Recherche automatisée de motifs dans les arbres phylogénétiques

Thomas Bigot

N° 85-2013 Année 2013

THÈSE DE L'UNIVERSITÉ DE LYON

Présentée

devant L'UNIVERSITÉ CLAUDE BERNARD LYON 1

pour l'obtention

du DIPLÔME DE DOCTORAT
(arrêté du 7 août 2006)

soutenue publiquement le
5 juin 2013

par

Thomas BIGOT

Recherche automatisée de motifs dans les arbres phylogénétiques

Directeur de thèse : Guy PERRIÈRE

Jury :	Céline BROCHIER	Présidente du jury
	Emmanuel DOUZERY	Rapporteur
	Simonetta GRIBALDO	Rapporteur
	Claudine MÉDIGUE	Examinateur
	Guy PERRIÈRE	Directeur de thèse
	Claude THERMES	Examinateur

UNIVERSITE CLAUDE BERNARD - LYON 1

Président de l'Université	**M. François-Noël GILLY**
Vice-président du Conseil d'Administration	M. le Professeur Hamda BEN HADID
Vice-président du Conseil des Etudes et de la Vie Universitaire	M. le Professeur Philippe LALLE
Vice-président du Conseil Scientifique	M. le Professeur Germain GILLET
Directeur Général des Services	M. Alain HELLEU

COMPOSANTES SANTE

Faculté de Médecine Lyon Est – Claude Bernard	Directeur : M. le Professeur J. ETIENNE
Faculté de Médecine et de Maïeutique Lyon Sud – Charles Mérieux	Directeur : Mme la Professeure C. BURILLON
Faculté d'Odontologie	Directeur : M. le Professeur D. BOURGEOIS
Institut des Sