97.47	63.64
Nico_undulata	103	120	77	74.76	64.17
Nymp_alba	79	120	77	97.47	64.17
Pana_ginseng	79	120	77	97.47	64.17
Pipe_cenocladum	79	120	77	97.47	64.17
Psil_nudum	95	124	77	81.05	62.1
Pter_aquilinum	84	117	77	91.67	65.81
Typh_latifolia	79	121	77	97.47	63.64
Viti_vinifera	79	120	77	97.47	64.17
Phoe_dactylifera	80	121	78	97.5	64.46
Phys_patens	85	122	78	91.76	63.93
Sesa_indicum	80	120	78	97.5	65
Sola_lycopersicum	80	120	78	97.5	65
Angi_evecta	85	119	79	92.94	66.39
Cyca_taitungensis	118	125	80	67.8	64
Gink_biloba	82	123	80	97.56	65.04
Hete_akashiwo	143	112	80	55.94	71.43
Hupe_lucidula	86	124	84	97.67	67.74
Neph_olivacea	147	130	89	60.54	68.46
Chlo_atmophyticus	113	131	91	80.53	69.47
Meso_viride	105	141	96	91.43	68.09
Ulna_acus	128	149	116	90.62	77.85
Odon_sinensis	138	155	119	86.23	76.77
Fist_DA0580	134	154	122	91.04	79.22
Thal_CCMP1005	142	152	124	87.32	81.58
Duri_baltica	127	156	125	98.43	80.13
Porp_purpurea	209	171	125	59.81	73.1
Phae_tricornutum	131	160	129	98.47	80.62
Kryp_foliaceum	139	167	136	97.84	81.44
Grac_liui	203	227	188	92.61	82.82

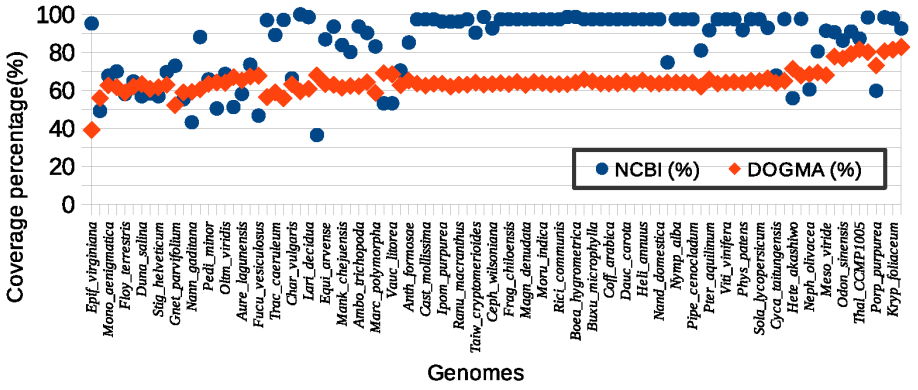
Table 6.3: Number of common genes obtained from NCBI and DOGMA annotations.

species. In this figure, two methods are used and compared using the same sample of genomes: in the first one, the gene prediction approach presented previously in Section 6.2.1 and published in [47, 48] has been used on genomes annotated by NCBI and DOGMA, while on the other one the quality test approach also published in [48] is applied on genomes annotated by DOGMA. Different thresholds have been examined for both approaches. The amount of final core genes within the two approaches is low, as the species considered here are highly divergent. However even in that particular situation, it is obvious that the quality test approach outperforms the other one for each tested threshold.

As stated previously, the main goal is to find the largest number of core genes compatible with biological background related to chloroplasts. In the quality test approach case, one genome (*Micromonas pusilla*, with accession number NC_012568.1) has been discarded from the sample, as we observed that this genome always has the lowest number of common genes in our selected data set. This latter can be explained by two reasons: (1) either one or more genomes consists of non-functional genes, or (2) the diversity is too large. With quality approach, an absence of genes in rooted core genome means



(a) Sizes of genomes based on NCBI and DOGMA annotations.



(b) Percentage of genes coverage between NCBI and DOGMA.

Figure 6.11: Original and coverage sizes between NCBI and DOGMA genomes based on a threshold of 60%. (a) The number of genes with DOGMA is larger than the ones with NCBI, because the former generates more tRNAs and rRNAs genes than NCBI. (b) The former outperforms the latter, as almost all genes in NCBI genomes have been covered with common genes, while most of the DOGMA genes are ignored. However, correlation of them with NCBI (after quality test) is 0.6731, while it is 0.9664 with DOGMA, this latter being thus more accurate than NCBI.

that we have two or more subtrees of organisms completely divergent among each other. Unfortunately, for the first approach with NCBI annotation, the generated cores did not provide a good biological distribution of targeted genomes. More precisely, *Micromonas pusilla* evoked previously is the only genome that totally destroys the final core genome with NCBI annotations[3], for both gene features and gene quality methods. Conversely, in the case of DOGMA annotation, the distribution of genomes is biologically relevant.

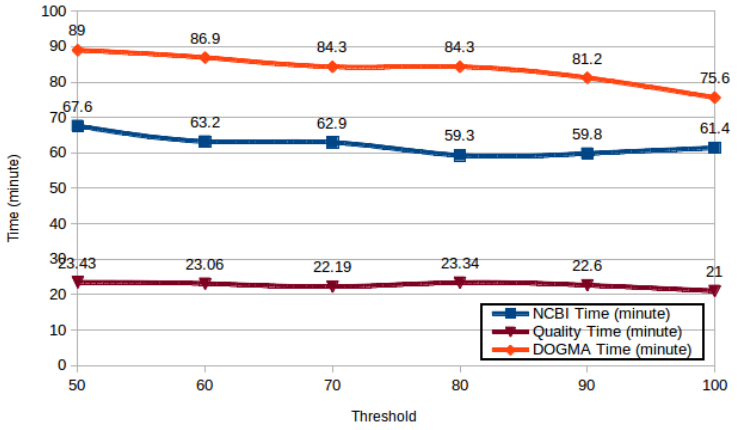
6.2.3.3/ EXECUTION TIME AND MEMORY USAGE

In computational biology, time and memory consumptions are two important factors due to high throughput operations among gene sequences. Figure 6.12 shows the amount of time and memory needed to extract core genes using the two approaches: in the first one, building the connected components depends on the construction of a distance matrix by

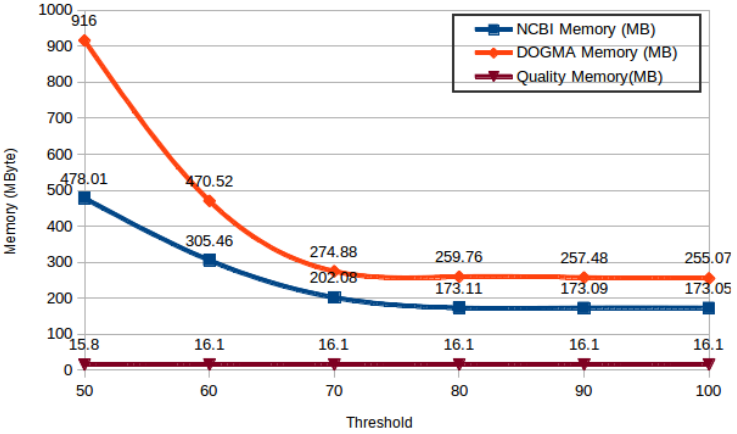
[3]NCBI cores are available on footnote 2.

considering the similarity scores from the global alignment tool, which takes a long time in the case of NCBI and DOGMA genomes.

Computation time is different for DOGMA and NCBI due to the size of genomes and the amount of gene sequences that need to be compared: NCBI genomes have 8,992 genes, instead of 11,242 in DOGMA genomes. Figure 6.12(a) presents the execution time needed for each method with respect to thresholds in range $[50 - 100]$. The DOGMA one requires more computational time (in minutes) for sequence comparisons, while gene quality method needs a low execution time to compare quality genes. Let us notice that once the "quality genomes" have been constructed, this method takes only 1.29 minutes to extract core genes on a personal computer running Ubuntu 12.04 32 bits with 6 Giga bytes of memory, and a quad-core Intel Core i5 processor with an operating frequency of 2.5 GHz.



(a) Time needed to execute each method.



(b) Memory usage (MB unit) (sizes usually available on personal computers).

Figure 6.12: Execution time and memory usage w.r.t. threshold.

The second important factor is the amount of memory used by each methodology, this one is highlighted in Figure 6.12(b). The low values show that the gene quality method based on gene sequence comparisons presents the most reasonable memory usage (when constructing quality genomes). It also depends on the size of genomes. Determining which method to choose depends on the user preferences: if we search for a fast

and semi-accurate method, then the second approach should be chosen. Otherwise, if an accurate but relatively slow approach is desired, then the first method with DOGMA annotations should be preferred.

6.3/ FEATURES VISUALIZATION

6.3.1/ THE CORE TREE

The last stage of the proposed pipeline is to take advantage of the produced core and pan genomes for biological studies. As this key stage is not directly related to the methodology for core and pan genomes discovery, we will only outline few tasks that can be done using the produced data.

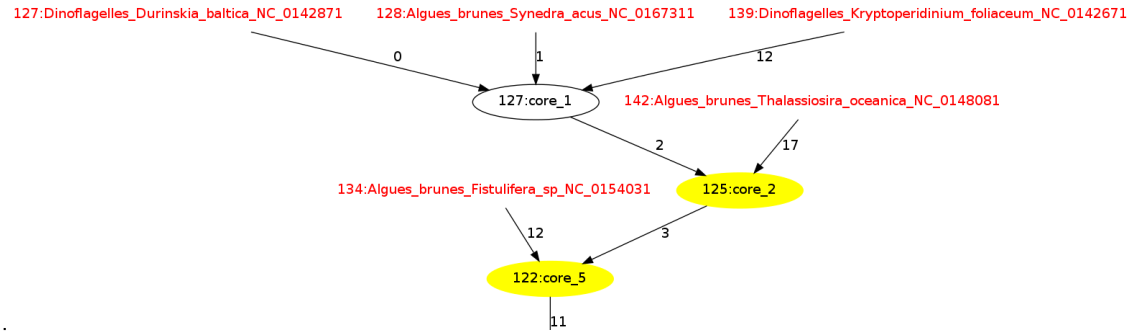


Figure 6.13: Part of a core genomes evolutionary tree (NCBI gene names)

Obtained results may be visualized by building a core genomes evolutionary tree, simply called *core tree* or a *dendrogram* in some literature. Each node in this tree represents a chloroplast genome of a predicted core, as depicted in Figure 6.13. In this figure, nodes labels are of the form *(Genes number:Family name_Scientific name_Accession number)*, while an edge is labeled with the number of gene loss when compared to its parents (a leaf genome or an intermediate core genome). Such numbers can answer questions like: how many genes are different between two species? Which functionality has been lost between an ancestor and its children? For complete core trees based either on NCBI names or DOGMA ones, see Footnote 2.

A second application of such data is obviously to build accurate phylogenetic trees, using tools like PHYML [60] or RAxML [61]. Consider, given a set of species, the least common core genome in a core tree that contains all shared common genes among these species. To infer a phylogenetic tree, these core genes can be multi-aligned to serve as an input to any phylogenetic tool mentioned above. Using core genomes here guarantee to build the phylogenetic tree on the largest possible common coding sequences of the considered species. An example of such a phylogenetic tree is provided in Figure 6.14, it is investigated more deeply in the next subsection.

6.3.2/ A FIRST PHYLOGENETIC STUDY

Having a common set of DNA sequences shared by all the chloroplasts thanks to our quality core genes approach, we can now focus on the first objective of our thesis, namely to infer their phylogeny based on their core genome. At this point, we do not consider all

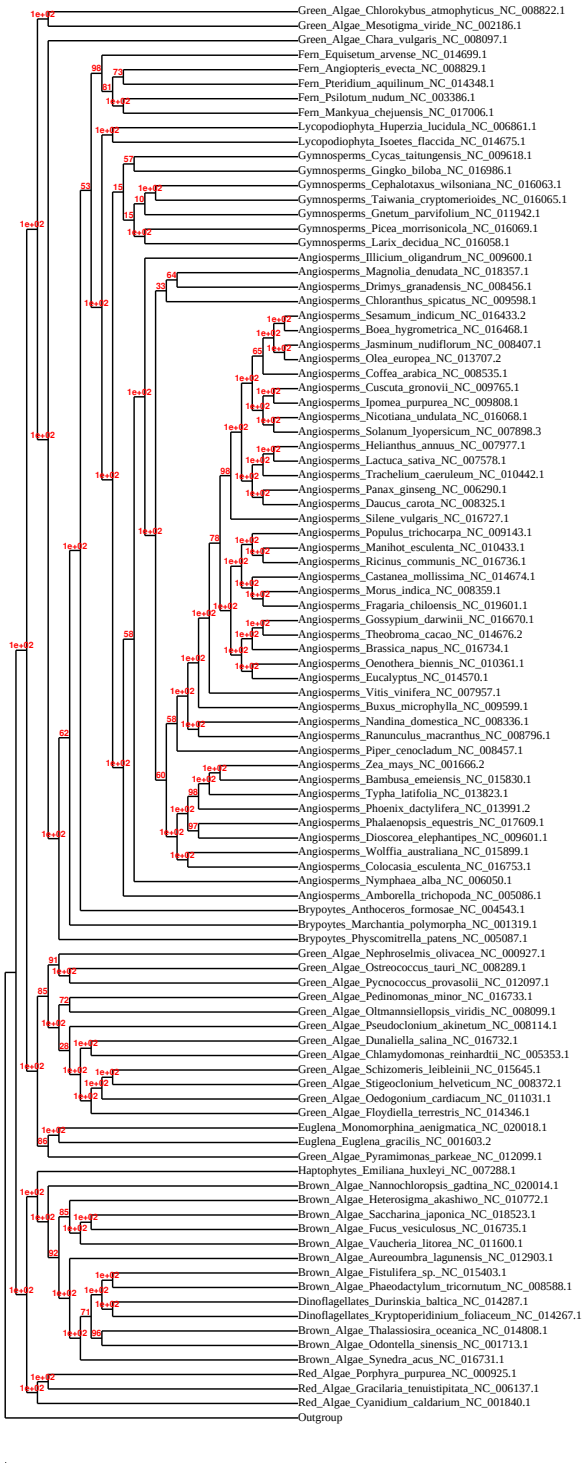


Figure 6.14: Phylogenetic tree based on DOGMA annotation.

available chloroplastic genomes, but we still focus on the 5 core genes of the 98 plant species used during our core and pan genome investigations.

To obtain such a tree, the RAxML [62] program has been employed to compute the phylogenetic maximum-likelihood (ML) function with the following setup: the General Time Reversible model of nucleotide substitution with the Γ model of rate heterogeneity and

the hill-climbing optimization method, while the *Prochlorococcus marinus* (NC_009091.1) cyanobacteria species is chosen as outgroup due to the supposed cyanobacteria origin of chloroplasts. The tree representation is obtained with Geneious [63] based on the RAxML information.

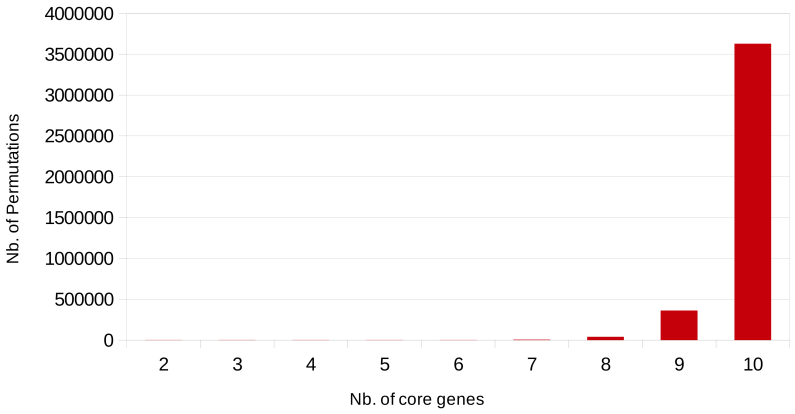


Figure 6.15: Amount of permutations w.r.t the number of core genes.

We first wonder whether the way to order the core genes in the alignment file may impact the inferred phylogeny. Thus, in order to find a well supported phylogenetic tree from all core genes, we have achieved the computation of 120 bootstrapped trees by considering all possible permutations, using *itertools* package, of the 5 core genes (remark that having a core genome larger than 7 genes leads to a searching space whose size explodes, see Figure 6.15). Among all these trees, we have then selected the one with the largest value of its lowest bootstrap b, this latter being denoted as the most accurate tree (MAT) in what follows, after having verified that gene order has no effect on the supports.

The obtained MAT has a lowest bootstrap equal to $b = 32$, which is very low. To improve this value, we have investigated in a second stage of experiments whether some core genes impact the robustness of the tree, for instance because they are homoplasic ones (see Table 6.4). In fact, when the core is large enough, it is possible to remove a few of them that obviously break the supports according to the maximum likelihood inference. After having systematically removing 1, 2, 3, and 4 genes, the best phylogenetic tree, having its lowest bootstrap value equal to 35, was obtained with one gene loss.

The low improvement previously observed when removing some core genes suggests that their number is not sufficient to produce a well-supported phylogenetic tree. Therefore we decided for the second experiment to split the set of species in two and to work with the core genome of the largest subset: 52 genomes lead to a core genome of 16 genes[4] (Core_81 in the core tree available online). As expected, working with this large core genome allows to really improve the lowest bootstrap value[5], since by removing randomly 1, 2, 3, and 4 genes the resulting MAT has 55 for its lowest bootstrap value. Figure 6.16 presents this best tree obtained after removing one gene (*atpI*). Let us notice that, for large core genomes such a systematic approach is intractable in practice, due to the dramatic number of core genes combinations to calculate (next chapters will investigate more deeply this scaling problem).

[4]Core genes in Core_81: *psbE, psbD, petG, psbF, psbA, psbC, rpl36, psbN, psbI, psbJ, atpH, psaJ, atpI, atpA, psaA, and psaC.*

[5]The lowest bootstrap value for 16 core genes is 15.

Core genes	Permutations	Combinations			
		rem. $gene_1$	rem. $gene_2$	rem. $gene_3$	rem. $gene_4$
2	2	2	0	0	0
3	6	3	0	0	0
4	24	4	6	0	0
5	120	5	10	10	0
6	720	6	15	20	15
7	5040	7	21	35	35
8	40320	8	28	56	70
9	362880	9	36	84	126
10	3628800	10	45	120	210

Table 6.4: Amounts of trees w.r.t removing homoplasy genes.

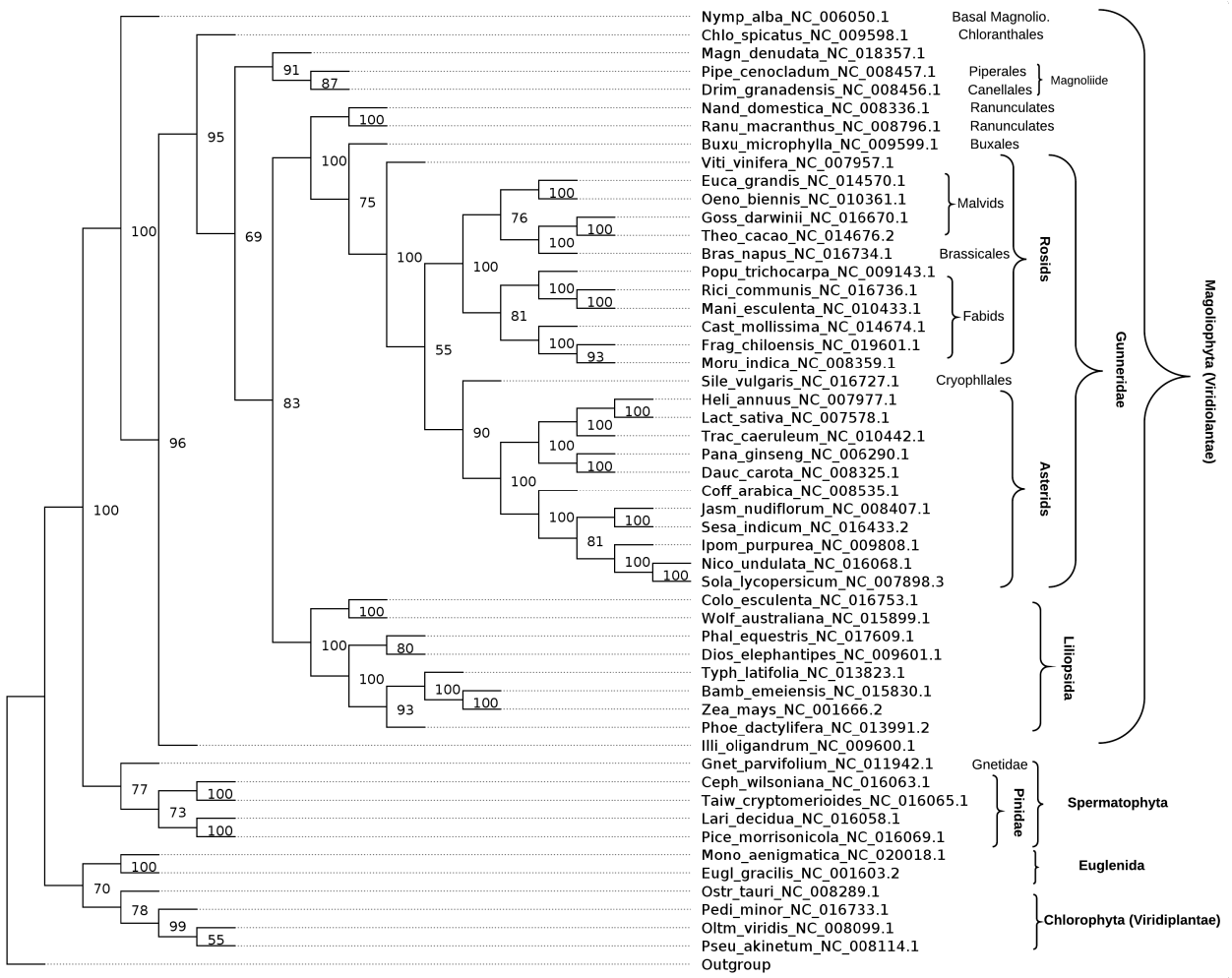


Figure 6.16: Core_81 phylogenetic tree with 15 core genes (1 gene removed randomly).

Finally, the support of the best phylogenetic tree can be improved again by using the whole knowledge inherited by all the constructed trees, that is, by merging all trees computed when removing genes. *SuperTripletes* [64] is one of the methods that can infer a

supertree from a collection of bootstrapping phylogenetic trees. This tool[6] receives a file that stores all bootstrap values. In this last experiment, phylogenetic trees with 1, 2, 3, and 4 random gene loss have been concatenated in one file and transmitted to *Super-Tripletes*. The obtained supertree with all taxa is provided in Figure 6.17. It can be seen that the minimum bootstrap has been further improved to 64.

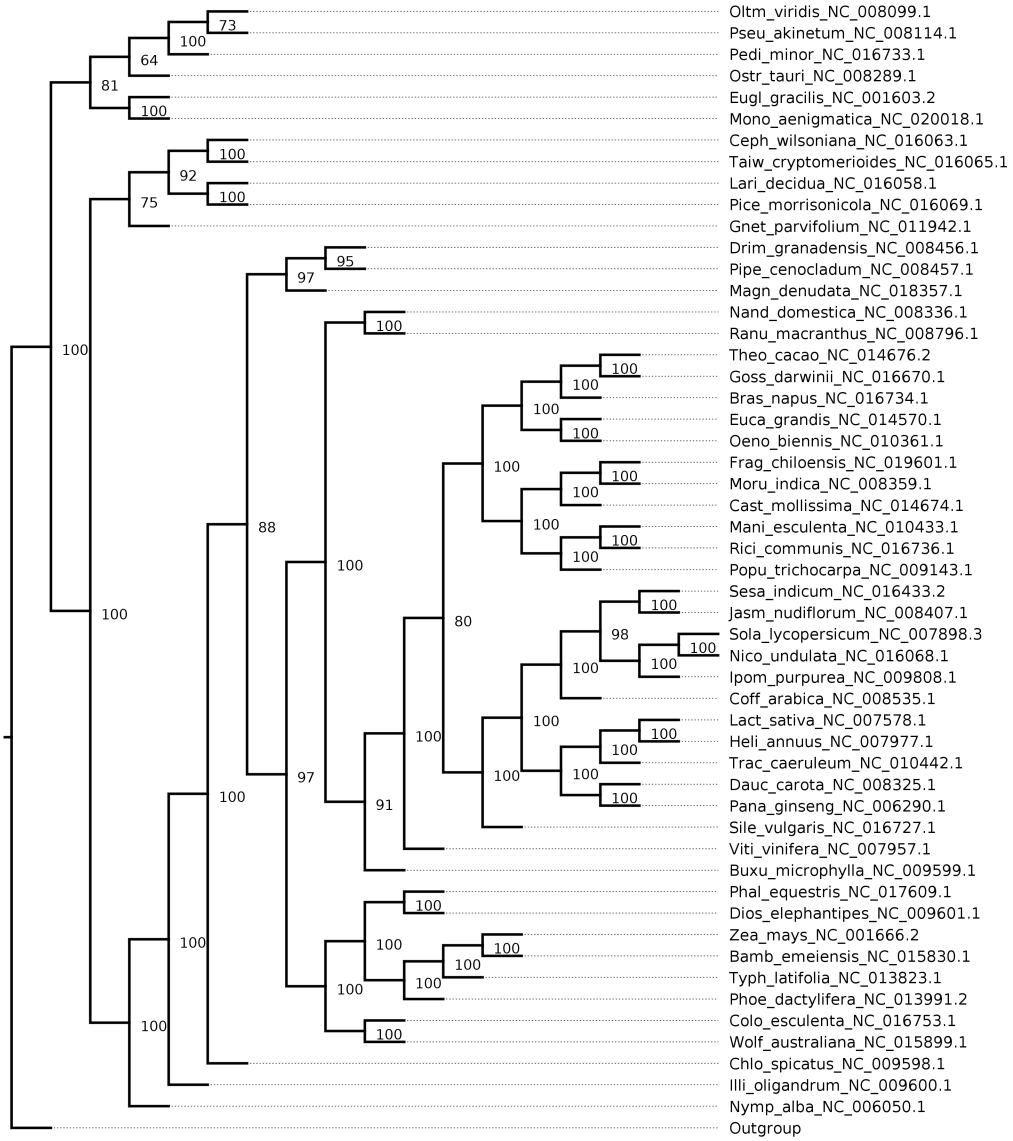


Figure 6.17: Supertree for Core_81 from 248 bootstrap phylogenetic trees after removing 1, 2, 3, or 4 genes randomly.

6.4/ DISCUSSION AND BIOLOGICAL EVALUATION

It is well known that the first plants' endosymbiosis ended in a glorious diversification of lineages comprising *Red Algae*, *Green Algae*, and *Land Plants* (terrestrial). Several second endosymbioses occurred then: two involving a *Red Algae* and other heterotrophic

[6] Available on http://www.supertriplets.univ-montp2.fr/index.php

eucaryotes and giving birth to both *Brown Algae* and *Dinoflagellates* lineages; another involving a *Green Algae* and a heterotrophic eucaryote and giving birth to *Euglens* [65].

The interesting point with the produced core trees (especially the one obtained with DOGMA, see 2) is that the organisms resulting from the first endosymbiosis are distributed in each of the lineages found in the chloroplast genome structure evolution. More precisely, all *Red Algae* chloroplasts are grouped together in one lineage, while *Green Algae* and *Land Plant* chloroplasts are all in a second lineage. Furthermore, organisms resulting from the secondary endosymbioses are well localized in the tree: both the chloroplasts of *Brown Algae* and *Dinoflagellates* representatives are found exclusively in the lineage also comprising the *Red Algae* chloroplasts from which they evolved, while the *Euglens* is related to *Green Algae* from which they evolved. This latter makes sense in terms of biology, history of lineages, and theories of chloroplasts origins (and so photosynthetic ability) in different Eucaryotic lineages [65].

Interestingly, the sole organisms under consideration that possess a chloroplast (and so a chloroplastic genome) but that have lost the photosynthetic ability (being parasitic plants) are found on the basis of the tree, and not together with their phylogenetically related species. This latter means that functional chloroplast genes are evolutionary constrained when used in the photosynthetic process, but lose their efficiency rapidly when not used, as recently observed for a species of Angiosperms [66]. These species are *Cuscuta gronovii*, an Angiosperm (flowering plant) at the base of the DOGMA Angiosperm-Conifers branch, and *Epifagus virginiana*, also an Angiosperm, at the complete basis of the DOGMA core tree.

Another interesting result is that *Land Plants* that represent a single sub-lineage originating from the large and diverse lineage of *Green Algae* in Eucaryotes history are present in two different branches of the DOGMA tree, both associated with *Green Algae*: one branch comprising the basal grade of *Land Plants* (mosses and ferns) and the second one containing the most internal lineages of *Land Plants* (conifers and flowering plants). Independently of their split in two distinct branches of the DOGMA tree, the *Land Plants* always show a larger number of functional genes in their chloroplasts than the *Green Algae* from which they emerged, probably meaning that the terrestrial way of life necessitates more functional genes for an optimal photosynthesis than the marine one. However, a more detailed analysis of selected genes is necessary to understand better the reasons why such a distribution has been obtained.

Remark 9: Biological relevance of the results

All biologically interesting results are apparent only in the core tree based on DOGMA, while they are not obvious in the NCBI one.

6.5/ CONCLUSION

In this chapter, we studied three methodologies for extracting core genes from a large set of chloroplast genomes, and we developed python programs to evaluate them.

We firstly considered extracting core genomes by the way of comparisons (global alignment) of DNA sequences downloaded from NCBI database. However, this method failed to produce biologically relevant core genomes, no matter the chosen similarity threshold, probably due to annotation errors. We then considered to use the DOGMA annotation

tool to enhance the gene prediction process.

The second method consisted in extracting gene names either from NCBI gene features or from DOGMA results. At the beginning an “intersection core matrix (ICM)” is built. In this matrix, each coefficient store the intersection cardinality of the two genomes placed at the extremities of its row and column. New ICMs are then successively constructed by selecting the maximum intersection score (IS) in this matrix, removing each time the two genomes having this score and adding the corresponding core genome in the next ICM construction.

Finally, we have employed a third method named “quality test approach” to extract core genes from a large set of chloroplastic genomes, and we compared it with the gene prediction approach developed at the beginning of this chapter. A two stage similarity measure, on names and sequences, has thus been proposed for clustering DNA sequences in genes, which merges the best results provided by NCBI and DOGMA. Results obtained with this quality control test have finally been deeply compared with our previously obtained results.

Core trees have been generated for each method, to investigate the distribution of chloroplasts and core genomes. The tree from second method, based on DOGMA, has revealed the best distribution of chloroplasts regarding their evolutionary history. In particular, it appears that each endosymbiosis event is well branched in the DOGMA core tree. Phylogenetic trees have finally been generated to investigate the distribution of chloroplasts and core genomes. We performed intensive computations on the mésocentre supercomputing facilities to produce the highest bootstrap valued tree by generating all the trees resulting from different gene orders and random removing of genes in the core genome. A supertree is then generated, leading to a quite accurate phylogenetic tree for a large amount of plant species.

In next chapter, we will study the gene content of each given core genome, and phylogenetic relations between all these species will be investigated too.

CHAPTER 7

Inferring Phylogenetic Trees using Genetic Algorithm

We now consider that, given a set of complete chloroplastic genomes, we are able to annotate them well and to extract their core genes. The next problem, in the quest of the last universal common ancestor, is to use them to obtain a well-supported phylogenetic tree.

The contribution of this chapter can be summarized as follows. We focus on situations where a large number of genes are shared by a set of species so that, in theory, enough data are available to produce a well-supported phylogenetic tree. However, a few genes tell a different evolutionary scenario than the majority of sequences, leading to phylogenetic noise blurring the phylogeny reconstruction. In this chapter, we propose a pipeline that attempts to solve such an issue by computing all phylogenetic trees that can be obtained by removing at most one core gene. In the case where such a preliminary systematic approach does not solve the phylogeny, new investigation stages are added to the pipeline, namely a Monte-Carlo based random approach and two invocations of a genetic algorithm, separated by a Lasso test. The pipeline is finally tested on various sets of chloroplast genomes. Note that this contribution has been presented and published in [67], the international conference Algorithms for Computational Biology (AlCoB 2015).

7.1/ General Presentation

The multiplication of complete chloroplast genomes should normally lead to the ability to infer trustworthy phylogenetic trees for plant species. Indeed, the existence of trustworthy coding sequence prediction and annotation softwares specific to chloroplasts (like DOGMA), with the right control of sequence alignment, and maximum likelihood or Bayesian inference phylogenetic reconstruction techniques, should imply the capability to determine accurately the sister relationship between species. More precisely, given a set of close species, their core genome (the set of genes in common) can be as large and accurately detected as possible to finally obtain a well-supported phylogenetic tree. However, all genes of the core genome are not necessarily constrained in a similar way: some genes have a larger ability to evolve than other ones due to their lower importance. Such minority genes tell their story instead of the species one, blurring so the phylogenetic

information.

To obtain a phylogenetic tree with high support values, the deletion of these problematic genes (which may result from homoplasy, stochastic errors, undetected paralogy, incomplete lineage sorting, horizontal gene transfers, or even hybridization) is an answer. To do so, we propose to construct the phylogenetic trees that correspond to all the combinations of core genes and to finally consider the tree that is as supported as possible while considering as many genes as possible.

The major drawback of this solution is its prohibitive computational cost, since testing all the possible combinations is totally intractable in practice (2^n phylogenetic tree reconstructions with $n \approx 100$ core genes of plants belonging to the same order). Therefore, we propose to remove the problematic genes without exhaustively testing all the combinations of genes. More precisely, our proposal is to combine various approaches to extract promising subsets of core genes, encompassing systematic deletion of genes, random selection of large subsets, statistical evaluation of gene effects, and genetic algorithms (GAs) [68, 69]. These latters are efficient, robust, and adaptive search techniques designed for solving optimization problems, which have the ability to produce suboptimal solutions [70, 71, 72].

7.2/ PRESENTATION OF THE PROBLEM

Let us consider a set of chloroplast genomes that have been annotated using DOGMA. We have then access to the core genome (genes present everywhere) of these species, whose size is about one hundred genes when the species are close enough. For further information on how we found the core genome, see Chapter 6. Sequences are then further aligned with MUSCLE [28] and the RAxML [46] tool infers the corresponding phylogenetic tree. If this resulting tree is well-supported, then the process is stopped without further investigations. Indeed, if all bootstrap values are larger than 95, then we can reasonably consider that the phylogeny of these species is resolved, as the largest possible number of genes has led to a very well supported tree.

In the case where some branches are not well-supported, we can wonder whether a few genes can be incriminated in this lack of support. If so, we face an optimization problem: *find the most supported tree using the largest subset of core genes*. Obviously, a brute force approach investigating all possible combinations of genes is intractable, as it leads to 2^n phylogenetic tree inferences for a core genome of size n. To solve this optimization problem, we propose a hybrid approach mixing a genetic algorithm with the use of some statistical tests for discovering problematic genes. The initial population for the genetic algorithm is built by both systematic and random pre-GA investigations. These considerations led to the pipeline detailed in Figure 7.1, whose stages will be developed thereafter.

7.3/ GENERATION OF THE INITIAL POPULATION

The objective is to obtain a phylogenetic tree with high-supporting values (applying bootstrap analysis) by using the largest possible subset of genes. If this goal cannot be reached by taking all core genes, the first thing to investigate is to check whether one particular gene is responsible for this problem. Therefore we apply two preliminary stages

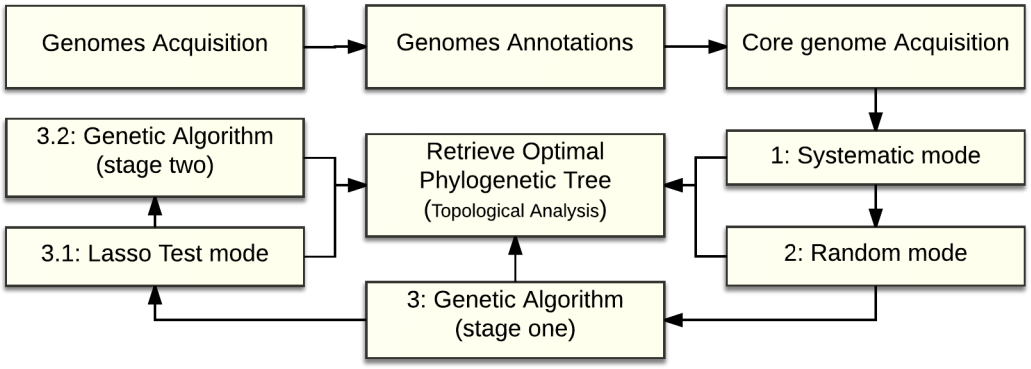
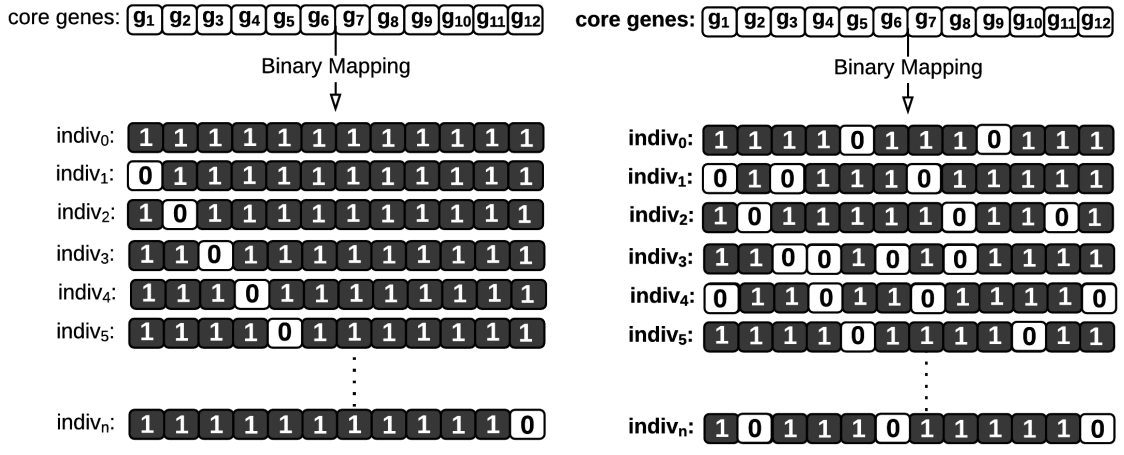


Figure 7.1: Overview of the proposed pipeline for phylogenies based on chloroplasts.

before applying genetic algorithm: the systematic stage and the random one.

The *systematic stage* consists to systematically compute all the trees we can obtain by removing exactly one gene from the core genome, leading to n new phylogenetic trees, where n is the core size (see Figure 7.2(a)). If, during this systematic approach, one well-supported tree is obtained, then it is returned as the phylogeny of the species under consideration. Conversely, if all obtained trees have at least one problematic branch, then deeper investigations are required. However the systematic approach has reached its limits.

Another preliminary stage to GA, called *random stage*, is then launched by investigating two directions:



(a) Systematic mapping. (b) Random mapping.

Figure 7.2: Binary mapping operation overview. (a) Initial individuals obtained in systematic mode stage. Two kinds of individuals are generated. First, by considering all genes in the core genome. Second, by omitting one gene sequentially depending on the core length. (b) Initial individuals are generated randomly in random mode stage by omitting 2-10 genes randomly.

1. Investigate deeply generated phylogenetic trees from systematic stage that have high-supporting values from bootstrap analysis. In other words, when the loss of one specific gene has led to a good tree, we try to improve it again by removing another gene (n-2 possibilities for each specific gene).

2. Generate numerous phylogenetic trees that can be obtained by removing randomly between 2 and 10 genes among the core genome.

In more details, the second direction of the random stage consists of a chosen number of iterations (for instance 200), where for each iteration an integer k between 2 and 10 is randomly picked. This random number defines the number of genes which are then randomly removed, and a phylogeny is inferred using the remaining genes. If during these iterations, by chance, a very well supported tree is obtained, a stop signal is sent to the master process and the obtained tree is returned.

However the number of cases explodes and we can only reasonably hope to investigate a slight proportion of all possibilities: it is illusory to hope to investigate all reachable trees by discarding 10% of a core genome having 100 genes. This explains why the *genetic algorithm* has been proposed. This latter supposes first to have an initial population of subset of core genes, which must be improved step by step.

And we now have enough data to build a good initial population for the genetic algorithm. More precisely, using the $n + 1$ computed trees from the systematic mode to initialize the population of the genetic algorithm results in a population which remains too small and too homogeneous. Indeed, all these trees have been computed in the same way, each inference being produced using 99% of the core genome (in systematic stage, we have removed at most 1 gene in a core genome having approximately 100 genes). Thus, the objective of the random stage is not really to find a well supported tree, but to increase the diversity of the initial population (see Figure 7.2(b)).

Let us now explain the main part of the pipeline, that is, the genetic algorithm.

7.4/ GENETIC ALGORITHM

A genetic algorithm (GA) is a well-known metaheuristic algorithm which has been described by a rich body of literature since its introduction [73, 74]. In the following, we will only discuss the choices we made regarding operators and parameters. For further information and applications regarding the genetic algorithm, see for example [68, 69, 75, 76].

7.4.1/ GENOTYPE AND FITNESS VALUE

Genes of the core genome are supposed to be lexicographically ordered. At each subset s' of the core genome corresponds thus a unique binary word w of length n: for each i lower or equal to n ($i \in \{1, \ldots, n\}$), w_i is 1 if the i-th core gene is in s', else w_i is equal to 0. At each binary word w of length n, we can associate its percentage p of 1's and the lowest bootstrap b of the phylogenetic tree we obtain when considering the subset of genes associated to w. At each word w we can thus associate as fitness value the score $b + p$, which must be as large as possible.

Remark 10: parameters values in scoring function

We currently consider that the lowest bootstrap value b and the percentage of genes p have the same importance in the scoring function. However, changing the weight of each parameter may be interesting in deeper investigations.

7.4.2/ GENETIC PROCESS

Until now, binary words (genotypes) of length n that have been investigated are:

1. the word having only 1's (systematic mode);
2. all words having exactly one 0 (systematic mode);
3. at least 200[1] words having between 2 and 10 0's randomly located (random mode).

To each of these words is attached its score $b + p$. This latter is used to select the 50 best words, or fittest individuals, in order to build the initial population (see the upper part of Figure 7.3). After that, the genetic algorithm loops during 200 iterations or until discovering a word such that its score is larger than 190 (corresponding approximately to a case where at least 95% of core genes are used, which produces a tree whose bootstraps are larger than 95).

During an iteration the algorithm applies the following steps to produce a new population P' given a population P (see Figure 7.4):

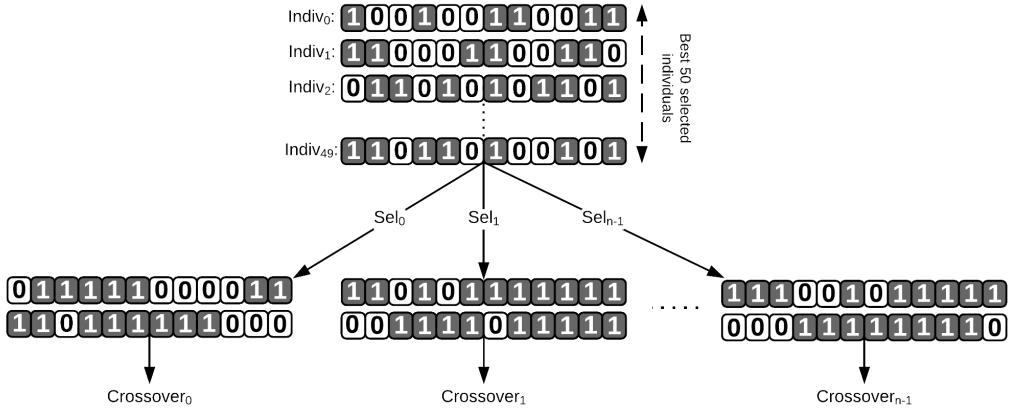


Figure 7.3: Random pair selections from given population.

- Repeat five times a random pickup of a couple of words and mix them using a crossover approach. The obtained words are added to the population P, as described in Section 7.4.3, resulting in population P_c.
- Mutate 5 words of the population P_c, the mutated words being added too to P_c, as detailed in Section 7.4.4, leading to population P_m.

[1] 200 is a parameter that has been specified according to our experiments: it seems to offer the best trade-off between computation time and quality of the initial population.

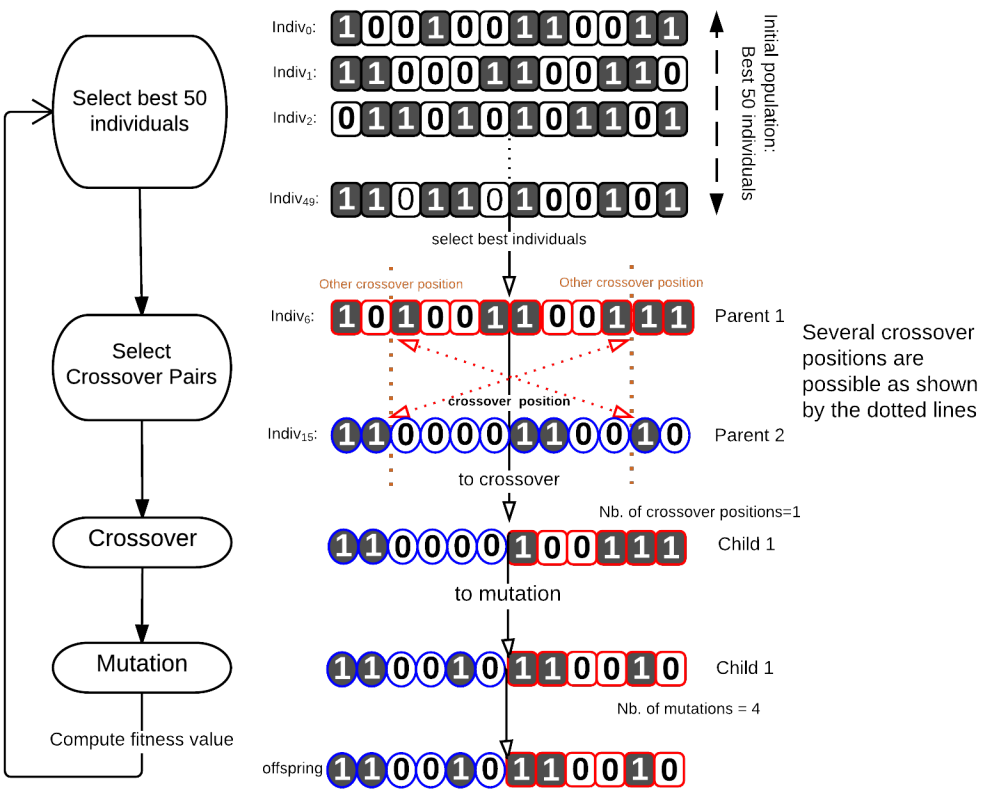


Figure 7.4: Outline of the genetic algorithm.

- Add 5 new random binary words having less than 10% of 0's (see Section 7.4.5) to P_m producing population P_r.
- Select the 50 best words in population P_r to form the new population P'.

Let us now explain with more details each step of this genetic algorithm.

7.4.3/ CROSSOVER STEP

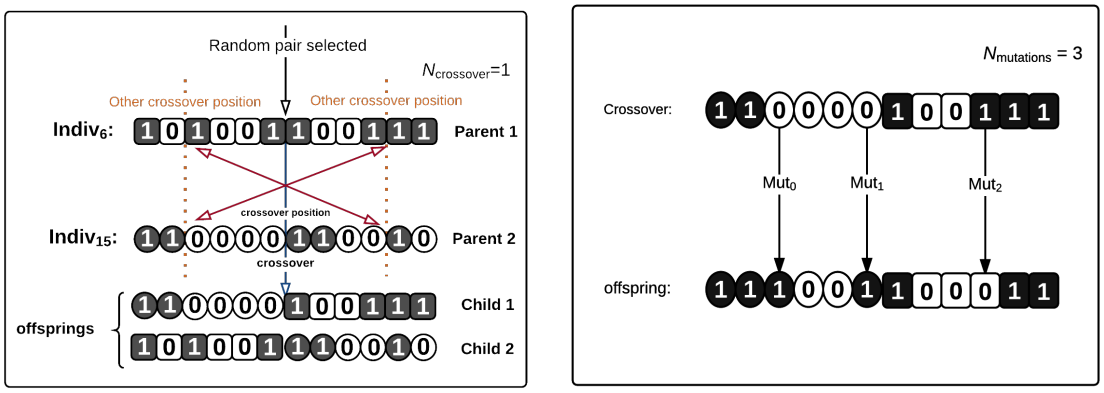
Given two words w^1 and w^2, the idea of the crossover operation is to mix them, hoping by doing so to generate a new word w having a better score (see Figure 7.5(a)). For instance, if we consider a one-point crossover located at the middle of the words, for $i < \frac{n}{2}$, $w_i = w_i^1$, while for $i \geqslant \frac{n}{2}$, $w_i = w_i^2$: in that case, for the first core genes, the choice (to take them or not for phylogenetic construction) in w is the same than in w^1, while the subset of considered genes in w corresponds to the one of w^2 for the last 50% of core genes.

More precisely, at each crossover step, we first pick randomly an integer $N_{crossover} = k$ where $k < \frac{n}{2}$, and randomly again k different integers $i_1, \ldots i_k$ such that $1 < i_1 < i_2 < \ldots < i_k < n$. Then w^1 and w^2 are randomly selected from the population P, and a new word w is computed as follows:

- $w_i = w_i^1$ for $i = 1, ..., i_1$,

- $w_i = w_i^2$ for $i = i_1 + 1, ..., i_2$,
- $w_i = w_i^3$ for $i = i_2 + 1, ..., i_3$,
- etc.

Then the phylogenetic tree based on the subset of core genes labeled by w is computed, the score S of w is deduced, and w is added to the population with the fitness value of S attached to it. Note that, as a parametric option, one word instead of two is generated from this step.



(a) Crossover operation.

(b) Mutation operation.

Figure 7.5: (a) Two individuals were selected from given population. The first portion from determined crossover position in the first individual is switched with the first portion of the second individual. The number of crossover positions is determined by $N_{crossover}$. (b) Random mutations are applied depending on the value of $N_{mutation}$, changing randomly gene state from 1 to 0 or vice versa.

7.4.4/ MUTATION STEP

In this step, we ask how small changes in a given subset of genes (removing and/or adding few genes) may by chance improve the support of the associated tree. Similarly speaking, we try here to improve the score of a given word by replacing a few 0's by 1 and/or a few 1's by 0 as shown in Figure 7.5(b).

In practice, an integer $N_{mutation} = k$ where $k \leqslant \frac{n}{4}$ corresponding to the number of changes, or "mutations", is randomly picked. Then k different integers $i_1, \ldots, i_k$ lower or equal to n are randomly chosen and a word w is randomly extracted from the current population. A new word w' is then constructed as follows: for each $i = 1, ..., n$,

- if i in $\{i_1, \ldots, i_k\}$, then $w'_i = (w_i + 1) \mod 2$ (the gene is mutated),
- else $w'_i = w_i$ (no modification).

Again, the phylogenetic tree corresponding to the subset of core genes associated with w' is computed, and w' is added to the population together with its score.

7.4.5/ RANDOM STEP

In this step, new words having a large amount of 1's are added to the population. Each new word is obtained by starting from the word having n 1s, followed by k random selection of 1s which are changed to 0, where k is an integer randomly chosen between 1 and 10. The new word is added to the population after having computed its score thanks to a phylogenetic tree inference.

7.5/ TARGETING PROBLEMATIC GENES USING STATISTICAL TESTS

7.5.1/ THE LASSO TEST

The Least Absolute Shrinkage and Selection Operator (LASSO) test [77] is an estimator that takes place in the category of least-squares regression analysis. Like all the algorithms in this group, it estimates a linear model which minimizes a residual sum with respect to a variable λ. Let us explain how this variable can be used to order genes with respect to their ability to modify the bootstrap support.

Definition 6: Configuration Matrix

Let X be a $m \times p$ matrix where each line $X_i = (X_{i1}, \ldots, X_{ij}, \ldots X_{ip})$, $1 \leq i \leq m$, is a configuration where X_{ij} is 1 if gene number j is present inside the configuration i and X_{ij} is 0 otherwise. For each X_i, let Y_i be the real positive support value for each problematic bootstrap b per topology and per gene.

According to [77], the Lasso test $\beta = (\beta_1, \ldots, \beta_i, \ldots, \beta_p)$ is defined by

$$\beta = \operatorname{argmin} \left\{ \sum_{i=1}^{m} \left(Y_i - \sum_{j=1}^{p} \beta_j X_{ij} \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}. \tag{7.1}$$

Note that, when λ has high value, beta is null vector. It is thus sufficient to decrease the value of λ to observe that some components β_j of vector beta are no more null. Moreover, the sign of β_j is positive (resp. negative) if the bootstrap support increases (resp. decreases) with respect to j.

After having carried out 200 iterations of the genetic algorithm detailed above, it may occur that no well-supported tree has been produced. Various reasons may explain this failure, like a lazy convergence speed, a large number of problematic genes (*e.g.*, homoplasic ones, or due to stochastic errors, undetected paralogy, incomplete lineage sorting, horizontal gene transfers, or hybridization), or close divergence species leading to very small branch lengths between two internal nodes. However, we now have computed enough word scores to determine the effects of each gene in topologies and bootstraps, which allows to remove the few genes that break supports.

The idea is to investigate each topology that appeared enough times among previous computations. In this study, we only consider topologies having a frequency of occurrence larger than 10%. Remark that this percentage value is convenient for the given case study, but it depends in fact on the number of obtained topologies. Then for each best word of these best topologies, and for each problematic bootstrap in its associated tree, we apply a Lasso test.

7.5.2/ SECOND STAGE OF GENETIC ALGORITHM

Targeting problematic genes using Lasso approach can solve the issue of badly supported values in some cases, especially when only one support is lower than the predefined threshold. In cases where at least two branches are not well supported, removing genes that break the first support may or may not have an effect on the second problematic support. In other words, each of the two problematic supports can be separately solved using Lasso investigations, but not necessarily both together.

However, the population has been improved, receiving very interesting words for each problematic branch. Therefore, a last genetic algorithm phase is launched on the updated population, in order to mix these promising words by crossover operations, hoping by doing so to solve in parallel all of the badly supported values. This last stage runs until either the resolution of all problematic bootstraps or the number of GA iterations reaches a fixed value (set to 1000 in our simulations).

7.6/ CASE STUDIES

7.6.1/ PIPELINE EVALUATION BY VARIOUS GROUPS OF PLANT SPECIES

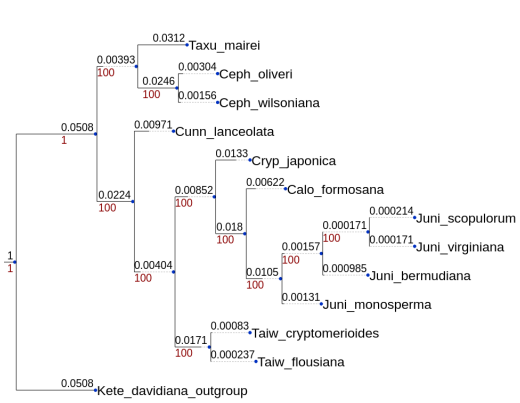
In this section, the proposed pipeline is tested on various sets of close plant species. An example of approximately 50 subgroups (including on average from 12 to 15 chloroplasts species) encompassing 356 plant species are presented in Table 7.1.

In this table, the column *Occ* represents the amount of generated phylogenetic trees from the corresponding searching space for each group. The column c represents the number of core genes included within each group. The *# taxa* column is the amount of species corresponding to the considered group. b is the lowest value from bootstrap analysis. The *Terminus* column contains the termination stage for each subgroup, namely: the systematic (1), random (2), or optimization (3) stage using genetic algorithm and/or Lasso test. Finally, the *Likelihood* column store the likelihood value of the best phylogenetic tree (*i.e.*, according to the lowest bootstrap value b). A large occurrence value in this table means that the associated p-value and/or subgroup has its computation terminated in either penultimate or last pipeline stage. An occurrence of 31 is frequent due to the fact that 32 MPI threads (one master plus 31 slaves) have been launched on our supercomputer facilities. Notice that the groups in Table 7.1 can be divided in four parts:

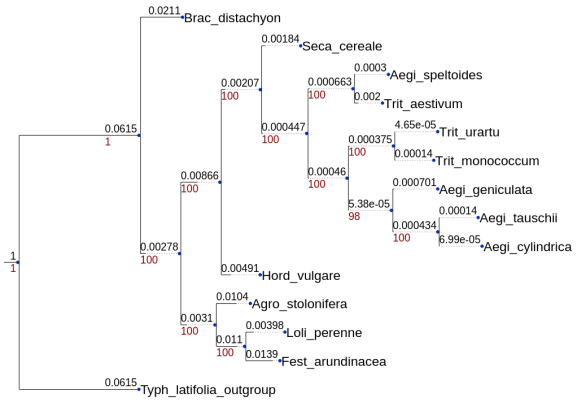
Some phylogenetic trees obtained for different chloroplast groups are shown in Figure 7.6.

Group	Occ.	c	# taxa	b	Terminus	Likelihood	Outgroup
Gossypium_group_0	85	84	12	26	1	-84187.03	*Theo_cacao*
Ericales	674	84	9	67	3	-86819.86	*Dauc_carota*
Eucalyptus_group_1	83	82	12	48	1	-62898.18	*Cory_gummifera*
Caryophyllales	75	74	10	52	1	-145296.95	*Goss_capitis-viridis*
Brassicaceae_group_0	78	77	13	64	1	-101056.76	*Cari_papaya*
Orobanchaceae	26	25	7	69	1	-19365.69	*Olea_maroccana*
Eucalyptus_group_2	87	86	11	71	1	-72840.23	*Stoc_quadrifida*
Malpighiales	422	78	10	96	3	-91014.86	*Mill_pinnata*
Pinaceae_group_0	76	75	6	80	1	-76813.22	*Juni_virginiana*
Pinus	80	79	11	80	1	-69688.94	*Pice_sitchensis*
Bambusoideae	83	81	11	80	3	-60431.89	*Oryz_nivara*
Chlorophyta_group_0	231	24	8	81	3	-22983.83	*Olea_europaea*
Marchantiophyta	65	64	5	82	1	-117881.12	*Pice_abies*
Lamiales_group_0	78	77	8	83	1	-109528.47	*Caps_annuum*
Rosales	81	80	10	88	1	-108449.4	*Glyc_soja*
Eucalyptus_group_0	2254	85	11	90	3	-57607.06	*Allo_ternata*
Prasinophyceae	39	43	4	97	1	-66458.26	*Oltm_viridis*
Asparagales	32	73	11	98	1	-88067.37	*Acor_americanus*
Magnoliidae_group_0	326	79	4	98	3	-85319.31	*Sacc_SP80-3280*
Gossypium_group_1	66	83	11	98	1	-81027.85	*Theo_cacao*
Triticeae	40	80	10	98	1	-72822.71	*Loli_perenne*
Corymbia	90	85	5	98	2	-65712.51	*Euca_salmonophloia*
Moniliformopses	60	59	13	100	1	-187044.23	*Prax_clematidea*
Magnoliophyta_group_0	31	81	7	100	1	-136306.99	*Taxu_mairei*
Liliopsida_group_0	31	73	7	100	1	-119953.04	*Drim_granadensis*
basal_Magnoliophyta	31	83	5	100	1	-117094.87	*Ascl_nivea*
Araucariales	31	89	5	100	1	-112285.58	*Taxu_mairei*
Araceae	31	75	6	100	1	-110245.74	*Arun_gigantea*
Embryophyta_group_0	31	77	4	100	1	-106803.89	*Stau_punctulatum*
Cupressales	87	78	11	100	2	-101871.03	*Podo_totara*
Ranunculales	31	71	5	100	1	-100882.34	*Cruc_wallichii*
Saxifragales	31	84	4	100	1	-100376.12	*Aral_undulata*
Spermatophyta_group_0	31	79	4	100	1	-94718.95	*Mars_crenata*
Proteales	31	85	4	100	1	-92357.77	*Trig_doichangensis*
Poaceae_group_0	31	74	5	100	1	-89665.65	*Typh_latifolia*
Oleaceae	36	82	6	100	1	-84357.82	*Boea_hygrometrica*
Arecaceae	31	79	4	100	1	-81649.52	*Aegi_geniculata*
PACMAD_clade	31	79	9	100	1	-80549.79	*Bamb_emeiensis*
eudicotyledons_group_0	31	73	4	100	1	-80237.7	*Eryc_pusilla*
Poeae	31	80	4	100	1	-78164.34	*Trit_aestivum*
Trebouxiophyceae	31	41	7	100	1	-77826.4	*Ostr_tauri*
Myrtaceae_group_0	31	80	5	100	1	-76080.59	*Oeno_glazioviana*
Onagraceae	31	81	5	100	1	-75131.08	*Euca_cloeziana*
Geraniales	31	33	6	100	1	-73472.77	*Ango_floribunda*
Ehrhartoideae	31	81	5	100	1	-72192.88	*Phyl_henonis*
Picea	31	85	4	100	1	-68947.4	*Pinu_massoniana*
Streptophyta_group_0	31	35	7	100	1	-68373.57	*Oedo_cardiacum*
Gnetidae	31	53	5	100	1	-61403.83	*Cusc_exaltata*
Euglenozoa	29	26	4	100	3	-8889.56	*Lath_sativus*

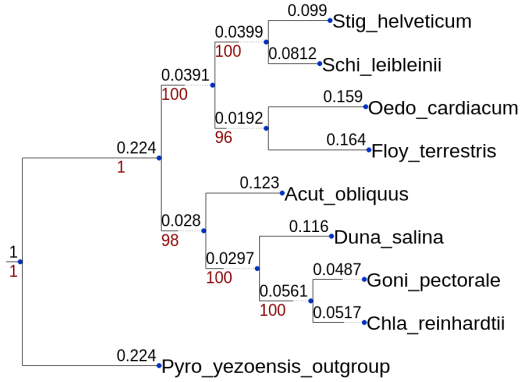
Table 7.1: Results of our pipeline approach on various families.



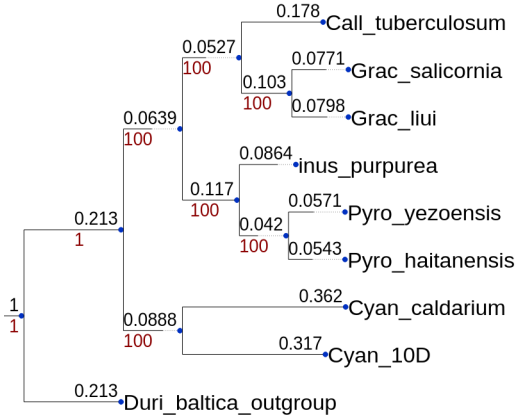
(a) Cupressales group.



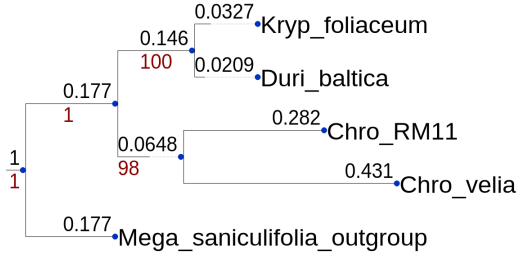
(b) Pooideae group.



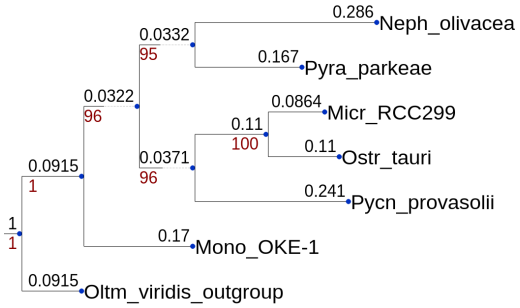
(c) Chlorophyceae group.



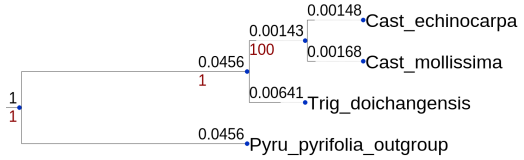
(d) Rhodophyta group.



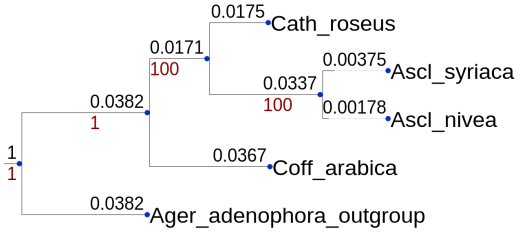
(e) Alveolata group.



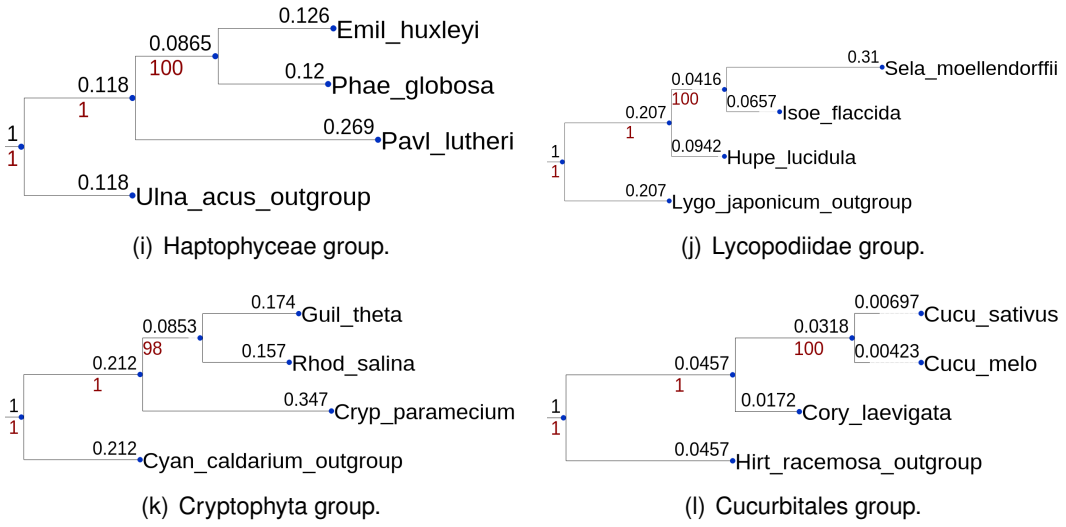
(f) Prasinophytes group.



(g) Fagales group.



(h) Gentianales group.



(i) Haptophyceae group. (j) Lycopodiidae group.

(k) Cryptophyta group. (l) Cucurbitales group.

Figure 7.6: Some phylogenetic trees obtained for different chloroplast groups.

- groups of species stopped in systematic stage with weak bootstrap values (which is due to the fact that an upper time limit has been set for each group and/or subgroups, while each computed tree in these remarkable groups needed a lot of times for computations),
- subgroups terminated during systematic stage with desired bootstrap value,
- groups or subgroups terminated in random stage with desired bootstrap value,
- finally, groups or subgroups terminated with optimization stages.

The majority of subgroups has its phylogeny satisfactorily resolved, as can be seen on all obtained trees which are downloadable at http://meso.univ-fcomte.fr/peg/phylo.

In what follows, an example of one problematic group, namely the *Apiales*, is more deeply investigated as a case study.

7.6.2/ INVESTIGATING *Apiales* ORDER

In our study, *Apiales* choroplasts consist of two sets, as detailed in Table 7.2: two species belong to the *Apiaceae* family set (namely *Daucus carota* and *Anthriscus cerefolium*), while the remaining seven species are in the *Araliaceae* family set. These latter are: *Panax ginseng, Eleutherococcus senticosus, Aralia undulata, Brassaiopsis hainla, Metapanax delavayi, Schefflera delavayi*, and *Kalopanax septemlobus*. Chloroplasts of *Apiales* are characterized by having highly conserved gene content and order [78].

7.6.2.1/ METHOD TO SELECT BEST TOPOLOGIES

We define $T = [t_0, t_1, ..., t_{m'}]$ as a list of $m' = 9053$ obtained trees from multiple execution of given pipeline, starting one time from systematic stage, and multiple times from random stage. By comparing each tree t_i in T with the other trees in T, a set of topologies is then numbered and defined as $W' = \{w'_0, w'_1, w'_2, ..., w'_n\}$, where w'_i is the topology of number

Species	Accession	Genome Id	Size	nb.Genes	Family
Daucus carota	NC_008325.1	114107112	155911 bp	138	Apiaceae
Anthriscus cerefolium	NC_015113.1	323149061	154719 bp	132	Apiaceae
Panax ginseng	NC_006290.1	52220789	156318 bp	132	Araliaceae
Eleutherococcus senticosus	NC_016430.1	359422122	156768 bp	134	Araliaceae
Aralia undulata	NC_022810.1	563940258	156333 bp	135	Araliaceae
Brassaiopsis hainla	NC_022811.1	558602891	156459 bp	134	Araliaceae
Metapanax delavayi	NC_022812.1	558602979	156343 bp	134	Araliaceae
Schefflera delavayi	NC_022813.1	558603067	156341 bp	134	Araliaceae
Kalopanax septemlobus	NC_022814.1	563940364	156413 bp	134	Araliaceae

Table 7.2: Genomes information of Apiales. The number of genes represents the restricted amount of genes.

i. Let $f(x)$ be a function on W' which represents the number of trees having x for their topology. We say that a given topology w'_i is selected as the best topology if and only if $f(w'_i) \geq lb$ where lb is the lower bound threshold computed by the following formula

$$lb = \frac{m' * \gamma}{100}$$

γ is a constant value in $[1, 10]$ and m' is the size of T. Then w'_i is stored as a best topology.

7.6.2.2/ Topological Analysis

In our case, $\gamma = 8$, which means that we exclude as noise the topologies representing less than 8% from the given trees. By doing so, among the 43 topologies which were obtained three of them can be considered as "best topologies" as their number of occurrences $f(x)$ were larger than $lb = 724$, see Table 7.3.

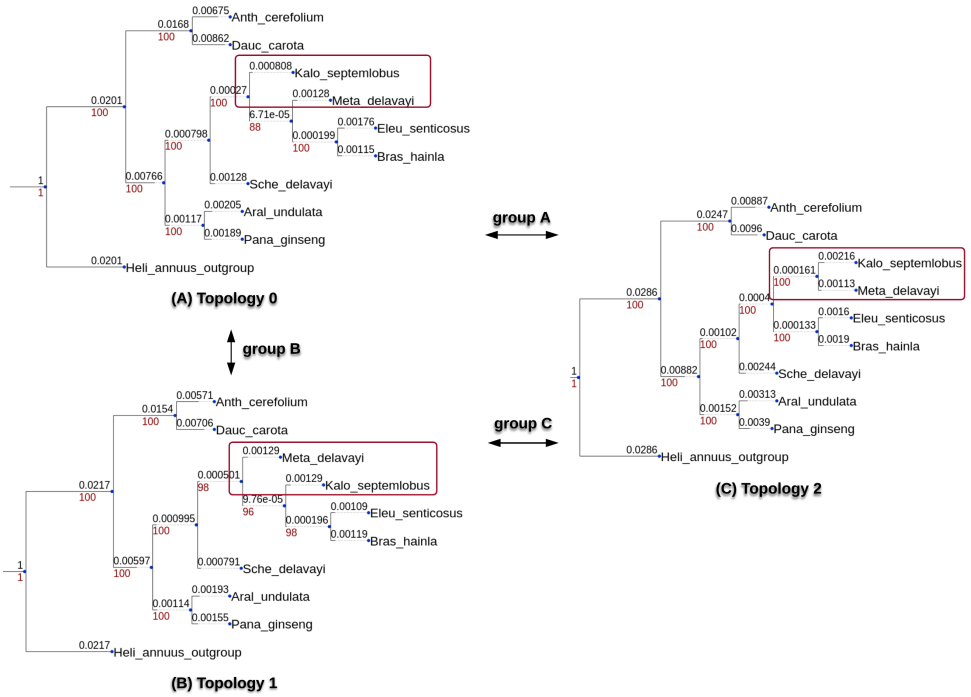


Figure 7.7: Best trees of topologies 0, 1, and 2.

Topo.	b	Av(b)	$p = \frac{(p'*100)}{\lvert c \rvert}$	p'	Av(p')	$\frac{b+p}{2}$	Occ.	LGI	MGI	Best Tree	Status
0	88	56	64.66	75	88.23	76.33	5422	1	97	0111111110110011110110111001011011001010010100110101011101111010001110111111110111111111110111010010111000110001100	Pass
1	96	76	44.83	52	65.02	70.41	2579	2	96	00001010010010000001000010011001011110000011001001110010111000010110010101111000110111001001100011001111101001011100	Pass
2	100	68	99.14	115	72.66	99.57	787	0	98	11111111111111111111111111111111111111111111111111111101111111111111111111111111111111111111111111111111111111111111	Pass
3	72	50	44.83	52	59.24	58.41	89	5	95	01101001010110110101000011010000010000010100111111010000000110110100111100001001001000001111101101011100100010101001	Fail
4	49	29	35.34	41	54.67	42.17	48	6	98	00110101100000100100001000000000110111000101111001010011000000011001010010001011001010010011110010000000010001000100	Fail
5	61	48	25.00	29	50.67	43	31	22	94	00100010000000000000111000001010110010000000100100000110001100010000000001110000000000000101101000000011000100001010000	Fail
6	80	48	34.48	40	43.27	57.24	21	8	100	0110110110011010110110001001001100000100010110001000100111000100001001001000000010010000100010000000011000011001000100	Fail
7	63	53	53.45	62	44.59	58.22	11	31	88	10010000011001101011011100100011000011011001010001001011100101111110111101101110100001101001001100111011111010011110	Fail
8	62	50	68.10	79	48.92	65.05	8	35	92	101111111111101011011100011010001010000111110111111110111101100010111100011111111101100110100110111111011111011010	Fail
9	56	47	43.97	51	32.11	49.98	4	65	93	111101010011010010111100010110010011000001000010110110000110001010001101001100010100000111111000000001000111011001010	Fail
10	46	35	29.31	34	40.95	37.66	4	52	82	010011001010000000000000001100001000001000101100001101000001010010000000011000000000000100101010001000111111011000010	Fail
11	63	55	62.07	72	38.36	62.54	4	44	92	01101101111011111111111110110000111000110111000001010001111001100011000011110111101011101111110111100010100111011011	Fail
12	28	24	45.69	53	38.36	36.85	4	63	85	00101001010111110101100011010101000000001100111111000010000110110100110100001001001000001101101101011100110000101011	Fail
13	35	30	18.97	22	16.95	26.98	3	94	99	000011110100000100100000010000000000000000000001000000110010010000000001001000100000001000000100100000000000010010100000	Fail
14	52	44	44.83	52	35.92	48.41	3	63	96	00011110010011000000010111000011010010100111010100100110010110010010010100011110111000100001001111001111100001001100	Fail
15	46	43	22.41	26	31.61	34.21	3	61	90	00000100100000001100000000000001000101000000000100010001000000000100010001001000111010100000000000010100010010010001010	Fail
16	24	21	29.31	34	21.55	26.66	2	82	100	100010000001010000011101000101101101000000001001000000001110010010000001000000110010010011000010010000000010011001000000	Fail
17	45	41	27.59	32	25	36.29	2	84	90	00100001101000000001001000110000001100000000010001010010010101001010000000000010110110100010100000001000000101010001	Fail
18	47	30	42.24	49	36.21	44.62	2	67	81	001101000000010001001010001000011110101001110010001000010011011010010000101011010100110011000100010111100001011110101	Fail
19	38	31	30.17	35	28.02	34.09	2	81	86	1001000000000000000100000000000111001001010101111100000101000000000100010100011101000100100001010001110100000100110000	Fail
20	28	25	61.21	71	44.83	44.60	2	45	83	110110011011010111100110111101101011110001110001101011111101100110101010111110111011110111111010100000010010110110000	Fail
21	17	17	18.97	22	18.97	17.98	1	94	94	00000000000000000000000010000000000010110000000011000000001000010001001001010011101000000000110000100000000001000000000011	Fail
22	4	4	0.86	1	0.86	2.43	1	115	115	00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010	Fail
23	39	39	17.24	20	17.24	28.12	1	96	96	000000000000000000100000100000000000000010000110101000000001010000010000000000001000111000000000010010000000010010100100	Fail
24	25	25	19.83	23	19.83	22.41	1	93	93	0000000000000100111010010001001000000010000100000100000100010010000100010000101100000000100000000000000100000001000010	Fail
25	4	4	11.21	13	11.21	7.60	1	103	103	0000000000001000000100100001000000000000000000000000000000000000000000000001100001000000100000000110000100100000000000000000000010	Fail
26	13	13	19.83	23	19.83	16.41	1	93	93	00001000000100100011000000010000000000000000000000010100000000100000000010000001001000110100100001000000011101100100000000	Fail
27	34	34	27.59	32	27.59	30.79	1	84	84	010000000000010100001011011000010000110000001011001100010000000000110110010001000000000000100100000000010110000111001	Fail
28	10	10	25.86	30	25.86	17.93	1	86	86	001011001000010000000000000010011000110000001000111000000010011000000010101000010000010100100000000011001001000000000110	Fail
29	3	3	22.41	26	22.41	12.70	1	90	90	010110000000000001000100000000010000010100000000000001000000000010000000101011101011000010110001100010000100010000000000	Fail
30	4	4	16.38	19	16.38	10.19	1	97	97	0000100000001000000010000000100000100100000000000000010000100001100100000000001000000000100011000010100000000000000011000000	Fail
31	14	14	14.66	17	14.66	14.33	1	99	99	0101000100010001100000000000000010000000100000000110000000001001000000000000001000000000000010000000100000000000100000000001	Fail
32	18	18	44.83	52	44.83	31.41	1	64	64	1010110101010011001001100010100100010100011001100100000100001100010010100010111010111011000111111100010011110010000	Fail
33	26	26	31.03	36	31.03	28.52	1	80	80	00010110000100100100000000100100100000000010110010100001000000010010010000011001100110000010000011010101100101100011	Fail
34	33	33	18.10	21	18.1	25.55	1	95	95	00000100110100100001001010000011100000000000010000000001100001000000000000000000010000000000010100100000000010000000010	Fail
35	46	46	39.66	46	39.66	42.83	1	70	70	0001100110010111000000100100100000100101101101101110000110011110100000000110001110000000100100101110110000010001010001	Fail
36	9	9	15.52	18	15.52	12.26	1	98	98	001101000000000001010001000000000100010000000000000000000001000000000001011000000000100010000001000000000000000001100001	Fail
37	35	35	42.24	49	42.24	38.62	1	67	67	01000101100010110010011001111101101011100101000010000100010000110101110000110000101111001111100100001000010010000010	Fail
38	5	5	15.52	18	15.52	10.26	1	98	98	0000000010000100000000000001001000010000000000000100000001000100000000000010000000000000010010000001011000010100010100	Fail
39	38	38	25.86	30	25.86	31.93	1	86	86	100000011000000000011101000000000001011011100010001000000000010101000011000000000010010000010001000011111010000000000000	Fail
40	23	23	16.38	19	16.38	19.69	1	97	97	100000000000000010010100000001000000000000110000000001000000000010000010001000001000000000000000000010011000000110110000	Fail
41	11	11	15.52	18	15.52	13.26	1	98	98	000110000000010100010000000000000010000001000000011100000000000100000001010010100000000000000000000010000000000000000100010	Fail
42	14	14	15.52	18	15.52	14.76	1	98	98	000110000100010000001010000000010100001001000000000100000100000000000000000011100000000000000000000000000000110000000010	Fail

Table 7.3: Information regarding obtained topologies where $\lvert c \rvert = 116$, and $lb = 724$.

In this table, p' is the count of 1's in each word, while p is the percentage of gene contents. $|c|$ is the cardinality of the set of core genes (the size of the core genome), and b is the lowest bootstrap value. Topologies 0 and 1 are delivered from optimization stages when the desired bootstrap value is set to 96, while topology$_2$ is obtained from the systematic stage when we increase the desired bootstrap to 100. Note that *Min.Bootstrap* b in the table is larger than *Avg(b)*, as the former represents the lowest bootstrap value of the best tree in the given topology, while *Avg.Bootstrap (Avg(b))* consists of the average lowest bootstrap in all trees having this topology. LGI is the lowest number of omitted genes and MGI is the maximum number of omitted genes within given topology. The best obtained phylogenetic trees from selected topologies are provided in Figure 7.7.

As it can be noted, only 3 of the 43 obtained topologies contain trees whose lowest bootstrap is larger than 87, namely the topologies number 0, 1, and 2. It is not so easy to make the decision, since all selected trees are very close to each other with small differences. The only notable difference between topologies 0 and 1 is the taxa position of *Kalo_septemlobus* and *Meta_delavayi*. In the same way, there is only one difference between topologies 0 and 1 with 2: grouping the same two taxa of *Kalo_septemlobus* and *Meta_delavayi*. Different comparisons on trees provided with selected topologies are summarized in Figure 7.8.

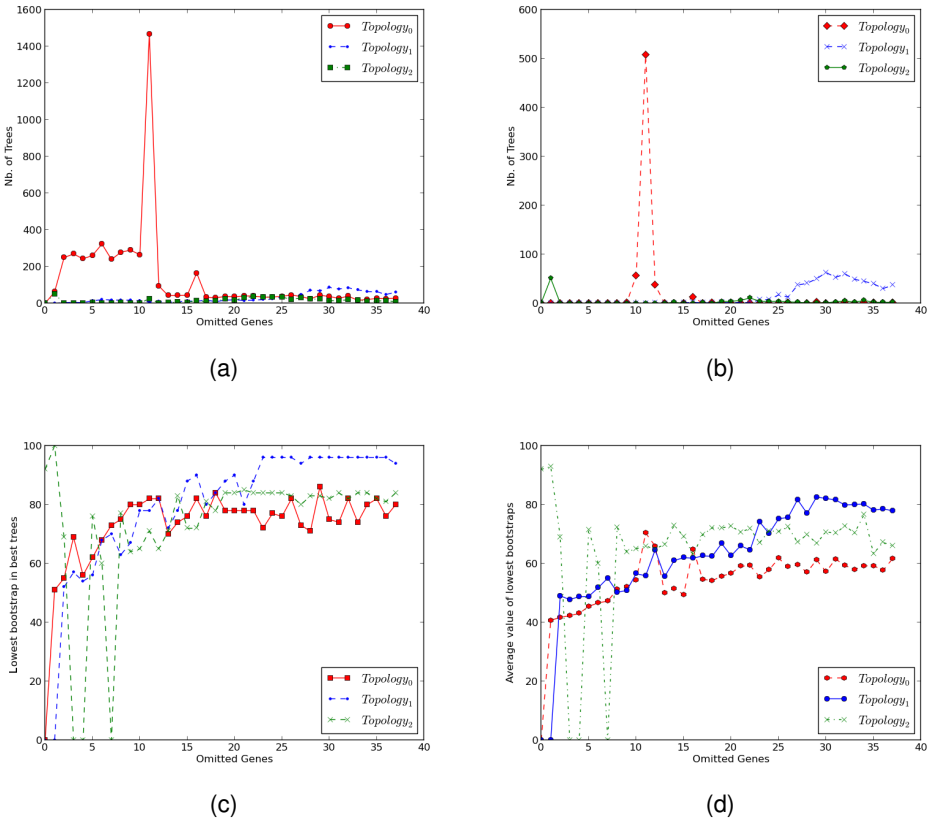


Figure 7.8: Different comparisons of the topologies w.r.t the amount of removed genes: the number of disregarded genes in these figures is specified by $\frac{n}{3}$ where n is the number of core genes. (a) Number of trees per topology, (b) number of trees whose lowest bootstrap is larger than or equal to 80, (c) lowest bootstrap in best trees, and (d) the average of lowest bootstraps.

A new question needs to be answered: which genes are responsible for changing the tree from topology$_0$ to topology$_1$, or to topology$_2$? Deep investigations are still needed in future work to discover the subset of genes in $group_A$, $group_B$, and $group_C$ that change one tree topology to another one (see Figure 7.7).

7.7/ CONCLUSION

In this chapter, five essential pipeline stages have been applied for inferring trustworthy phylogenetic trees from various plant groups. We have verified that inferring a phylogenetic tree based on either the full set or some subsets of common core genes does not always lead to sufficient supports in phylogenetic reconstruction. In both systematic and random stages, many trees have been generated based on omitting some genes. When the desired score is not reachable, a genetic algorithm is then applied inside two specific stages using previously generated trees, to find new optimized solutions after performing crossover and mutation operations. Furthermore, we applied a statistical lasso test for identifying and removing systematically blurring genes, discarding so those which have the worst impact on supports.

We have tested this pipeline on 322 different plant groups, where 63 of them are real families while the remaining ones are random species, these latter playing the role of skeletons when reconstructing the supertree. A case study regarding Apiales order has been analyzed, and three “best” topologies stand out from the 43 obtained ones. In the next chapter, in order to reconstruct the phylogenetic tree for chloroplasts and to apply ancestral investigation on it, we plan to deepen our analysis by investigating another artificial intelligence approach instead of genetic algorithms, namely the particle swarm optimization.

CHAPTER 8

Inferring Phylogenetic Trees using DPSO

In the previous chapter, we shown how to extract the largest subset of core sequences in order to obtain the most supported species tree. Due to computational complexity of such a task, we have proposed a pipeline based on genetic algorithm. We now propose, for the sake of comparison, a distributed Binary Particle Swarm Optimization (DPSO). This work is dedicated to the core genes of *Rosales* order, but it can be applied to any other species. This chapter was accepted and presented in the 12^{th} international conference on Computational Intelligence methods for Bioinformatics and Biostatistics (CIBB 2015, [79]).

8.1/ Discrete Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a stochastic optimization technique developed by Eberhart and Kennedy in 1995 [80]. The most recent update of this study was realized by Kennedy in 2010 [81]. PSOs have been successfully applied in function optimization, artificial neural network training, and fuzzy system control. Basically, each particle follows a very simple behavior which consists in learning from the success of neighboring particles, which are also called individuals. An emergent behavior enables individual swarm members to take benefit from the discoveries or from previous experiences of the other members that have obtained more accurate solutions. PSO is thus a stochastic optimization method that relies on an iterative evolution of a set (the swarm) of candidate solutions in the shape of individuals. Particles move in the solution space and follow the current optimal individual. In the case of the standard binary PSO model [82], the particle position is a vector of N parameters that can be set as “yes” or “no”, “true” or “false”, “include” or “not include”, *etc.* (binary values). A function associates to such kind of vector a real number score according to the optimization problem. The objective is then to define a way to move the particles in the N dimensional binary search space so that they produce the optimal binary vector w.r.t. the scoring function.

In details, each particle i is thus represented by a binary vector X_i (its position). Its length N corresponds to the dimension of the search space, that is, the number of binary parameters to investigate. An 1 in coordinate j in this vector means that the associ-

ated j-th parameter is selected. A swarm of n particles is then a list of n vectors of positions $(X_1, X_2, \ldots, X_n)$ together with their associated velocities $(V_1, V_2, ..., V_n)$, which are N-dimensional vectors of real numbers between 0 and 1. These latter are initially set randomly. At each iteration, the new velocity is computed as follows:

$$V_i(t+1) = u.V_i(t) + \phi_1(P_i^{best} - X_i) + \phi_2(P_g^{best} - X_i) \tag{8.1}$$

where u, ϕ_1, and ϕ_2 are weighted parameters setting the level of each 3 trends for the particle, which are respectively to continue in its adventurous direction, to move in the direction of its own best position P_i^{best}, or to follow the gregarious instinct to the global best known solution P_g^{best}. Both P_i^{best} and P_g^{best} are obtained according to the scoring function.

The new position of the particle is then obtained using the equation below:

$$X_{ij}(t+1) = \begin{cases} 1, & \text{if } r_{ij} \leq \mathit{Sig}(V_{ij}(t+1)), \\ 0, & \text{otherwise,} \end{cases} \tag{8.2}$$

where r_{ij} is a chosen threshold that depends on both the particle i and the parameter j, while the *Sig* function which operates as selection criterion is the sigmoid one in [82], that is:

$$\mathit{Sig}(V_{ij}(t+1)) = \frac{1}{1 + e^{-V_{ij}(t+1)}}. \tag{8.3}$$

8.2/ Application to Phylogeny

Let us consider, for illustration purpose, a set of chloroplast genomes of *Rosales*, which has already been analyzed in the previous chapter using an hybrid genetic algorithm and Lasso test approaches [67]. We sampled 9 ingroup species and 1 outgroup of (*Mollissima*), see Table 8.1 for details, which have been annotated using DOGMA. We can then compute the core genome (genes present everywhere), whose size is equal to 82 genes, by using for instance the methods described in Chapter 6. After having aligned them using MUSCLE, we can infer a phylogenetic tree using RAxML [62], as described in Chapter 4. If all bootstrap values are larger than 95, then we can reasonably consider that the *Rosales* phylogeny is resolved, as the largest possible number of genes has led to a very well supported tree.

In case where some branches are not well supported, we can wonder whether a few genes can be incriminated in this lack of support, for a large variety of reasons already listed in previous chapter, which encompass homoplasy, stochastic errors, undetected paralogy, incomplete lineage sorting, horizontal gene transfers, or even hybridization. As previously stated, trying to find these blurring genes lead to an optimization problem, which is to find the largest subset of core genes that lead to the tree of largest support values. Obviously, a brute force approach investigating all possible combinations of core genes is practically intractable (2^N phylogenetic trees for N core genes, with $N = 82$ for *Rosales*).

As previously, genes of the core genome are supposed to be lexicographically ordered. Each subset s' of the core genome is thus associated with a unique binary word w of length n: for each i, $1 \leq i \leq n$, w_i is 1 if the i-th core gene is in s' and 0 otherwise (see Figure 8.1).

Species	Accession	Seq.length	Family	Genus
Chiloensis	NC_019601	155603 bp	*Rosaceae*	*Fragaria*
Bracteata	NC_018766	129788 bp	*Rosaceae*	*Fragaria*
Vesca	NC_015206	155691 bp	*Rosaceae*	*Fragaria*
Virginiana	NC_019602	155621 bp	*Rosaceae*	*Fragaria*
Kansuensis	NC_023956	157736 bp	*Rosaceae*	*Prunus*
Persica	NC_014697	157790 bp	*Rosaceae*	*Prunus*
Pyrifolia	NC_015996	159922 bp	*Rosaceae*	*Pyrus*
Rupicola	NC_016921	156612 bp	*Rosaceae*	*Pentactina*
Indica	NC_008359	158484 bp	*Moraceae*	*Morus*
Mollissima	NC_014674	160799 bp	*Fagaceae*	*Castanea*

Table 8.1: Genomes information of *Rosales* species under consideration

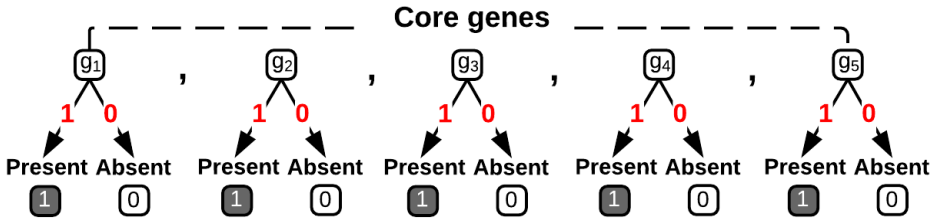


Figure 8.1: Core genes in lexicographical order. Each gene has two possible binary states: either present (1) or absent (0).

Any n-length binary word w can be associated with its percentage p of 1's and the lowest bootstrap b of the phylogenetic tree obtained when considering the subset of genes associated to w. Each word w is thus associated with a fitness score value $\frac{b+p}{2}$ which must be as large as possible. Initial binary individuals are shown in Figure 8.2.

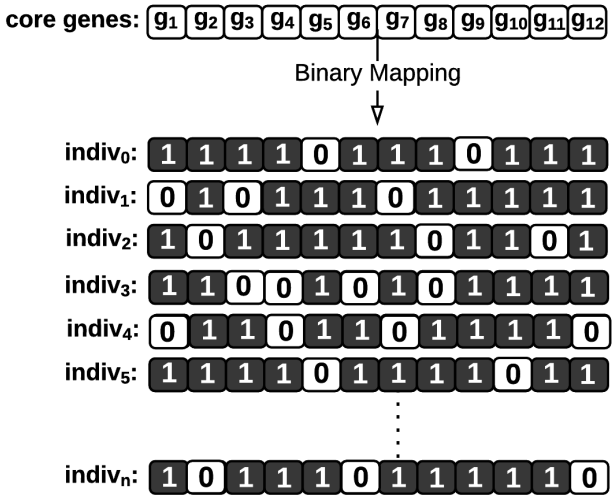


Figure 8.2: Binary words w where the state of each gene in w is randomly selected.

Back in the DPSO context, the search space is then $\{0,1\}^N$. Each node of this N-cube is associated with the set of following data: its subset of core genes s', the deduced phylogenetic tree, its lowest bootstrap b and the percentage p of considered core genes,

and, finally, the score is computed as:

$$S = \frac{b + p}{2} \tag{8.4}$$

Remark 11: Close N-cube nodes

Two close nodes of the N-cube have two close percentages of core genes.

We thus have to construct two phylogenies based on close sequences, leading to a high probability to the same topology with close bootstrap. In other words, the score remains essentially unchanged when moving from a node to one of its neighbors. It allows to find optimal solutions using approaches like PSO.

Algorithm 2: PSO algorithm

$population \leftarrow 10$, $maxiter \leftarrow 10$, $iter \leftarrow 1$
for each particle in $population$ **do**
  $particle[position] \leftarrow [randint(0, 1)$ for each gene in core genome$]$
  $particle[velocity] \leftarrow [rand(0, 1)$ for each gene in core genome$]$
  $particle[score] \leftarrow 0$
  $particle[best] \leftarrow$ Empty list
end for
$fitness \leftarrow 0$, $b \leftarrow 0$, $p \leftarrow$ percentage of gene contents
$u \leftarrow$ calculate initial inertia value from equation (8.5)
while $fitness < S$ and $iter < Maxiter$ **do**
  for each particle in population **do**
    Calculate $new\_fitness$
    if $new\_fitness > fitness$ **then**
      $particle[score] \leftarrow new\_fitness$
      $particle[best] \leftarrow particle[position]$
      $b \leftarrow min($bootstrap of particle[position]$)$
    end if
  end for
  $fitness \leftarrow max(particle[score])$
  $Gbest \leftarrow position[Max(Particle[score]$in population$)]$
  Update the inertia weight u from equation (8.5)
  for each particle in population **do**
    Calculate particle velocity according to Equation (8.1)
    Update particle position according to Equations (8.3) and (8.2)
  end for
  $iter \leftarrow iter + 1$
end while

Initially, the L (set to 10 in our experiments) particles are randomly distributed among all the vertices (binary words) of the N-cube that have a large percentage of 1. The objective is then to move these particles in the cube, hoping that they will converge to an optimal node. At each iteration, the particle velocity is updated according to the fitness and its best position. It is influenced by constant weight factors according to Equation (8.1).

$$u = u_{max} - \frac{u_{max} - u_{min}}{I_{\text{max}}} * I_{\text{cur}} \tag{8.5}$$

In this one, $\phi_1 = c_1.r_1$ and $\phi_2 = c_2.r_2$, where we have set $c_1 = 1$ and $c_2 = 1$. r_1, r_2 are random numbers between 0.1 and 0.5, and u is the inertia weight whose initial value is determined by Equation (8.5), as presented in [83]. In this equation, u_{max} and u_{min} are the boundaries for u, which are set to 0.9 and 0.4 respectively. I_{max} is set to 10, and I_{cur} is equal to *iter* values. This latter determines the contribution rate of a particle's previous velocity to its velocity at the current time step.

To increase the number of included components in a particle, we reduced the interval of Equation (8.1) to [0.1, 0.5]. For instance, if the velocity V_i of an element is equal to 0.511545 and $r = 0.83$, then $Sig(0.51) = 0.62$. So $r > Sig(V_i)$ and this will lead to 0 in the vector elements of the particle. By minimizing the interval we increase the probability of having $r < Sig(V_i)$, and this will lead to more 1s, which means more included elements in the particle. A large inertia weight facilitates a global search while a small inertia weight tends more to a local investigation [84].

Remark 12: Inertia Weight

On the one hand, a larger value of u allows a deep exploration of areas, on the other hand a small one promotes exploitation of areas.

This is why Eberhart and Shi suggested to decrease u over time, typically from 0.9 to 0.4, thereby gradually changing from exploration to exploitation.

Finally, each particle position is updated according to Equation (8.2), see Algorithm 2 for further details. In this algorithm, the particle is defined by its position (a binary word) in the cube together with its velocity (a real vector).

8.3/ EXPERIMENTAL RESULTS AND DISCUSSION

8.3.1/ EXPERIMENTAL PROTOCOL AND RESULTS

We have implemented the proposed DPSO algorithm on the *Mésocentre de calculs* supercomputer facilities of the University of Franche-Comté. Investigated *Rosales* species are listed in Table 8.1. 10 swarms having a variable number of particles have been launched 10 times, with $c_1 = 1, c_2 = 1$, and u linearly decreasing from 0.9 to 0.4.

Swarm	Removed genes	b(%)	p(%)	$(p+b)/2$
1	4	73	95.1	84.05
2	6	76	92.7	84.35
3	20	88	75.6	81.80
4	52	89	36.6	62.8
5	3	72	96.3	84.15
6	19	92	76.8	84.40
7	47	92	42.7	67.35
8	9	74	89	81.50
9	10	73	87.8	80.40
10	13	84	84.1	84.05

Table 8.2: Best tree in each swarm.

The obtained results are summarized in Table 8.2 that contains, for each 10 runs of each 10 swarms, the number of removed genes and the minimum bootstrap of the best tree.

Remark 13: Bootstrap value vs removing genes

Some bootstraps are not so far from the intended ones (larger than 95), whereas the number of removed genes are in average larger than what we desired.

We computed the sum of the number of occurrences in Table 8.3, which is only equal to 715 trees (after deleting frequencies). In this table, we obtained 7 unique topologies after either convergence or $maxIter$ iterations. But, we kept only the ones that have a good minimum bootstrap and a low omitted gene. Only 3 of them have occurred a representative number of time, namely Topologies 0, 2, and 4, which are illustrated in Figure 8.4.

Topology	Swarms	b	p	Occurrences
0	1, 2, 3, 4, 5, 6, 7, 8, 9, 10	92	63	568
1	1, 2, 3, 4, 5, 6, 10	63	45	11
2	1, 2, 3, 4, 5, 6, 7, 8, 9, 10	76	67	55
3	8, 1, 2, 3, 4	56	41	5
4	1, 2, 3, 4, 5, 6, 7, 8, 9, 10	89	30	65
5	1, 3, 4, 5, 6, 9	71	33	9
6	5, 6	25	45	2

Table 8.3: Best topologies obtained from the generated trees. b is the lowest bootstrap of the best tree having this topology, while p is the number of considered genes to obtain this tree.

These three topologies are almost well supported, except in a few branches. According to Figure 8.2, the differences in these topologies are based on the sister relationship of two species named *Fragaria vesca* and *Fragaria bracteata*, and on the relation between *Pentactina rupicola* and *Pyrus pyrifolia*. Due to its larger score and number of occurrences, we tend to select $Topology_0$ as the best representative of the *Rosale* phylogeny.

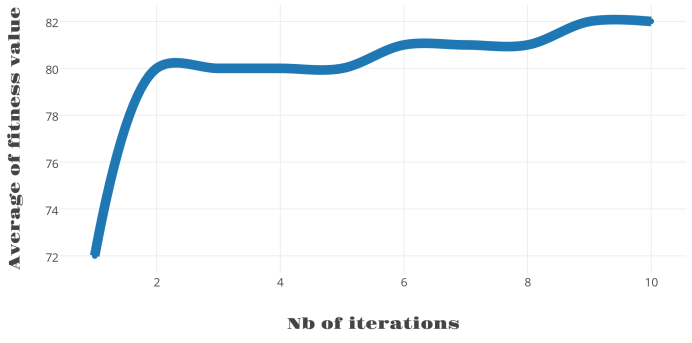
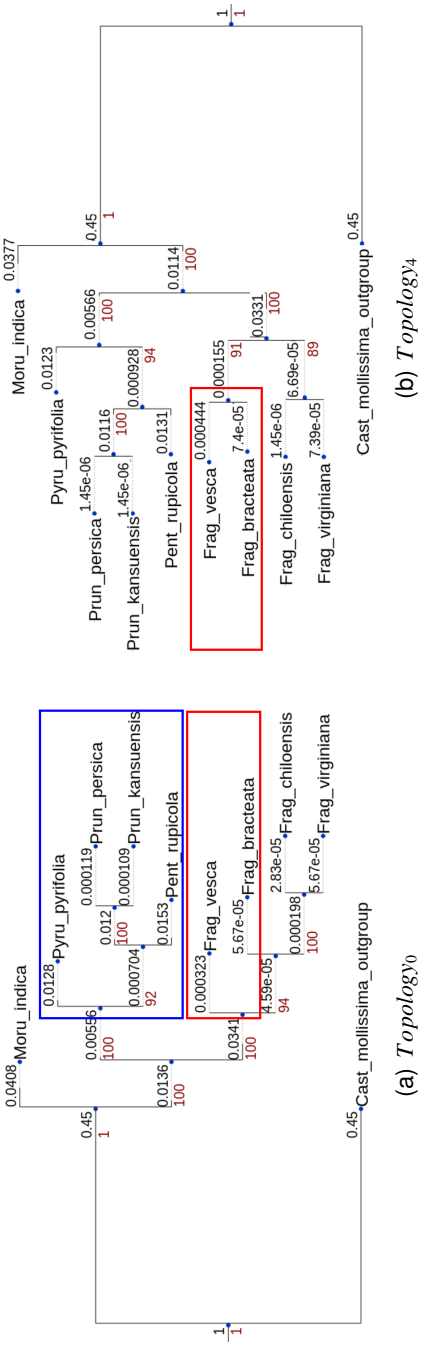


Figure 8.3: Average fitness of *Rosales* order



(a) $Topology_0$ (b) $Topology_4$ (c) $Topology_2$

Figure 8.4: Best obtained topologies for *Rosales* order.

8.3.2/ SELECTING BEST PHYLOGENETIC TREE USING PER-SITE ANALYSIS

To further validate this choice, Consel [85] software has been used on per site likelihoods of each best tree obtained using RAxML [62]. Consel ranks the trees after having computed the p-values of various well-known statistical tests. In Table 8.4, several well known statistical tests such as *bootstrap probability* (BP), *Shimodaira-Hasegawa* (SH), and *Weighted Shimodaira-Hasegawa* (WSH) are used by Consel to give a measure of confidence to a set of candidate trees.

The procedure is simple, it starts by computing the p-value from maximum likelihood (ML) model (*i.e.*, GTR model of RAxML) based on different bootstrap replications, then candidate trees are ranked based on computed minimal ML values. For each given tree, statistical methods are then used to compute the probability value (between 0 and 1) from bootstrap replications and select the tree with greater p-values. In Table 8.4, we can find this latter in the tree provided by $topology_0$, which has larger support values than $topologies_4$ and 2.

Rank	Topo	obs	au	np	bp	pp	kh	sh	wkh	wsh
1	0	-1.4	0.774	0.436	0.433	0.768	0.728	0.89	0.672	0.907
2	4	1.4	0.267	0.255	0.249	0.194	0.272	0.525	0.272	0.439
3	2	3	0.364	0.312	0.317	0.037	0.328	0.389	0.328	0.383

Table 8.4: Consel results regarding best trees

8.4/ MPI: PROPOSED METHODOLOGY

This section presents the strategy deployed to design a parallel version of PSO algorithm.

8.4.1/ THE MASTER-SLAVE PROPOSAL

Traditional PSO algorithms are time consuming in sequential mode. The parallel version shown in Figure 8.5 has thus been proposed to minimize the execution time as much as possible. The general idea of Algorithm 3 is simple: a processor is employed for each particle in order to compute its fitness function, while a last processor called the master centralizes the obtained results. In other words, if we have a swarm of ten particles, we use ten processors as workers and one processor as master (or supervisor).

The master initiates the particles of the swarm, and it distributes the information of the particles to the worker processors. Each worker receives the information of one particle, it computes the fitness function. When one worker finishes its job, it sends a “terminate” signal with the fitness value to the master node. This latter waits that all the workers have finished their jobs. Then, it determines the position of the particle that has the best fitness value as the global best position. This mechanism is repeated until a particle achieves to have a fitness value larger than or equal to 95% with a large set of included genes.

Let us now explain why some calls need to be blocked. In the hierarchical approach of Algorithm 3, a point to point communication has been chosen. There are several types of point to point communication models, but we preferred to work with the standard model to get the most confidential results. In both sending and receiving modes, the buffer is used to cover the message that can be frequently used resources. The problems arise when

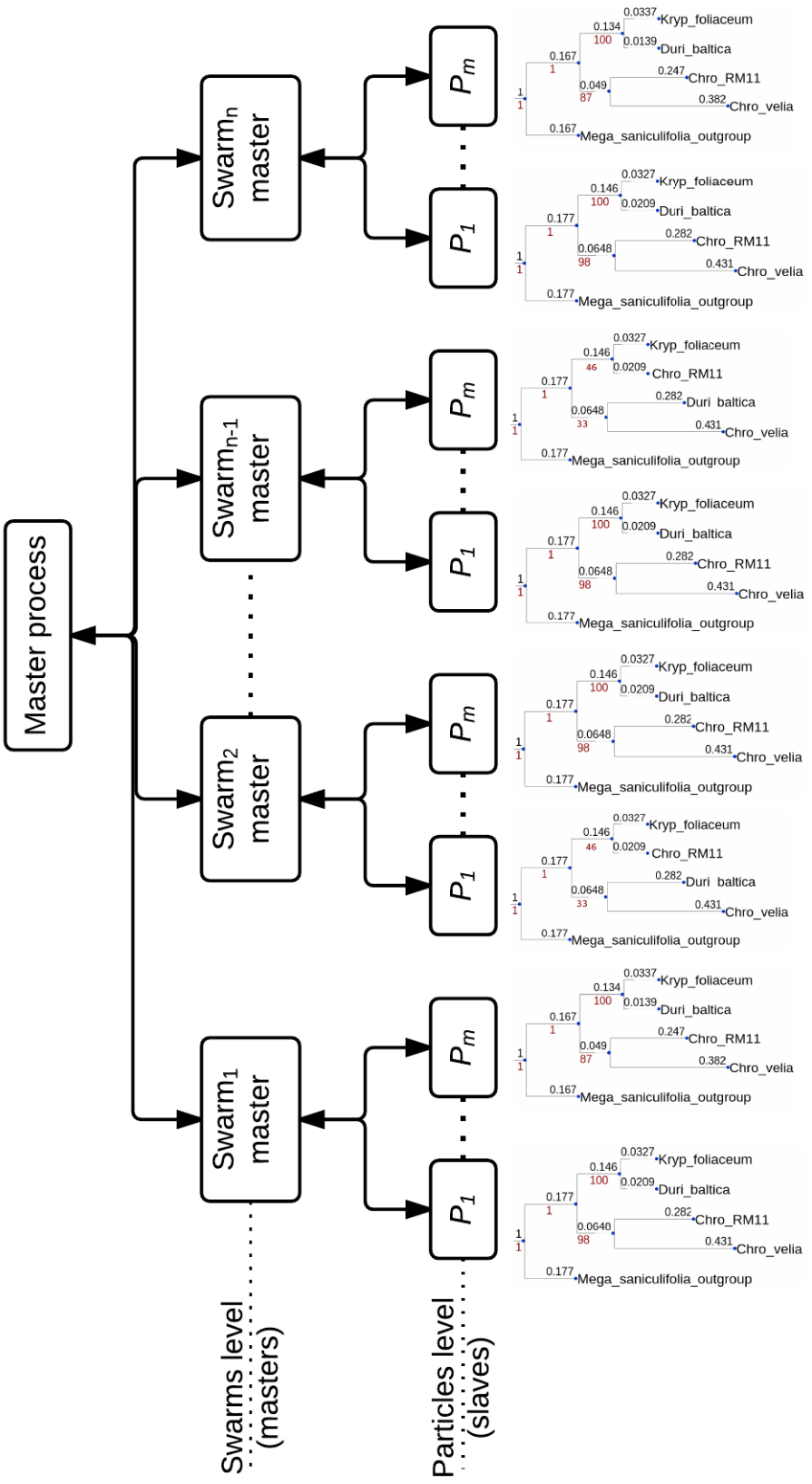


Figure 8.5: The parallel structure of PSO algorithm.

it is used before the completeness of on-going transaction. Blocking communications ensure that this never happens: when control returns from the blocking call, the buffer can safely be modified without any corruption risks of some other part of the process.

Tasks need to be synchronized. Our algorithm requires that cooperating processes must be kept in a more or less strict lockstep which represents computing fitness of each particle. As this step takes a long time, non-blocking calls become less useful. Besides, synchronization is what the blocking calls are intended to provide. The master processor must be blocked until all worker processors finish their computations for determining the best position.

Algorithm 3: Proposed Parallel Algorithm

$rank \leftarrow GetProcessorRank()$
********** Master part **********
$population \leftarrow 10$, $maxiter \leftarrow 10$,$i \leftarrow 0$
$NBworkers \leftarrow GroupSize - 1$
if $rang = 0$ **then**
  for each particle in population **do**
    $particle[position] \leftarrow [randint(0,1)$ for each gene in core genome]
    $particle[velocity] \leftarrow [rand(0,1)$ for each gene in core genome]
    $particle[score] \leftarrow 0$
    $particle[best] \leftarrow position$
  end for
  while $fitness < 95$ and $iter < Maxiter$ **do**
    for each particle in population **do**
      Send (particle, $destination \leftarrow i + 1$) /*i, is the number of worker, sending particle parameters for workers*/
    end for
    for each particle in population **do**
      $particle \leftarrow Receive(source \leftarrow i)$ /*receiving results from worker*/
    end for
    $Fitness \leftarrow Max(Particle[score]inpopulation)$ /*determining the global best according to fitness of particles*/
    $Gbest \leftarrow position[Max(Particle[score]inpopulation)]$/*assigning global best to particle which has best fitness*/
    for each particle **do**
      Calculate particle velocity according to Equation (8.3)
      Update particle position according to Equations (8.1) and (8.2)
    end for
  end while
end if
********** Slave part **********
if $rang > 0$ **then**
  Receive($source \leftarrow 0$, particle) /*receive data from master*/
  Calculate fitness /*according to received parameters*/
  Send(particle, $destination \leftarrow 0$)/*sending results to master*/
end if

8.4.2/ DISTRIBUTED BPSO WITH MPI

Traditional PSO algorithms are time consuming in sequential mode. The distributed version shown in Figure 8.5 has thus been proposed to minimize the execution time as much as possible. The general idea of the proposed algorithm is simple: a processor core is employed for each particle in order to compute its fitness value, while a last core called the master centralizes the obtained results. In other words, if we have a swarm of ten particles, we use ten cores as workers and one core as master (or supervisor).

More precisely, the master initializes the particles of the swarm and distributes them to the workers. When one worker finishes its job, it sends a "terminate" signal with the fitness value to the master. This latter waits until all the workers have finished their jobs. Then, it determines the position of the particle that has the best fitness value as the global best position and sends this information to the workers that update their respective particle velocity and position. This mechanism is repeated until a particle achieves a fitness value larger than or equal to 95 with a large set of included genes. In the following, two distributed versions of the BPSO described previously are considered: in version I the equation used to update the velocity is slightly changed as shown below, and in version II we use the equations of Section 8.1.

8.4.2.1/ DISTRIBUTED BPSO ALGORITHM: VERSION I

In this version Equation (8.1), which is used to update the velocity vector, is replaced by:

$$V_i(t+1) = x \cdot [V_i(t) + C_1(P_i^{best} - X_i) + C_2(P_g^{best} - X_i)] \tag{8.6}$$

where x, C_1, and C_2 are weighted parameters setting the level of each three trends for the particle. The default values of these parameters are $C_1 = c_1 \cdot r_1 = 2.05$, $C_2 = c_2 \cdot r_2 = 2.05$, while x which represents the constriction coefficient is computed according to formula [86, 87]:

$$x = \frac{2 \times k}{|2 - C - (\sqrt{C \times (C-4)})|}, \tag{8.7}$$

where k is a random value between [0,1] and $C = C_1 + C_2$, where $C \geq 4$. According to Clerc [87], using a constriction coefficient results in particle convergence over time.

8.4.2.2/ DISTRIBUTED BPSO ALGORITHM: VERSION II

This version is a distributed approach of the sequential PSO algorithm presented previously in Section 8.1.

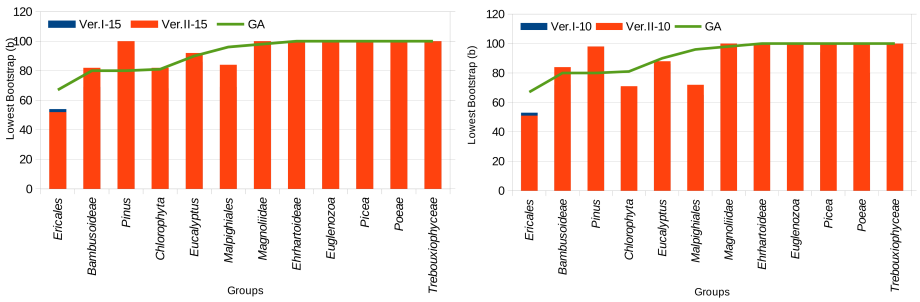
8.4.3/ GENETIC ALGORITHM VS PARTICLE SWARM ALGORITHM

In order to test our method, two versions of PSO have been compared on several plant datasets. We compared too these swarm methods with the GA one presented previously in Chapter 7.

12 groups from the 49 ones contained in Table 7.1 of Chapter 7 have been considered in order to compare the two algorithms. We have tested 5, 10, and 15 particles in the initial population of the swarm approach. As can be seen in Tables 8.5 and 8.6, we do not obtained the same kind of results between 5 particles and 10-15 ones. On the one hand, seven difficult groups[1] are selected whose terminus passed the third stage in GA method. Some groups of light groups[2] are also selected which were passed from the first stage in GA.

From these tables, for difficult groups, we notice that the minimum bootstrap (b) of the best topology obtained of *Chlorophyta*, *Pinus* and *Bambusoideae* is larger than that the one in GA. *Euglenozoa* and *Magnoliidae*, *Eucalyptus*, *Picea*, *Ehrartoideae* and *Trebouxiophyceae* have got the same value of b with GA. But, *Ericales* has got less minimum bootstrap value than in GA, we think this is because the time limitation reserved for each swarm or due to some biological reasons. *Malpighiales* has good b but the number of removed genes is high. For light groups, *Pinus* data set has got minimum bootstrap (b) larger than that in GA. *Picea* and *Trebouxiophyceae* have got the same values of b as in GA.

In Table 8.5 and Table 8.6, *Topo.* is the number of topologies and *NbTrees* is the total number of obtained trees from 10 times of executions (10 swarms). b is the minimum bootstrap value of selected w, $100 - p$ is the number of missing genes in w (note that, p is not in percentage), and *Occ.* is the number of occurrences of the best obtained topology from 10 swarms. More comparison results between GA and both versions of PSOs are provided in Figure 8.6.



(a) PSO with 15 particles vs. GA (b) PSO with 10 particles vs. GA

Figure 8.6: PSO with 10 and 15 particles vs. GA.

According to this figure, we can conclude that the two approaches lead to quite equivalent bootstrap values in most data sets, while on particular subgroups obtained results are complementary. In particular, PSO often produces better bootstraps that GA (see *Magnoliidae* or on *Bambusoideae*), but with a larger number of removed genes. Finally, using 15 particles instead of 10 does not improve so much the obtained results (see Figure 8.6 and Table 8.7).

[1] That is, a set of taxa that finally require deeper investigations, and/or that consume high memory and time.

[2] Light groups: is a set of taxa in which they do not need a lot of time to acquire the high support phylogenetic relations among its taxa.

Group	Topo.	NbTrees	b	$\|c\|$	$100 - p'$	Occ.	Swarms	Particles
Pinus	3	508	98	79	32	462	1,2,3,4,5,6,7,8,9,10	10
Pinus	3	530	94	79	11	129	1,2,3,4,5,6,7,8,9,10	15
Picea	1	100	100	85	42	100	1,2,3,4,5,6,7,8,9,10	10
Picea	1	428	100	85	13	428	1,2,3,4,5,6,7,8,9,10	15
Magnoliidae	3	750	100	79	20	613	1,2,3,4,5,6,7,8,9,10	10
Magnoliidae	3	845	100	79	19	707	1,2,3,4,5,6,7,8,9,10	15
Ericales	30	344	53	84	26	185	1,2,3,4,5,6,7,8,9,10	10
Ericales	34	555	54	84	5	363	1,2,3,4,5,6,7,8,9,10	15
Bambusoideae	8	496	72	94	37	456	1,2,3,4,5,6,7,8,9,10	10
Bambusoideae	11	694	69	94	18	621	1,2,3,4,5,6,7,8,9,10	15
Eucalyptus	16	828	86	83	7	632	1,2,3,4,5,6,7,8,9,10	10
Eucalyptus	20	1073	86	80	4	845	1,2,3,4,5,6,7,8,9,10	15
Malpighiales	34	327	65	78	35	233	1,2,3,4,5,6,7,8,9,10	10
Malpighiales	38	483	69	78	40	326	1,2,3,4,5,6,7,8,9,10	15
Chlorophyta	25	191	70	24	11	109	1,2,3,4,5,6,7,8,9,10	10
Chlorophyta	29	94	68	24	11	1	1,2,3,4,5,6,7,8,9,10	15
Euglenozoa	3	450	100	26	7	292	1,2,3,4,5,6,7,8,9,10	10
Euglenozoa	3	520	100	26	4	491	1,2,3,4,5,6,7,8,9,10	15
Ehrhartoideae	2	23	100	81	0	23	1,2,3,4,5,6,7,8,9,10	10
Ehrhartoideae	3	455	100	81	0	451	1,2,3,4,5,6,7,8,9,10	15
Trebouxiophyceae	3	409	100	41	2	405	1,2,3,4,5,6,7,8,9,10	10
Trebouxiophyceae	3	415	100	41	8	354	1,2,3,4,5,6,7,8,9,10	15
Poeae	1	971	100	80	9	971	1,2,3,4,5,6,7,8,9,10	10
Poeae	1	1399	100	80	20	1399	1,2,3,4,5,6,7,8,9,10	15

Table 8.5: Families applied on DPSO Version1

Group	Topo.	NbTrees	b	$\|c\|$	$100 - p'$	Occ.	Swarms	Particles
Pinus	3	615	98	79	14	275	1,2,3,4,5,6,7,8,9,10	10
Pinus	3	628	100	79	12	558	1,2,3,4,5,6,7,8,9,10	15
Picea	1	635	100	85	14	635	1,2,3,4,5,6,7,8,9,10	10
Picea	1	821	100	85	15	821	1,2,3,4,5,6,7,8,9,10	15
Magnoliidae	3	494	100	79	16	73	1,2,3,4,5,6,7,8,9,10	10
Magnoliidae	3	535	100	79	42	384	1,2,3,4,5,6,7,8,9,10	10
Bambusoideae	6	952	84	81	23	94	1,2,3,4,5,6,7,8,9,10	10
Bambusoideae	9	1450	82	81	18	113	1,2,3,4,5,6,7,8,9,10	15
Eucalyptus	17	972	88	80	18	618	1,2,3,4,5,6,7,8,9,10	10
Eucalyptus	23	1439	92	80	10	843	1,2,3,4,5,6,7,8,9,10	15
Chlorophyta	25	529	71	24	6	397	1,2,3,4,5,6,7,8,9,10	10
Chlorophyta	46	1500	82	24	11	397	1,2,3,4,5,6,7,8,9,10	10
Ericales	30	97	51	84	11	56	1,2,3,4,5,6,7,8,9,10	10
Ericales	34	1257	52	84	7	800	1,2,3,4,5,6,7,8,9,10	15
Malpighiales	35	725	72	79	25	445	1,2,3,4,5,6,7,8,9,10	10
Malpighiales	86	1464	84	79	45	359	1,2,3,4,5,6,7,8,9,10	15
Euglenozoa	3	197	100	26	1	165	1,2,3,4,5,6,7,8,9,10	10
Euglenozoa	3	450	100	26	10	393	1,2,3,4,5,6,7,8,9,10	15
Ehrhartoideae	1	24	100	81	10	24	1,2,3,4,5,6,7,8,9,10	10
Ehrhartoideae	1	20	100	81	9	20	1,2,3,4,5,6,7,8,9,10	15
Trebouxiophyceae	3	319	100	41	1	313	1,2,3,4,5,6,7,8,9,10	10
Trebouxiophyceae	3	818	100	41	2	81	1,2,3,4,5,6,7,8,9,10	15
Poeae	1	991	100	80	22	991	1,2,3,4,5,6,7,8,9,10	15
Poeae	1	1490	100	80	26	1490	1,2,3,4,5,6,7,8,9,10	15

Table 8.6: Groups applied on DPSO Version2

	PSO Ver.I		PSO Ver.II		
Group	**10**	**15**	**10**	**15**	**GA**
Ericales	53	54	51	52	67
Bambusoideae	72	69	84	82	80
Pinus	98	94	98	100	80
Chlorophyta	70	68	71	82	81
Eucalyptus	86	86	88	92	90
Malpighiales	65	69	72	84	96
Magnoliidae	100	100	100	100	98
Ehrhartoideae	100	100	100	100	100
Euglenozoa	100	100	100	100	100
Picea	94	100	100	100	100
Poeae	80	80	100	100	100
Trebouxiophyceae	100	100	100	100	100

Table 8.7: PSO vs GA.

8.5/ CONCLUSION

In this chapter, a discrete particle swarm optimization algorithm has been proposed, which focuses on the problem of finding the largest subset of core sequences having the most supported phylogenetic tree. This heuristic approach has then been applied to the *Rosales* order of 82 core genes. Like in the previous chapter, the scoring function is based on two parameters: the lowest bootstrap value b and the percentage of gene p. These two parameters have the same importance in the scoring function, any modification on any one leading us to deeply investigations.

A per site analysis by Consel is applied after the phylogenetic discovery stage, where a special topological process analysis all trees generated from swarms and classifying them based on its topology. Few topologies of high scores are then selected from from all available topologies using a threshold of lower bound formula. If there is only one nominated tree, then its done, else we use per site analysis to choose the relevant one.

Two parallel versions of discrete particle swarm optimization algorithm was developed in order to reduce the time and memory. 12 groups of plant genomes are applied on two swarm versions. In one hand, we used a swarm of 10 particles with the two versions of algorithms, while in other hand, we employ a swarm of 15 particles with the two algorithm versions. We used the Mésocentre de calcul facilities for the computation of all versions. Various results of hard and light groups are obtained and compared with genetic algorithm one.

CHAPTER 9

Ancestral Reconstruction

Ancestral genome reconstruction has already been investigated several times in the literature [88, 89], but usually it deals with permutations of integers. In other words, tools like Badger [90] or MLGO [91] do not support genomes of various length and with repeated/missing genes. Our problem applied to chloroplasts may appear as more difficult, as we relax the permutation hypothesis. However, in the classical Multiple Genome Rearrangement Problem [92], targeted genomes are bacterial or nucleus ones, which have much more genes than a chloroplast. Furthermore, gene order and content do not evolve so much when considering related plant species. Such observations explain why state-of-the-art algorithms cannot be applied to our particular problem even if this latter should be solvable. In this chapter, we applied a pipeline of two suggested methods to compute the ancestral genomes in all presented internal nodes using a well-supported phylogenetic tree of *Apiales* species.

9.1/ General Presentation of the Problem

Given a set of n genomes and a well-supported phylogenetic tree T, the problem consists in finding the genomes at each of the internal nodes, as described in Figure 9.2. Doing so will provide the evolution of genomes from the root until the leaves.

Any rooted phylogenetic tree as shown in Figure 9.1 is composed by the subtrees provided in Figure 9.2. In the α-tree illustrated in Figure 9.2(a), only one ancestor genome reconstruction is required. The two other main subtrees, shown in Figures 9.2(b) and 9.2(c), have one or two additional taxa compared with the α-tree (they are indeed aggregations of α-trees). More precisely, at each time a new taxon is added to the tree, a new internal (ancestor) node is created in the tree, and its ancestor is then computed. The reconstruction operation must be as parsimonious as possible, according to the recombination operations already listed in this manuscript. At this point, we need to keep in mind the following remark.

Remark 14: Global optimum over the tree

The global optimum over the tree may be obtained with a few local solutions (one ancestor of two genomes) that are not optimal.

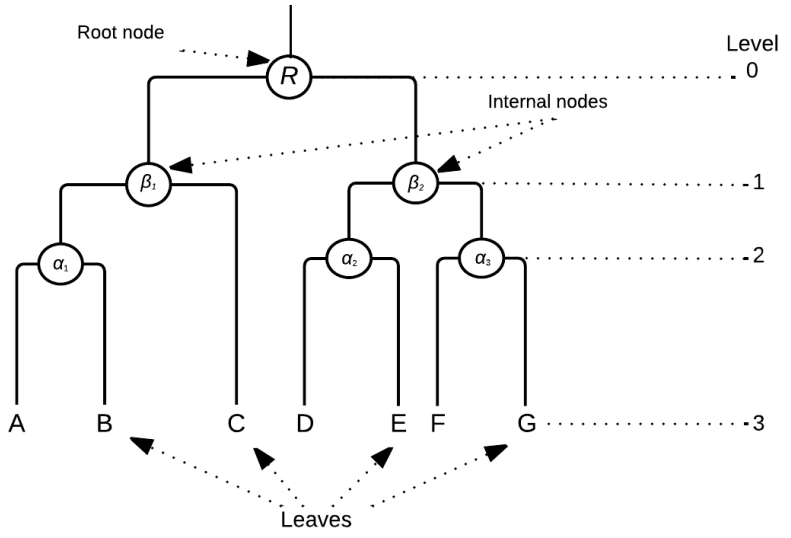
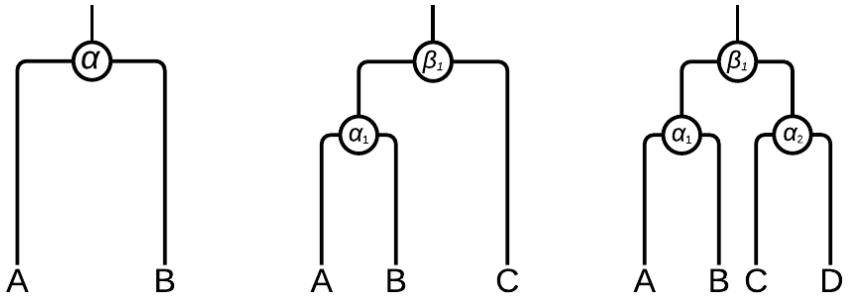


Figure 9.1: The general overview of rooted phylogenetic tree with internal and root nodes.



(a) α-tree.

(b) Composite tree of one α-tree.

(c) Composite tree of two α-trees.

Figure 9.2: The subtrees forms located in any rooted phylogenetic tree.

Given ordered lists of genes at their leaves, the ancestral states in each tree of Figure 9.2 are determined according to the minimum number of edition operations (measured by an edit distance d) required to obtain the leaves starting from the ancestors, as described thereafter:

- In α-tree: the ancestor node α can be determined according to the following formula:

$$\alpha(A,B) = \begin{cases} A \text{ or } B & \text{if } d(A,B) = 0, \\ ? & \text{else.} \end{cases} \tag{9.1}$$

In this tree, α-node is the ancestor of leaves A and B. In gaps-free sequences, if the distance $d \in [0,1]$ between gene lists A and B is not zero, we have at least one recombination operation (insertion, deletion, or replacement) between these two sequences. In this case, we cannot determine which situation to consider in the α-ancestor genome. So, shared genes in the two given sets will appear in the ancestral genome plus question marks in problematic positions.

- Composite one-α-tree: the ancestor nodes α and β, as shown in Figure 9.2(b), can be determined according to the following formula:

$$\beta_1(\alpha_1, C) = \alpha(\alpha_1(A, B), C)$$

where α_1-ancestor can be deduced from the following formula:

$$\alpha_1(A,B)\begin{cases} A \text{ or } B & \text{if } d(A,B)=0, \\ A & \text{if } d(A,C)=0, \text{ and } d(B,C)\neq 0, \\ B & \text{if } d(B,C)=0, \text{ and } d(A,C)\neq 0, \\ ? & \text{else.} \end{cases} \tag{9.2}$$

where C in left or right α-tree is considered as a reference. More precisely, the β-tree is composed of two α-trees: the inner one represented by $\alpha_1 = (A \cap B)$ and the outer one represented by $\beta = (\alpha_1 \cap C)$. We now explain how to deduce the ancestral genome of α_1 node in the inner α-tree. Consider for instance the case where one gene$_i$ in A does not match with its correspondent in B. In this case, we consider the outer branch C as a reference (cousin) to take a decision, by observing the i-th gene in C. If this latter matches with gene$_i$ in A, then this gene will be put inside the ancestor α_1.

A particular case can occur, when the gene in position C_i matches neither with gene A_i nor with B_i. In this case, if possible, we need to investigate another outer (but close) genome X, by computing the distances to the inner α-tree, and selecting at each time the minimum one based on the following formula:

$$\beta_1(\alpha_1, C) = \alpha_1(A, B) \cap X'$$

thus, X is computed as follows:

$$X = \min[\min(d(A, O_1), d(B, O_1)), \ldots, \min(d(A, O_m), d(B, O_m))]$$

where $O_1, O_2, \ldots, O_m$ are the set of m outer (but close) branches to current α-tree, and X' is the name of the minimum distant branch to A and B. If we cannot deduce the character state of position i, we simply put "?" in the ancestor. $\alpha_1(A, B)$ is then deduced as follows:

$$\alpha_1(A,B)\begin{cases} A \text{ or } B & \text{if } d(A,B)=0, \\ A & \text{if } A=X', \text{ and } d(A,B)\neq 0, \\ B & \text{if } B=X', \text{ and } d(A,B)\neq 0, \\ ? & else. \end{cases} \tag{9.3}$$

- Composite tree with two-α-trees: two α-trees are included in this type of trees, leading to a hard computational problem due to the number of distance calculations. To determine the ancestor genome β_1, we need first to deduce the ancestors α_1 and α_2 from taxonomy units ($A, B, C,$ and D) by the following formula:

$$\beta(\alpha_1, \alpha_2) \begin{cases} \alpha_1 \cap \alpha_2 & \text{if } d(\alpha_1, \alpha_2) = 0, \\ \alpha_{1i} \cap \alpha_{2i} \text{ or } ?_i & \text{if } d(\alpha_1, \alpha_2) \neq 0. \end{cases} \tag{9.4}$$

  where α_1 and α_2 are computed using Equation 9.3. To calculate α_1-genome, we consider that X' has the minimum genome distance from $\alpha_2(C, D)$, and vice-versa.

Taking into consideration all subtree cases, we can conclude the ancestor genomes in most phylogenetic trees. Note that inversions, sometimes located among taxonomy units, are excluded from this first approach: they must be considered, but in more sophisticated ancestor reconstruction algorithms.

9.2/ Ancestral Reconstruction Pipeline

In this section, our new pipeline for ancestral reconstruction problem is explained with more details, in the particular case of chloroplast genomes, see Figure 9.3. It is fundamentally based on the gestalt pattern matching algorithm [93], via the use of naked eye investigation and matching results with the `SequenceMatcher` of the Python `difflib` module. In this pipeline:

1. The first stage consists of handling the input data: the phylogenetic tree and the ordered lists of genes. We then need to decide whether duplicated genes must be considered or not. In case where they are under consideration, this first stage provides too some information regarding them.
2. The second step deals with genome comparisons of sister species, from leaves until the root. Desired operations are performed such as gene matching, deletion, or insertion within one of both genomes. The obtaining results are essential in order to build ancestor genomes.
3. The final stage provides statistical information and all ancestral genomes, for each internal node until reaching the root.

9.2.1/ Data Preparation

Let us consider a set of complete chloroplastic genomes for close plant species, like the *Apiales* order as shown in Table 9.1.

We assume first that:

1. Each genome has been annotated with Dogma [54], already presented in this manuscript. By doing so, the same gene prediction and naming process has been applied to the same average quality of annotation. In particular, when a gene appears twice in the considered set of genomes, it receives twice the same name

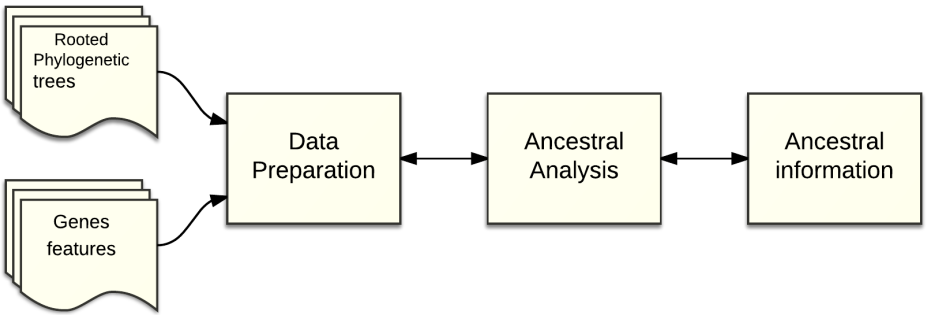


Figure 9.3: General overview of the proposed pipeline. In this pipeline, you can see some arrows are in both sides because we need to prepare the chosen new cousin, or to acquire new information.

Organism name	Sequence length	nb. genes	dup.genes	Lineage
Daucus carota	155911 bp	166	31	Apiaceae
Anthriscus cerefolium	154719 bp	166	32	Apiaceae
Panax ginseng	156318 bp	169	31	Araliaceae
Eleutherococcus senticosus	156768 bp	169	31	Araliaceae
Aralia undulata	156333 bp	169	31	Araliaceae
Brassaiopsis hainla	156459 bp	168	31	Araliaceae
Metapanax delavayi	156343 bp	169	31	Araliaceae
Schefflera delavayi	156341 bp	170	31	Araliaceae
Kalopanax septemlobus	156413 bp	169	31	Araliaceae

Table 9.1: Genomes information of *Apiales*

(no spelling error). At this level, each genome is described by an ordered list of gene names, with possible duplication. Other approaches are possible, see, *e.g.*, [48, 47, 49].

2. The sequences in the core genome (genes present everywhere in the considered set of species) have been multi-aligned using MUSCLE, and a well-supported phylogenetic tree has been obtained based on this alignment as shown in Figure 9.4 for *Apiales* order. This stage may necessitate the deletion of a few core genes that possibly blur the phylogenetic signal (for various reasons encompassing homoplasy, incomplete lineage sorting, horizontal gene transfers, *etc.*), for instance by using methods detailed in [79, 67, 94].

For all three steps of reconstruction, a set of authorized operations are provided, which are:

- Insertion, deletion, duplication, or inversion of one or a block of genes, at gene lists level.
- Operations commonly considered in the Needleman-Wunsch edit distance [95] (insertion, modification, or deletion of a nucleotide, together with opening and enlarging a gap), at DNA sequence levels.

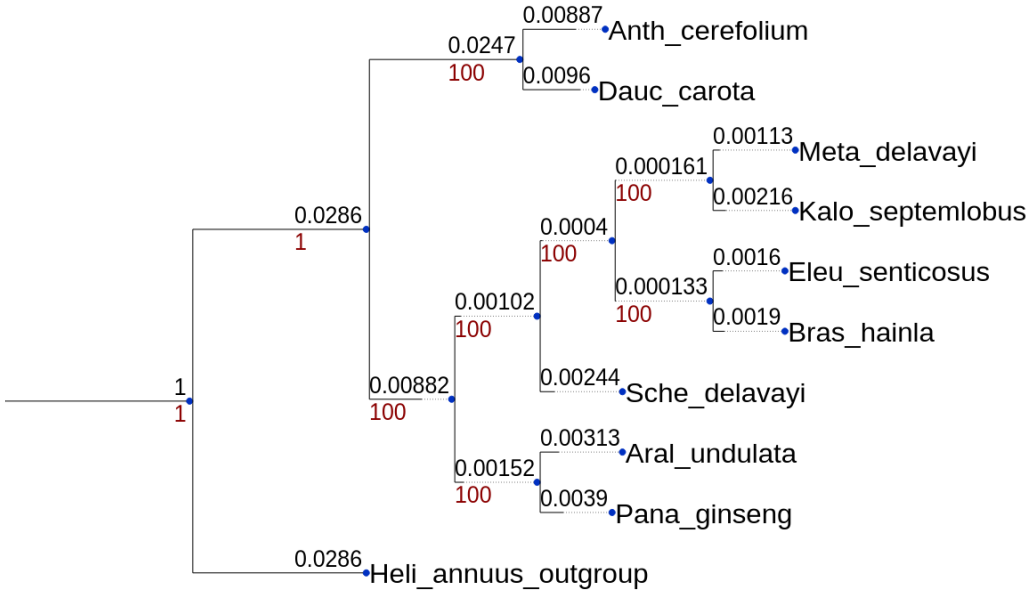


Figure 9.4: High supported phylogenetic tree of *Apiales* order.

In what follows, two general algorithms for ancestral reconstruction of chloroplastic genomes are proposed. In the first one we do not tackle with gene duplication, while they are considered in the second algorithm. In both cases, inversions are not considered too.

9.2.2/ Ancestral Analysis Methods

We now present the two methods we have used in order to predict the set of ancestral genomes, provided a set of chloroplast genomes and a well-supported phylogenetic tree. The first one is manual, while the second one is an algorithm. Note that they only represent our first basic approaches in the problem of ancestral reconstruction of chloroplasts.

More precisely, the first method is based on finding the ancestor genes (*e.g.*, core genes) using naked eye investigation between close genomes, and to identify the set of rearrangement operations (gene duplication is considered). In the second method, we start by removing gene duplication based on a renaming duplicated gene names process. We use sequence comparisons with reference genome as a preliminary step, then we follow the same remaining stages than in the first method.

9.2.2.1/ Ancestor Prediction based on Gene Contents

This method encompasses the following general steps:

- **Step 1: Preliminary stage:** In this step, all internal nodes from leaf nodes to the root are named following an alphabetical order. Each letter in internal node represents an ancestor genome. This latter can be the ancestor of two leaves, an internal ancestor and a leaf node, or two internal ancestors. The result of this step is shown in Figure 9.5.

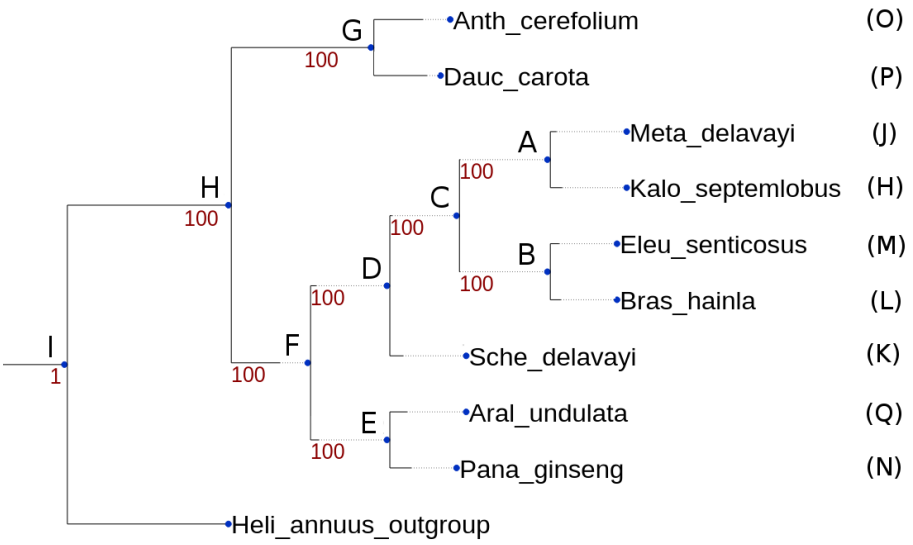
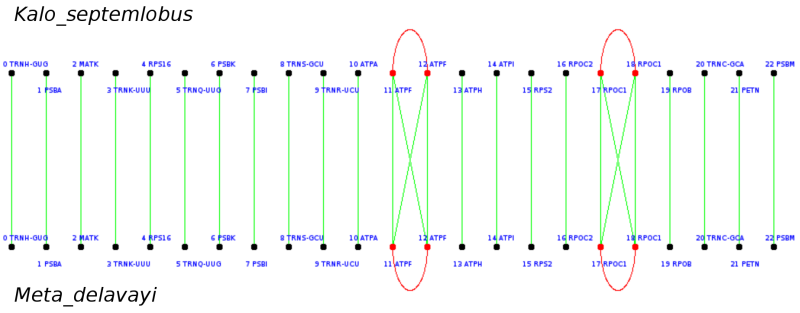
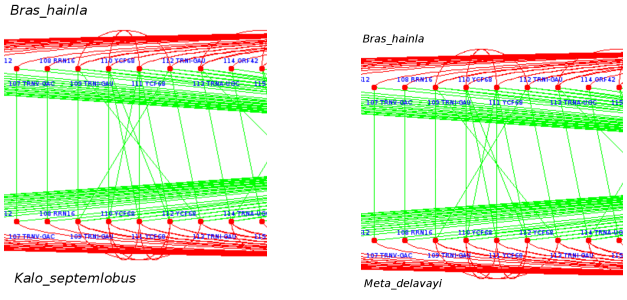


Figure 9.5: Phylogenetic tree of *Apiales* order.

In this tree, we applied a bottom-up procedure for predicting the ancestor genome at each internal node. More precisely, starting from leaves nodes, the given tree can be divided according to the subtrees presented in Figure 9.2 where the node C could be interpreted as a composite tree of two α-trees A and B. However, predicting the ancestor from bottom to top levels will lead to the last common ancestor (I) at the root level.



(a)



(b) (c)

Figure 9.6: Graphical presentation of genes alignment between two genomes.

- **Step 2: Genome Selection:** Figure 9.5 presents the best topology for *Apiales* order obtained in a previous chapter. We then start by selecting two close genomes at leaf level (*e.g.*, *Meta_Delavayi* (J) and *kalo_septemlobus* (H) for ancestor genome (A), and *Eleu_senticosus* (M) and *Bras_hainla* (L) for ancestor genome (B)) as organized in the phylogenetic tree. Selected genomes are aligned graphically as shown in Figure 9.6. We then identify, by using naked eyes investigation and our mind, the most parsimonious scenario applied on a deduced ancestor, which can lead to these two children using the lowest number of edit operations (such as inserted and deleted genes). Figure 9.6(a) shows this matching process applied on *Meta_Delavayi* and *kalo_septemlobus*, which have a core genome of 169 genes (only the 23 first genes are depicted). Note that, in this example, *Meta_Delavayi* (J) and *kalo_septemlobus* (H) match completely, so the ancestor A is very easy to obtain ($A = H \cap J$ has 169 genes).

- **Step 3: Genes Investigation:** In the easiest situation presented above, all couple of genes match completely between the two sister species (which thus have the same length). In this case, whose frequency of occurrences depends on the considered family, the ancestor is easily deduced as being the same than its children. If we have at least one problematic situation between the selected genomes (that is, if we have at least one deleted, duplicated, or inserted gene in one genome), like between *Eleu_senticosus* and *Bras_hainla* (renamed U_1 and U_2 respectively), then a deeper investigation is initiated using one or more cousin genome(s). For instance, in our example, *Meta_Delavayi* and *kalo_septemlobus* will be considered as cousin genomes to take the final decision in the treatment of such problematic situations.

  Mathematically speaking, for all genes in $U_1 \cup U_2$, if gene g_i in U_1 matches properly in name, position, and orientation with g'_i in U_2, then adds it in the ancestor genome γ at position i. Else, consider the position i in cousin genome: if g_i or g'_i equal to g''_i in cousin genome, then add the most frequent gene to the ancestor genome γ in position i; otherwise, this gene is considered as *Insertion*.

  Figure 9.7 gives a simulation example of the considered procedure. In this figure, suppose that A, B, C, D, and E in the leaves are genes, and we want to predict the ancestor α_1. Remark that genes A, B, and D match in positions. The problematic C gene between these two genomes needs a cousin genome to determine its presence in α_1 ancestor genome or not. One or both genomes in α_2 subtree are considered to be cousin(s) to treat the problem of gene C. The two cousin genomes have one copy of gene C in their gene lists. According to our voting system, gene C will be in α_1 ancestor and one delete operation is recorded (that is, AB_D). An insert state is also marked in α_2 subtree, where gene E did not appear neither in cousin genomes of α_1 tree, nor in its brother. With our tree, the graphical presentation of delete operation of gene number 112 is illustrated in Figures 9.6(b) and 9.6(c) comparing with two cousin genomes.

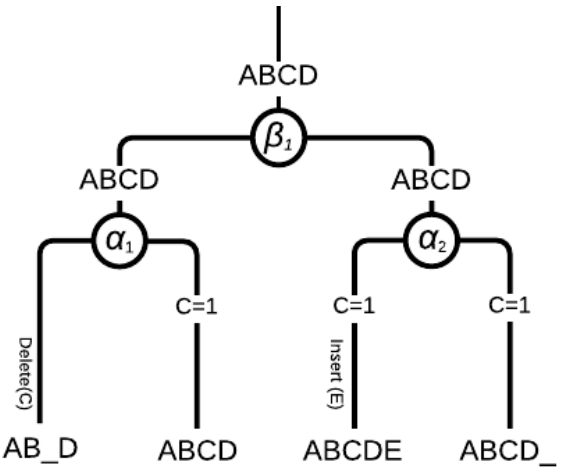


Figure 9.7: Simulation of gene investigation step between two genomes

- **Step 4: Save Ancestor:** After defining all operations between given genomes in gene investigation step, the gene list of ancestral genome is determined. We need to keep this list for further investigation with other leaves or sub-ancestral from other internal nodes until reaching the root node (R). A python pickle file is used to save each predicted internal ancestor with its correspondent node letter.
- **Step 5:** Repeat step 2 until the prediction of root ancestor.

Note that all matching genes are directly assigned to the ancestor. For non-matching genes, the process consists in the selection of a third genome, among the cousins according to the provided tree. The selected cousin is the closest one to the two considered genomes, according to the chosen distance. It is then compared to the two sister species for each non-matching gene: if the cousin agrees with one sister, then the considered gene is added to the ancestor. Figure 9.8 simulates the entire process of ancestral reconstruction between two genomes.

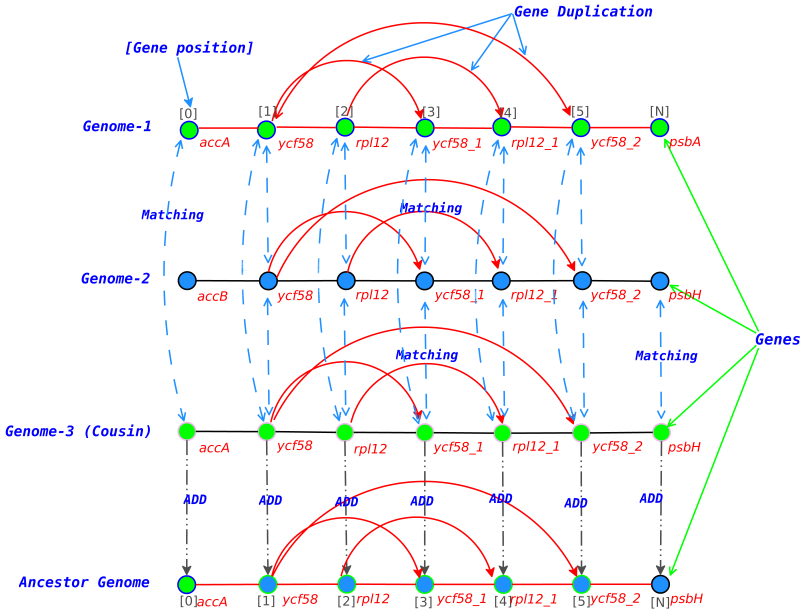


Figure 9.8: Simulation of ancestral reconstruction process between two genomes

The results from this algorithm is shown in Figure 9.9. In this figure, the gene list of the ancestor genome in internal nodes are well predicted by considering sometimes multiple cousin genomes for selecting the most frequent gene for problematic position. To understand the evolution of Apiales order, we use the concept of top-down tree tracing. More precisely, we regards the ancestor of top root node and then we trace down the branches connected to it. The number of inserted and deleted genes are then written on an arc directed from top to bottom.

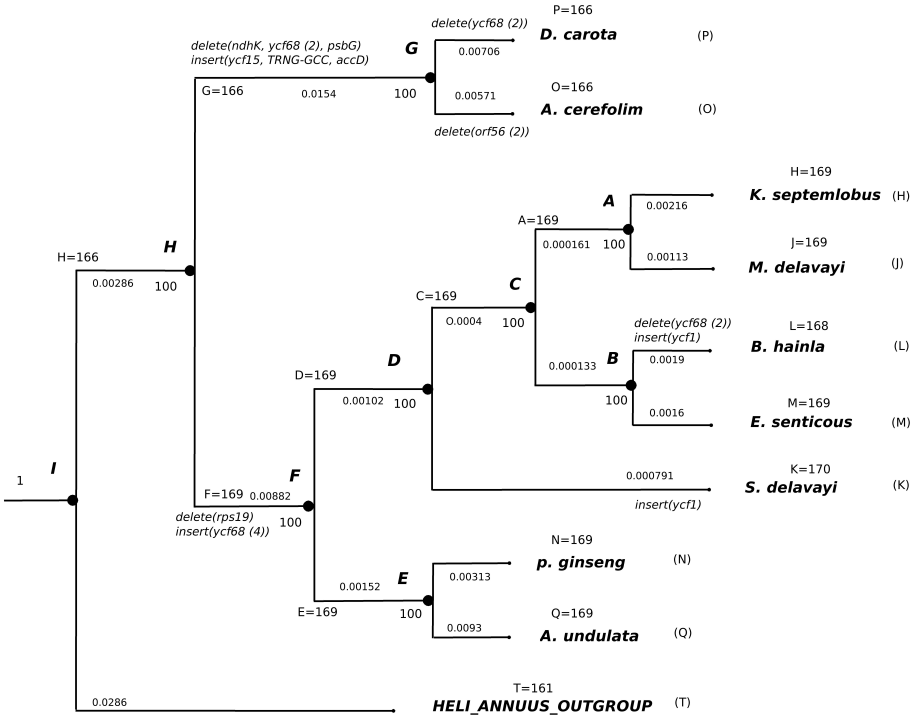


Figure 9.9: Ancestral results from gene contents method.

This method can generally provide the ancestor gene list in each internal node in order fashion, but it also has some limitations: for example, the matching criteria depends on finding the shared genes between closed genomes based on gene name, while sequences are not taken into account. To tackle this problem and to provide more accurate gene lists for internal node ancestors, deeper investigations should be considered.

9.2.2.2/ Ancestor Prediction based on Sequence Comparison

In this section, we consider the same stages than in previous method, except that we add an initialization renaming stage preliminary to Step 1. We consider two datasets, according to genome names set in the previous method. Duplicated genes in each genome are then renamed, depending on the Needleman–Wunsch algorithm [96] sequence similarity distance, by adding an index number to the end of the working gene name.

For illustration purpose, let us consider Figure 9.10. We have three copies of gene C in genome A, and two copies of the same gene in genome B. We need first to rename all duplicated genes in genome A to be a reference genome. Suppose that we want to rename all duplicated genes in B according to A (which operates here as a reference). Duplication in gene names of genome A are firstly renamed according to their position,

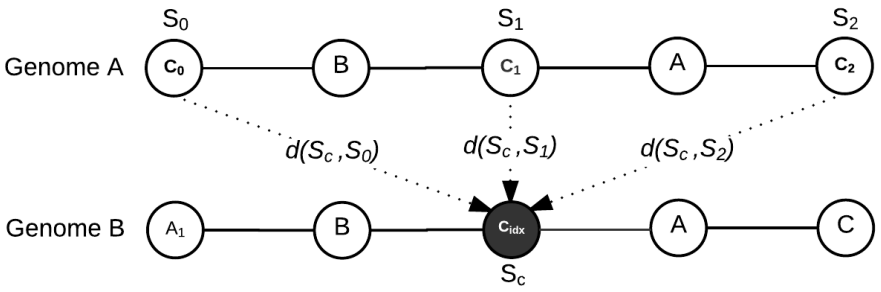


Figure 9.10: General process of renaming genes based on sequence comparisons.

the i-th copy of gene g becoming g_{i-1}, while g_0 is simply noted g, see Example 4.

Example 4: Renaming duplicated gene numerically

Genome A: $[A, B, A, C, B, B, C, D]$

Duplicated genes A, B, and C are renamed numerically to:

Genome A: $[A, B, A_1, C, B_1, B_2, C_1, D]$

Note that, the first copy of duplicated genes is labeled starting from zero label (ex., A_0). All genes of zero label are represented as the same gene name.

Each copy of gene C in genome B is secondly compared to each copy of the same gene in genome A using global sequence alignment distance. It then receives the same index that the gene in A having the best similarity score, based on the Equation 9.5:

$$C_{idx} = \begin{cases} 0 & \text{if } \min(d(S_c,S_0), d(S_c,S_1), d(S_c,S_2)) = d(S_c,S_0), \\ 1 & \text{if } \min(d(S_c,S_0), d(S_c,S_1), d(S_c,S_2)) = d(S_c,S_1), \\ 2 & \text{if } \min(d(S_c,S_0), d(S_c,S_1), d(S_c,S_2)) = d(S_c,S_2), \\ 99 & \text{else.} \end{cases} \tag{9.5}$$

where S_c represents the coding sequence of target gene C in genome B, $d(S_c, S_0)$ represents the similarity score from global sequence alignment comparisons, and S_0, S_1, and S_2 represent the reference coding sequences of gene C in genome A.

The results from this stage are duplication free gene lists for genome A and genome B. If we applied a cousin genome, then we will have a duplication free gene list for the cousin genome. We applied the same next steps from the previous method.

This method has been applied on *Apiales* order. Figure 9.11(b) presents the graphical matching relations among the genes between *Meta_Delavayi* and *kalo_septemlobus* considering this method, which can be compared with Figure 9.11(a) that illustrates our previous method. The tree of *Apiales*, with ancestors generated from this improved method, is shown in Figure 9.12. In this tree, inserted or deleted genes are maybe more accurate now, as we considered sequence comparisons using Needleman-Wunsch algorithm.

Note that a gene is indexed by 99 when it has revealed a low similarity score (lower than 60%) with all associated gene sequences on the reference genome. This particular gene should be further investigated, to identify where it has been inserted during the chloroplast evolution.

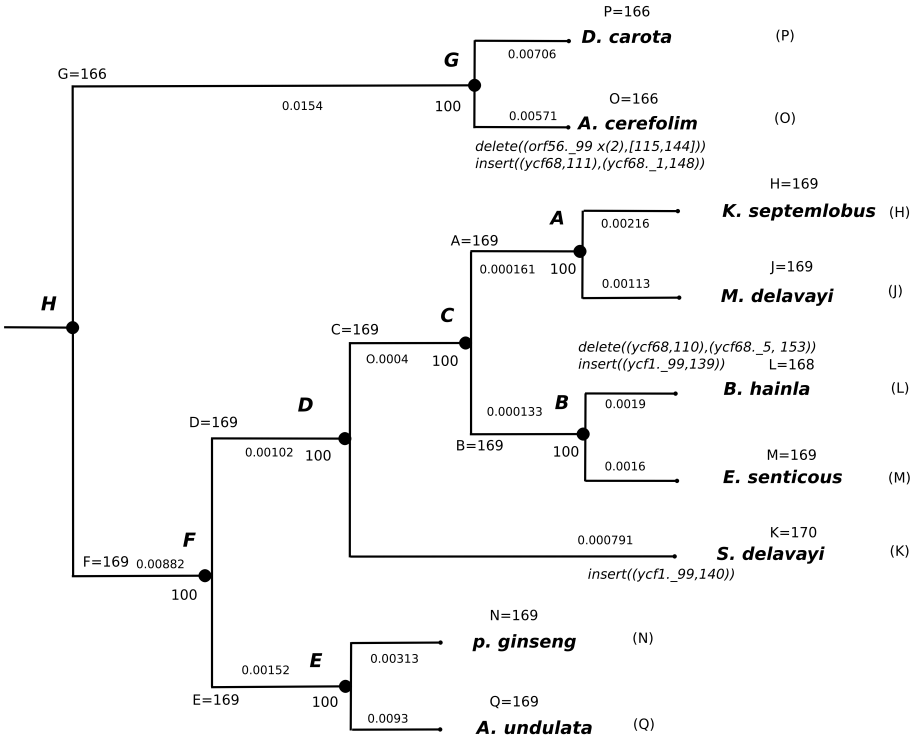


Figure 9.12: Ancestral results from sequence comparison method.

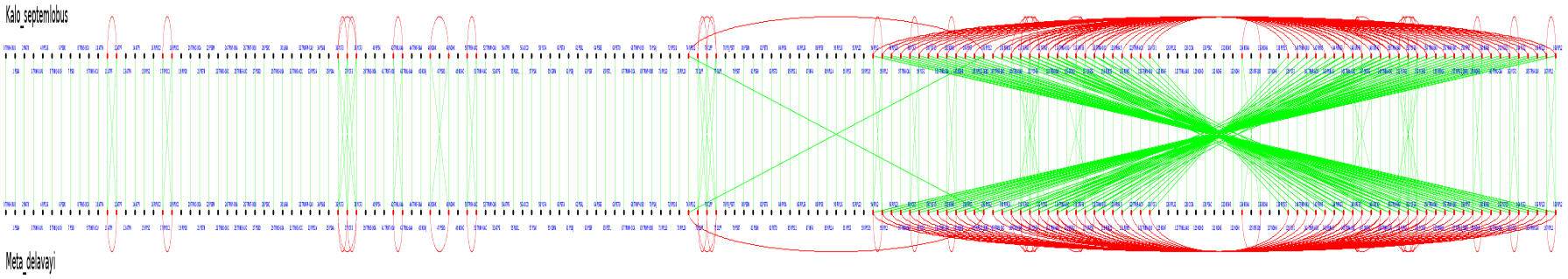
9.2.3/ ANCESTRAL INFORMATION

We now present information regarding different ancestor node in target phylogenetic trees. These information are stored for statistical computations or for future ancestral computation. Table 9.2 presents the amount of duplicated genes inside the ancestors.

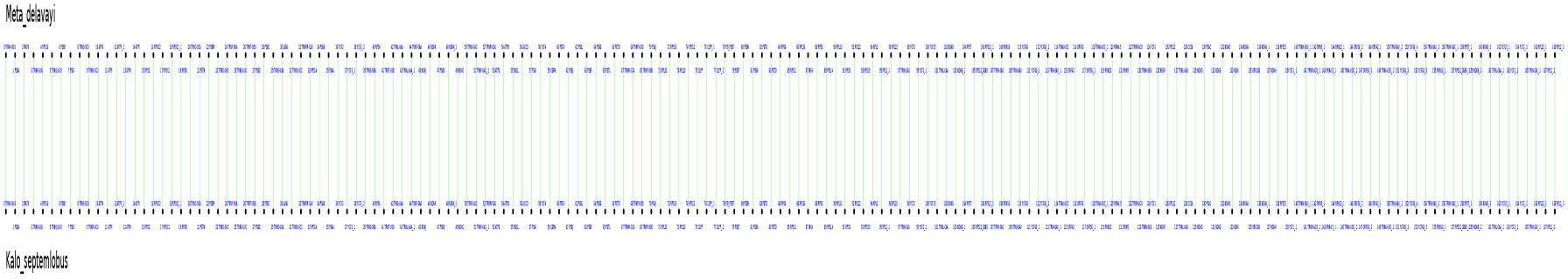
An annotation-based approach has been performed in a pipeline for core genomes by two genomes using NCBI and DOGMA annotation tools. The implicit evolutionary model considered for the ideal method occurs when comparing two genomes (containing the same set of genes, but with different number of copies) developed from a common ancestor: in such case, each genome contains exactly the same copy of each gene, through a series of independent gene duplications and reversals. As demonstrated previously in Figure 9.8.

The central idea of this approach is to keep members of each gene family (from two genomes), its actual prototype, which best reflect the original position of the gene in the ancestor genome. This process proceeds in a bottom-up fashion: it starts from the most recent ancestors of two leaves in the tree and finishes at the root of the tree.

The genomes of the extant species are recorded. The file is in an FASTA format, where, instead of the gene sequence, we list the gene name, ID, with plus (+) or minus (–) signs representing orientations. In this study, we have described another method for ancestral genome reconstruction for chloroplasts. We selected $Apiales$ species. We suggest for all



(a) *Meta_Delavayi* and *kalo_septemlobus* from gene content method.



(b) *Meta_Delavayi* and *kalo_septemlobus* from gene renaming method.

Figure 9.11: Graphical representations between *Meta_Delavayi* and *kalo_septemlobus* from gene content method and gene renaming method.

Gene name	*B. hainla*	*E. senticosus*	*A. undulata*	*P. ginseng*	*M. delavayi*	*A. Cerefolium*	*D. carota*	*S. delavayi*	*K. septemlobus*
accD	0	0	0	0	0	2	2	0	0
rps12	2	2	2	2	2	2	2	2	2
ndhA	2	2	2	2	2	2	2	2	2
ndhK	2	2	2	2	2	0	0	2	2
rpl2	4	4	4	4	4	4	4	4	4
rps7	2	2	2	2	2	2	2	2	2
rpoC1	2	2	2	2	2	2	2	2	2
ycf2	4	4	4	4	4	4	4	4	4
ycf3	3	3	3	3	3	3	3	3	3
rpl23	2	2	2	2	2	2	2	2	2
ycf1	3	2	2	2	2	2	2	3	2
clpP	3	3	3	3	3	3	3	3	3
atpF	2	2	2	2	2	2	2	2	2
orf56	4	4	4	4	4	2	4	4	4
rrn23	2	2	2	2	2	2	2	2	2
ycf68	4	6	6	6	6	2	0	6	4
rrn5	2	2	2	2	2	2	2	2	2
rrn4.5	2	2	2	2	2	2	2	2	2
ycf15	2	2	2	2	2	4	4	2	2
rrn16	2	2	2	2	2	2	2	2	2
orf42	2	2	2	2	2	0	2	2	2
rps19	0	0	0	0	0	2	2	0	0
tRNV-GAC	2	2	2	2	2	2	2	2	2
tRNL-UAA	2	2	2	2	2	2	2	2	2
tRNL-CAA	2	2	2	2	2	2	2	2	2
tRNV-UAC	2	2	2	2	2	2	2	2	2
tRNR-ACG	2	2	2	2	2	2	2	2	2
tRNN-GUU	2	2	2	2	2	2	2	2	2
tRNA-UGC	4	4	4	4	4	4	4	4	4
tRNI-GAU	4	4	4	4	4	4	4	4	4
tRNI-CAU	2	2	2	2	2	2	2	2	2
rps12_3end	2	2	2	2	2	2	2	2	2

Table 9.2: Gene duplication for each genome in *Apiales species*

gene duplications, each copy refers to a separate coding sequence.

As presented in a tree in Figure 9.5 of $Apiales$ order, considering two brothers genomes as leaves: the first step of the algorithm starts by evaluating the distance between all node in the clade. This process will help to select the cousin genome to ensure our results for each gene presents into the internal ancestor genome.

Table 9.2 describes the name and the number of copies of each duplicated gene. We can notice in some cases, if there are differences in the amount of gene duplications between

two selected genomes, then we depend on the cousin genome by regarding the position of the same gene name to make our final decision. In some cases more than one cousin is chosen if one cousin can not perfectly guide the decision.

The evolutionary tree presented in Figure 9.9 shows the basic evolutionary scenario based on two rearrangement operations: *insert* and *delete*. It also illustrates the amount of core genes in each internal ancestor. For both brothers genomes, *E. senticosus* and *B. hainla*, the best cousin genome is *K. septemlobus*. The results from similarity matching process declares that there is only one insertion and one deletion operations difference in gene duplication (for example, the gene *ycf1* is presented twice in *E. senticosus*, while it is presented in three copies in *B. hainla*). In the same way, the gene *Ycf68* is presented in four copies in *B. hainla* and in six copies in *E. senticosus*. The results from the matching process will built the ancestor genome *B*. Both brother genomes *A. undulata* and *P. ginseng* are similar in all genes; the result is assigned to ancestor genome *E*. However, genomes such as *A. undulata* and *p. ginseng*, *M. delavayi* and *k. septemlobus*, *S.delavayi* and *M. delavayi*, *k. septemlobus* and *E. sentucosus* are sharing the same genes names, positions, and number of copies of gene duplication, which means a perfect matching.

This process is repeated until the prediction of the ancestor genome of the root node. Figure 9.9 shows the results of *Apiales* order. In this figure, edit operations (such as insertions and deletions) are recoded in the tree. One delete and one insert gene in internal node *(C)*, The gene *psbG* was presented in all genomes except in *(C)*, while there is an inserted gene of *tRNG-GCC* in *(C)*. The remaining genes are matching perfectly in names and positions.

9.3/ CONCLUSION

In this chapter, we suggested a pipeline of three stages to evaluate how the accuracy of ancestral genomes reconstruction depends on species sampling and quality of the phylogenetic tree. Two methods were applied in ancestral analysis stage: ancestor prediction based on gene contents and ancestor prediction based on sequence comparisons. These two methods are sharing almost the same implementation stages. There is a preliminary stage for sequence comparisons method called gene renaming, where each copy of duplicated gene are compared with all copies of the same gene in reference genome.

The pipeline provides in the last stage some graphical and non-graphical images to highlight the possible rearrangement operations (*e.g.*, insertion, deletion, replace, inversion) on the given tree.

This chapter is a step forward in developing approximation algorithms and investigating the properties of building an accurate phylogenetic tree and reconstructing the ancestral genomes. Finally, this work continues to handle all other species in the domain of *Eukaryotic*, such as (*Ericales, Solanales, Gentianales*, and *Lamiales_Oleaceae*). Many questions still need some deep investigations to completely understand the evolution process in *Eukaryotic* domains by including more gene features and gene sequences. Furthermore, other edit distance operations (such as the role of inversions, transposition, replacement, *etc.*) need to be taken into consideration in future studies.

III

Conclusion and Future Work

CHAPTER 10

Conclusion

10.1/ CONCLUSION

During our thesis, we have investigated some scientific and technical problems that may arise when trying to reconstruct the last universal common ancestor (LUCA) of a large set of all available complete chloroplastic sequences. The problems of genome annotation, core extraction, phylogenetic inference, and ancestral reconstruction have specifically been regarded as key elements in that LUCA quest.

Concerning the automatic annotation problem of complete chloroplastic sequences, we have tested various solutions and decided that an updated version of DOGMA was the best compromise. The problem in this context was that, except DOGMA, most coding sequence prediction or annotation tools were not specific to chloroplasts and so their accuracies were perfectible, as it has been verified on well humanly curated genomes. Conversely, annotations on databases like the NCBI one were of too much varying quality, some genomes being annotated and curated well while other ones embedded obvious annotation errors. The objective at this level was to take the best from both systematic annotations from DOGMA and humanly curated ones from NCBI, by using name and sequence similarities. We have shown that such an approach may introduce other kind of artifacts, and that finally DOGMA alone provides annotations of sufficiently accurate level.

Given a set of close annotated chloroplastic genomes, we then have investigated, how to extract the largest subset of core genes that lead to the most supported phylogenetic tree. We have proposed two artificial intelligence ways to reach this goal, namely by the mean of genetic algorithms or particle swarm optimization. A second stage encompassing both LASSO test and dummy binary logistic regression has been added to the algorithm, in order to describe the effect of each gene on topology selection and on support evolution. These algorithms have been deployed on the Mésocentre de Calculs de Franche-Comté thanks to a distributed master/slave approach.

On such trees, we have proposed a first ancestral reconstruction of gene content and order. This algorithm is based on SequenceMatcher tool, and obtained results have been verified to what can be deduced by naked eye on well defined families. Ways to merge

the forest of phylogenetic trees in a supertree have been regarded too, and the way gene content evolves through a tree of core genomes has finally been presented.

10.2/ Future Investigative Directions

In future work, our main objective will be to complete what has been initiated during our thesis, until being able to reconstruct the last universal common ancestor of chloroplasts.

The current annotation process using DOGMA seems reliable and convenient, but we need to investigate more deeply the case of fragmented genes. Testing all possible combinations of fragments is a very costly task, and choosing the combination that has the higher similarity score with a gene database we have constructed may, in some cases, provide artifacts or chimeras. This is why we believe that the fragmented genes must be deeply regarded, by considering two times a list of such genes, and checking if the automatically obtained defragmented genes are coherent with what can be manually inferred.

The way to separate the large set of inputted genomes is currently based on taxonomy information according to the NCBI database. However this latter is not completely reliable, and new sequencing capability makes that the whole taxonomy is currently evolving. The risk in this case is to consider by error a divergent species in a subset of coherent species – this latter being then misplaced after the supertree reconstruction, leading to errors in all its ancestral nodes. A way to reinforce confidence put in the subset selection is to compare gene content and sequence similarity too, to be sure that we put together only close species.

Other artificial intelligence approaches like simulated annealing should be compared to our genetic algorithm and our particle swarm optimization, when we attempt to extract the largest subset of core genes that produces the most supported tree, and hybrid approaches must be investigated too. They must be compared to a brute force approach on a small family, to see if the optimum produced by our algorithm is really the “best tree”. Investigating the per site likelihood level instead of the gene one may be more relevant, and thus it must be regarded. However the number of observed characters dramatically explodes, so it is likely that such an approach is impossible in practice. Statistical results concerning the gene effects on topology and supports must be explained biologically, among other things by investigating the gene functionality, or by regarding whether gene transfer may explain it. Finally, the trees we produce must be compared with gene trees, and similarities or differences must be explained.

We currently reconstruct the supertree by hand, and we have not yet proposed an algorithm to achieve this goal. No solution can be found in the literature, as supertree reconstruction currently supposes that the same genes are shared among the trees to merge, or other restrictive hypotheses of that kind, which are incompatible with our supertree problem.

Concerning ancestral reconstruction of gene order and content, the first solution we proposed is perfectible. In particular, we do not obtain exactly the same ancestor that what we have reconstructed manually. Divergences must be understood and our algorithm must be updated and simplified. Sequence level must be added too to the algorithm.

Finally, the whole pipeline must be finalized, deployed on the Mésocentre or on DARI resources, and it should be launched on all currently available chloroplastic genomes, and obtained results must be carefully regarded. In particular, the last universal common ancestor must be compared to cyanobacterial genomes, to see if a cyanobacterial origin of chloroplasts can be assessed by the mean of ancestral reconstruction. Gene content and recombination events must be investigated too in the supertree, to see if some branches

in the tree can be related to hot spots of evolution. Endosymbiosis events among the supertree must be searched too, while possible gene transfer to the nucleus genome must finally be studied.

List of Figures

List of Tables

List of Definitions

List of remarks

List of examples

Bibliography

[1] NIGEL CHAFFEY. Alberts, b., johnson, a., lewis, j., raff, m., roberts, k. and walter, p. molecular biology of the cell. *Annals of Botany*, 91(3):401–401, 2003.

[2] Bettina Stoebe, William Martin, and Klaus V Kowallik. Distribution and nomenclature of protein-coding genes in 12 sequenced chloroplast genomes. *Plant Molecular Biology Reporter*, 16(3):243–255, 1998.

[3] Daniel Grzebyk, Oscar Schofield, Costantino Vetriani, and Paul G Falkowski. The mesozoic radiation of eukaryotic algae: The portable plastid hypothesis1. *Journal of Phycology*, 39(2):259–267, 2003.

[4] Itai Sharon, Ariella Alperovitch, Forest Rohwer, Matthew Haynes, Fabian Glaser, Nof Atamna-Ismaeel, Ron Y Pinter, Frédéric Partensky, Eugene V Koonin, Yuri I Wolf, Nathan Nelson, and Oded Béjà. Photosystem i gene cassettes are present in marine virus genomes. *Nature*, 461(7261):258–262, 2009.

[5] Matteo De Chiara, Derek Hood, Alessandro Muzzi, Derek J Pickard, Tim Perkins, Mariagrazia Pizza, Gordon Dougan, Rino Rappuoli, E Richard Moxon, Marco Soriani, and Claudio Donati. Genome sequencing of disease and carriage isolates of non typeable haemophilus influenzae identifies discrete population structure. *Proceedings of the National Academy of Sciences*, 111(14):5439–5444, 2014.

[6] Stefan Kurtz, Adam Phillippy, Arthur L Delcher, Michael Smoot, Martin Shumway, Corina Antonescu, and Steven L Salzberg. Versatile and open software for comparing large genomes. *Genome biology*, 5(2):R12, 2004.

[7] Hervé Tettelin, Vega Masignani, Michael J Cieslewicz, Claudio Donati, Duccio Medini, Naomi L Ward, Samuel V Angiuoli, Jonathan Crabtree, Amanda L Jones, A Scott Durkin, Robert T. DeBoy, Tanja M. Davidsen, Marirosa Mora, Maria Scarselli, Immaculada Margarit, Jeremy D. Peterson, Christopher R. Hauser, Jaideep P. Sundaram, William C. Nelson, Ramana Madupu, Lauren M. Brinkac, Robert J. Dodson, Mary J. Rosovitz, Steven A. Sullivan, Sean C. Daugherty, Daniel H. Haft, Jeremy Selengut, Michelle L. Gwinn, Liwei Zhou, Nikhat Zafar, Hoda Khouri, Diana Radune, George Dimitrov, Kisha Watkins, Kevin J. B. O'Connor, Shannon Smith, Teresa R. Utterback, Owen White, Craig E. Rubens, Guido Grandi, Lawrence C. Madoff, Dennis L. Kasper, John L. Telford, Michael R. Wessels, Rino Rappuoli, and Claire M. Fraser. Genome analysis of multiple pathogenic isolates of streptococcus agalactiae: implications for the microbial "pan-genome". *Proceedings of the National Academy of Sciences of the United States of America*, 102(39):13950–13955, 2005.

[8] Michael S Rosenberg. *Sequence alignment: methods, models, concepts, and strategies*. Univ of California Press, 2009.

[9] GR Reeck, C DE HAËN, DC TELLER, RF DOOLITTLE, WM FITCH, RE DICKERSON, P CHAMBON, AD MCLACHLAN, E MARGOLIASH, and TH JUKES. E. zuckerkandl. 1987. homology in proteins and nucleic acids: a terminology muddle and a way out of it. *Cell*, 50:667.

[10] Jonathan Pevsner. *Bioinformatics and functional genomics*. John Wiley & Sons, 2005.

[11] Peter Weiner. Linear pattern matching algorithms. In *Switching and Automata Theory, 1973. SWAT'08. IEEE Conference Record of 14th Annual Symposium on*, pages 1–11. IEEE, 1973.

[12] Esko Ukkonen. Algorithms for approximate string matching. *Information and control*, 64(1):100–118, 1985.

[13] Allan C Wilson and Vincent M Sarich. A molecular time scale for human evolution. *Proceedings of the National Academy of Sciences*, 63(4):1088–1093, 1969.

[14] Motoo Kimura. A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *Journal of molecular evolution*, 16(2):111–120, 1980.

[15] Masami Hasegawa, Hirohisa Kishino, and Taka-aki Yano. Dating of the human-ape splitting by a molecular clock of mitochondrial dna. *Journal of molecular evolution*, 22(2):160–174, 1985.

[16] FJLOJ Rodriguez, JL Oliver, A Marin, and J Rß Medina. The general stochastic model of nucleotide substitution. *Journal of theoretical biology*, 142(4):485–501, 1990.

[17] Koichiro Tamura and Masatoshi Nei. Estimation of the number of nucleotide substitutions in the control region of mitochondrial dna in humans and chimpanzees. *Molecular biology and evolution*, 10(3):512–526, 1993.

[18] Jacques M Bahi, Christophe Guyeux, and Antoine Perasso. Predicting the evolution of two genes in the yeast saccharomyces cerevisiae. *Procedia Computer Science*, 11:4–16, 2012.

[19] Peter Godfrey-Smith and Kim Sterelny. Biological information. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Fall 2008 edition, 2008.

[20] Margaret O Dayhoff and Robert M Schwartz. A model of evolutionary change in proteins. In *In Atlas of protein sequence and structure*. Citeseer, 1978.

[21] Federico Abascal, Rafael Zardoya, and David Posada. Prottest: selection of best-fit models of protein evolution. *Bioinformatics*, 21(9):2104–2105, 2005.

[22] J.-P. Comet. *Programmation dynamique et comparaison de séquences biologiques*. PhD thesis, Université de Technologie de Compiègne, Novembre 1998.

[23] Steven Henikoff and Jorja G Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences*, 89(22):10915–10919, 1992.

[24] Steven Henikoff, Jorja G. Henikoff, and Shmuel Pietrokovski. Blocks+: a non-redundant database of protein alignment blocks derived from multiple compilations. *Bioinformatics*, 15(6):471–479, 1999.

[25] Stephen F Altschul, Warren Gish, Webb Miller, Eugene W Myers, and David J Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, 1990.

[26] David J Lipman and William R Pearson. Rapid and sensitive protein similarity searches. *Science*, 227(4693):1435–1441, 1985.

[27] Julie D Thompson, Toby Gibson, Des G Higgins, et al. Multiple sequence alignment using clustalw and clustalx. *Current protocols in bioinformatics*, pages 2–3, 2002.

[28] Robert C Edgar. Muscle: multiple sequence alignment with high accuracy and high throughput. *Nucleic acids research*, 32(5):1792–1797, 2004.

[29] Cédric Notredame, Desmond G Higgins, and Jaap Heringa. T-coffee: A novel method for fast and accurate multiple sequence alignment. *Journal of molecular biology*, 302(1):205–217, 2000.

[30] Temple F Smith and Michael S Waterman. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197, 1981.

[31] Osamu Gotoh. An improved algorithm for matching biological sequences. *Journal of molecular biology*, 162(3):705–708, 1982.

[32] Vladimir Iosifovich Levenshtein. Binary codes with correction for deletions and insertions of the symbol 1. *Problemy Peredachi Informatsii*, 1(1):12–25, 1965.

[33] Li Yujian and Liu Bo. A normalized levenshtein distance metric. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(6):1091–1095, 2007.

[34] Eric C. Rouchka Jeffrey Rizzo. Review of phylogenetic tree construction. *University of Louisville Bioinformatics Laboratory Technical Report Series*, (TR-ULBL-2007-01):2–7, 2007.

[35] Luigi L Cavalli-Sforza and Anthony WF Edwards. Phylogenetic analysis. models and estimation procedures. *American journal of human genetics*, 19(3 Pt 1):233, 1967.

[36] Pamela S Soltis, Douglas E Soltis, and Mark W Chase. Angiosperm phylogeny inferred from multiple genes as a tool for comparative biology. *Nature*, 402(6760):402–404, 1999.

[37] Deren AR Eaton and Richard H Ree. Inferring phylogeny and introgression using radseq data: an example from flowering plants (pedicularis: Orobanchaceae). *Systematic Biology*, 62(5):689–706, 2013.

[38] Naruya Saitou and Masatoshi Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular biology and evolution*, 4(4):406–425, 1987.

[39] William J Bruno, Nicholas D Socci, and Aaron L Halpern. Weighted neighbor joining: a likelihood-based approach to distance-based phylogeny reconstruction. *Molecular Biology and Evolution*, 17(1):189–197, 2000.

[40] Y Tateno, N Takezaki, and M Nei. Relative efficiencies of the maximum-likelihood, neighbor-joining, and maximum-parsimony methods when substitution rate varies with site. *Molecular Biology and Evolution*, 11(2):261–277, 1994.

[41] John P Huelsenbeck, Fredrik Ronquist, Rasmus Nielsen, and Jonathan P Bollback. Bayesian inference of phylogeny and its impact on evolutionary biology. *science*, 294(5550):2310–2314, 2001.

[42] Chuong B Do, Mahathi SP Mahabhashyam, Michael Brudno, and Serafim Batzoglou. Probcons: Probabilistic consistency-based multiple sequence alignment. *Genome research*, 15(2):330–340, 2005.

[43] Amarendran R Subramanian, Jan Weyer-Menkhoff, Michael Kaufmann, and Burkhard Morgenstern. Dialign-t: an improved algorithm for segment-based multiple sequence alignment. *BMC bioinformatics*, 6(1):66, 2005.

[44] Kazutaka Katoh, George Asimenos, and Hiroyuki Toh. Multiple alignment of dna sequences with mafft. In *Bioinformatics for DNA sequence analysis*, pages 39–64. Springer, 2009.

[45] Jimin Pei, Ruslan Sadreyev, and Nick V Grishin. Pcma: fast and accurate multiple sequence alignment based on profile consistency. *Bioinformatics*, 19(3):427–428, 2003.

[46] Alexandros Stamatakis, Thomas Ludwig, and Harald Meier. Raxml-iii: a fast program for maximum likelihood-based inference of large phylogenetic trees. *Bioinformatics*, 21(4):456–463, 2005.

[47] Bassam Alkindy, Jean-François Couchot, Christophe Guyeux, Arnaud Mouly, Michel Salomon, and Jacques M. Bahi. Finding the core-genes of chloroplasts. *Journal of Bioscience, Biochemistery, and Bioinformatics*, 4(5):357–364, 2014.

[48] Bassam AlKindy, Christophe Guyeux, Jean-François Couchot, Michel Salomon, and Jacques M Bahi. Gene similarity-based approaches for determining core-genes of chloroplasts. In *Bioinformatics and Biomedicine (BIBM), 2014 IEEE International Conference on*, pages 71–74. IEEE, 2014.

[49] Bassam AlKindy, Huda Al-Nayyef, Christophe Guyeux, Jean-François Couchot, Michel Salomon, and Jacques M. Bahi. Improved core genes prediction for constructing well-supported phylogenetic trees in large sets of plant species. In Francisco Ortuño and Ignacio Rojas, editors, *Bioinformatics and Biomedical Engineering*, volume 9043 of *Lecture Notes in Computer Science*, pages 379–390. Springer International Publishing, 2015.

[50] Eric W Sayers, Tanya Barrett, Dennis A Benson, Evan Bolton, Stephen H Bryant, Kathi Canese, Vyacheslav Chetvernin, Deanna M Church, Michael DiCuccio, Scott Federhen, Michael Feolo, Ian M. Fingerman, Lewis Y. Geer, Wolfgang Helmberg, Yuri Kapustin, David Landsman, David J. Lipman, Zhiyong Lu, Thomas L. Madden, Tom Madej, Donna R. Maglott, Jian Ye, Aron Marchler-Bauer, Vadim Miller, Ilene

Mizrachi, James Ostell, Panchenko, Anna, Lon Phan, Kim D. Pruitt, Gregory D. Schuler, Edwin Sequeira, Stephen T. Sherry, Martin Shumway, Karl Sirotkin, Douglas Slotta, Alexandre Souvorov, Grigory Starchenko, Tatiana A. Tatusova, Lukas Wagner, Yanli Wang, W. John Wilbur, and Eugene Yaschenko. Database resources of the national center for biotechnology information. *Nucleic acids research*, 39(suppl 1):D38–D51, 2011.

[51] Tamara Kulikova, Ruth Akhtar, Philippe Aldebert, Nicola Althorpe, Mikael Andersson, Alastair Baldwin, Kirsty Bates, Sumit Bhattacharyya, Lawrence Bower, Paul Browne, et al. Embl nucleotide sequence database in 2006. *Nucleic acids research*, 35(suppl 1):D16–D20, 2007.

[52] Hideaki Sugawara, Osamu Ogasawara, Kousaku Okubo, Takashi Gojobori, and Yoshio Tateno. Ddbj with new system and face. *Nucleic acids research*, 36(suppl 1):D22–D24, 2008.

[53] Peter Bakke, Nick Carney, Will DeLoache, Mary Gearing, Kjeld Ingvorsen, Matt Lotz, Jay McNair, Pallavi Penumetcha, Samantha Simpson, Laura Voss, et al. Evaluation of three automated genome annotations for halorhabdus utahensis. *PLoS One*, 4(7):e6291, 2009.

[54] Stacia K. Wyman, Robert K. Jansen, and Jeffrey L. Boore. Automatic annotation of organellar genomes with dogma. *BIOINFORMATICS, oxford Press*, 20(172004):3252–3255, 2004.

[55] Javier De Las Rivas, Juan Jose Lozano, and Angel R Ortiz. Comparative analysis of chloroplast genomes: functional annotation, genome-based phylogeny, and deduced evolutionary patterns. *Genome research*, 12(4):567–583, 2002.

[56] Chang Liu, Linchun Shi, Yingjie Zhu, Haimei Chen, Jianhui Zhang, Xiaohan Lin, and Xiaojun Guan. Cpgavas, an integrated web server for the annotation, visualization, analysis, and genbank submission of completely sequenced chloroplast genome sequences. *BMC genomics*, 13(1):715, 2012.

[57] Genis Parra, Keith Bradnam, and Ian Korf. Cegma: a pipeline to accurately annotate core genes in eukaryotic genomes. *Bioinformatics*, 23(9):1061–1067, 2007.

[58] Ewan Birney, Michele Clamp, and Richard Durbin. Genewise and genomewise. *Genome research*, 14(5):988–995, 2004.

[59] Genís Parra, Enrique Blanco, and Roderic Guigó. Geneid in drosophila. *Genome research*, 10(4):511–515, 2000.

[60] Stephane Guindon, Franck Lethiec, Patrice Duroux, and Olivier Gascuel. Phyml online—a web server for fast maximum likelihood-based phylogenetic inference. *Nucleic acids research*, 33(suppl 2):W557–W559, 2005.

[61] Alexandros Stamatakis. The raxml 7.0. 4 manual. *Department of Computer Science. Ludwig-Maximilians-Universität München*, 2008.

[62] Alexandros Stamatakis. Raxml version 8: A tool for phylogenetic analysis and post-analysis of large phylogenies. *Bioinformatics*, 2014.

[63] Matthew Kearse, Richard Moir, Amy Wilson, Steven Stones-Havas, Matthew Cheung, Shane Sturrock, Simon Buxton, Alex Cooper, Sidney Markowitz, Chris Duran, et al. Geneious basic: an integrated and extendable desktop software platform for the organization and analysis of sequence data. *Bioinformatics*, 28(12):1647–1649, 2012.

[64] Vincent Ranwez, Alexis Criscuolo, and Emmanuel JP Douzery. Supertriplets: a triplet-based supertree approach to phylogenomics. *Bioinformatics*, 26(12):i115–i123, 2010.

[65] Geoffrey Ian McFadden. Primary and secondary endosymbiosis and the origin of plastids. *Journal of Phycology*, 37(6):951–959, 2001.

[66] Xi Li, Ti-Cao Zhang, Qin Qiao, Zhumei Ren, Jiayuan Zhao, Takahiro Yonezawa, Masami Hasegawa, M. James C Crabbe, Jianqiang Li, and Yang Zhong. Complete chloroplast genome sequence of holoparasite *Cistanche deserticola* (orobanchaceae) reveals gene loss and horizontal gene transfer from its host *Haloxylon ammodendron*(chenopodiaceae). *PLoS ONE*, 8(3):e58747, 03 2013.

[67] Bassam AlKindy, Christophe Guyeux, Jean-François Couchot, Michel Salomon, Christian Parisod, and Jacques M. Bahi. Hybrid genetic algorithm and lasso test approach for inferring well supported phylogenetic trees based on subsets of chloroplastic core genes. In Adrian-Horia Dediu, Francisco Hernández-Quiroz, Carlos Martín-Vide, and David A. Rosenblueth, editors, *Algorithms for Computational Biology*, volume 9199 of *Lecture Notes in Computer Science*, pages 83–96. Springer International Publishing, 2015.

[68] Dinabandhu Bhandari, CA Murthy, and Sankar K Pal. Genetic algorithm with elitist model and its convergence. *International Journal of Pattern Recognition and Artificial Intelligence*, 10(06):731–747, 1996.

[69] Lashon B. Booker, David E. Goldberg, and John H. Holland. Classifier systems and genetic algorithms. *Artificial intelligence*, 40(1):235–282, 1989.

[70] S-I Tate, Ikuo Yoshihara, Kunihito Yamamori, and Moritoshi Yasunaga. A parallel hybrid genetic algorithm for multiple protein sequence alignment. In *Computational Intelligence, Proceedings of the World on Congress on*, volume 1, pages 309–314. IEEE, 2002.

[71] Mridu Gupta and Shailendra Singh. A novel genetic algorithm based approach for optimization of distance matrix for phylogenetic tree construction. *International Journal of Computer Applications*, 52(9):14–18, 2012.

[72] Hideo Matsuda. Construction of phylogenetic trees from amino acid sequences using a genetic algorithm. In *Proceedings of Genome Informatics Workshop*, volume 6, pages 19–28, 1995.

[73] John H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, 1975.

[74] John H. Holland. *Adaptation in Natural and Artificial Systems - Second edition*. MIT Press, Cambridge, MA, USA, 1992.

[75] Eric Krevice Prebys. The genetic algorithm in computer science. *MIT Undergrad. J. Math*, 2007:165–170, 2007.

[76] David E Goldberg. Genetic algorithms in search, optimization and machine learning. *Reading: Addison-Wesley*, 1993.

[77] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society (Series B)*, 58:267–288, 1996.

[78] Jeffrey D Palmer. Plastid chromosomes: structure and evolution. *The molecular biology of plastids*, 7:5–53, 1991.

[79] Reem Alsrraj, Bassam AlKindy, Christophe Guyeux, Laurent Philippe, and Jean-François Couchot. Well-supported phylogenies using largest subsets of core-genes by discrete particle swarm optimization. *Proceedings of CIBB*, 2:1, 2015.

[80] James Kenndy and RC Eberhart. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948, 1995.

[81] James Kennedy. Particle swarm optimization. pages 760–766, 2010.

[82] Mohammad Teshnehlab Mahdi Aliyari Shoorehdeli Mojtaba Ahmadieh Khanesar, Hassan Tavakoli. Novel binary particle swarm optimization. *www.intechopen.com*, (978-953-7619-48-0):11, 2009.

[83] K Premalatha and AM Natarajan. Hybrid pso and ga for global maximization. *Int. J. Open Problems Compt. Math*, 2(4):597–608, 2009.

[84] Tim Blackwell Riccardo Poli, James Kennedy. Particle swarm optimization. *Springer Science + Business Media*, 1(10.1007/s11721-007-0002-0):33–57, 2007.

[85] Hidetoshi Shimodaira and Masami Hasegawa. Consel: for assessing the confidence of phylogenetic tree selection. *Bioinformatics*, 17(12):1246–1247, 2001.

[86] D. Sedighizadeh and E. Masehian. Particle swarm optimization methods, taxonomy and applications. *International Journal of Computer Theory and Engineering*, 1(5):486–502, 2009.

[87] M. Clerc. The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 3. IEEE, 1999.

[88] Mathauieu Blanchette, Abdoulaye Baniré Diallo, Eric D Green, Webb Miller, and David Haussler. Computational reconstruction of ancestral dna sequences. In *Phylogenomics*, pages 171–184. Springer, 2008.

[89] Virginie Lopez Rascol, Pierre Pontarotti, and Anthony Levasseur. Ancestral animal genomes reconstruction. *Current opinion in immunology*, 19(5):542–546, 2007.

[90] Bret Larget, Donald L. Simon, Joseph B. Kadane, and Deborah Sweet. A bayesian analysis of metazoan mitochondrial genome arrangements. *Molecular Biology and Evolution*, 22(3):486–495, 2005.

[91] Fei Hu, Yu Lin, and Jijun Tang. MLGO: phylogeny reconstruction and ancestral inference from gene-order data. *BMC Bioinformatics*, 15:354, 2014.

[92] Sridhar Hannenhalli, Colombe Chappey, Eugene V Koonin, and Pavel A Pevzner. Genome sequence comparison and scenarios for gene rearrangements: A test case. *Genomics*, 30(2):299–311, 1995.

[93] John W. Ratcliff and David E. Metzener. Pattern matching: The gestalt approach. 13(7):46, 47, 59–51, 68–72, July 1988.

[94] Bassam Alkindy, Christophe Guyeux, Jean-François Couchot, Michel Salomon, and Jacques Bahi. Using genetic algorithm for optimizing phylogenetic tree inference in plant species. In *MCEB15, Mathematical and Computational Evolutionary Biology*, Porquerolles Island, France, June 2015. poster.

[95] S. Needleman and C. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, March 1970.

[96] Tolga Güyer, Bilal Atasoy, and Sibel Somyürek. Measuring disorientation based on the needleman-wunsch algorithm. *The International Review of Research in Open and Distributed Learning*, 16(2), 2015.

Document generated with LaTeX and:
the LaTeX style for PhD Thesis created by S. Galland — http://www.multiagent.fr/ThesisStyle
the tex-upmethodology package suite — http://www.arakhne.org/tex-upmethodology/

Abstract:

In Bioinformatics, understanding how DNA molecules have evolved over time remains an open and complex problem. Algorithms have been proposed to solve this problem, but they are limited either to the evolution of a given character (for example, a specific nucleotide), or conversely focus on large nuclear genomes (several billion base pairs), the latter having known multiple recombination events - the problem is NP complete when you consider the set of all possible operations on these sequences, no solution exists at present. In this thesis, we tackle the problem of reconstruction of ancestral DNA sequences by focusing on the nucleotide chains of intermediate size, and have experienced relatively little recombination over time: chloroplast genomes. We show that at this level the problem of the reconstruction of ancestors can be resolved, even when you consider the set of all complete chloroplast genomes currently available. We focus specifically on the order and ancestral gene content, as well as the technical problems this raises reconstruction in the case of chloroplasts. We show how to obtain a prediction of the coding sequences of a quality such as to allow said reconstruction and how to obtain a phylogenetic tree in agreement with the largest number of genes, on which we can then support our back in time - the latter being finalized. These methods, combining the use of tools already available (the quality of which has been assessed) in high performance computing, artificial intelligence and bio-statistics were applied to a collection of more than 450 chloroplast genomes.

Keywords: core genome, clustering algorithms, genetic algorithm, particle swarm optimization, dynamic systems, intelligent algorithms, ancestral reconstruction, phylogenetic tree.

Résumé :

En bio-informatique, comprendre comment les molécules d'ADN ont évolué au cours du temps reste un problème ouvert et complexe. Des algorithmes ont été proposés pour résoudre ce problème, mais ils se limitent soit à l'évolution d'un caractère donné (par exemple, un nucléotide précis), ou se focalisent a contrario sur de gros génomes nucléaires (plusieurs milliards de paires de base), ces derniers ayant connus de multiples événements de recombinaison – le problème étant NP complet quand on considère l'ensemble de toutes les opérations possibles sur ces séquences, aucune solution n'existe à l'heure actuelle. Dans cette thèse, nous nous attaquons au problème de reconstruction des séquences ADN ancestrales en nous focalisant sur des chaînes nucléotidiques de taille intermédiaire, et ayant connu assez peu de recombinaison au cours du temps : les génomes de chloroplastes. Nous montrons qu'à cette échelle le problème de la reconstruction d'ancêtres peut être résolu, même quand on considère l'ensemble de tous les génomes chloroplastiques complets actuellement disponibles. Nous nous concentrons plus précisément sur l'ordre et le contenu ancestral en gènes, ainsi que sur les problèmes techniques que cette reconstruction soulève dans le cas des chloroplastes. Nous montrons comment obtenir une prédiction des séquences codantes d'une qualité telle qu'elle permette ladite reconstruction, puis comment obtenir un arbre phylogénétique en accord avec le plus grand nombre possible de gènes, sur lesquels nous pouvons ensuite appuyer notre remontée dans le temps – cette dernière étant en cours de finalisation. Ces méthodes, combinant l'utilisation d'outils déjà disponibles (dont la qualité a été évaluée) à du calcul haute performance, de l'intelligence artificielle et de la bio-statistique, ont été appliquées à une collection de plus de 450 génomes chloroplastiques.

Mots-clés : génome noyau, algorithme génétique, optimisation par essaim de particules, systèmes dynamiques, algorithmes intelligents, reconstruction ancestrale, arbre phylogénétique.

SPIM

■ École doctorale SPIM 16 route de Gray F - 25030 Besançon cedex
■ tél. +33 (0)3 81 66 66 02 ■ ed-spim@univ-fcomte.fr ■ www.ed-spim.univ-fcomte.fr

UFC
UNIVERSITÉ DE FRANCHE-COMTÉ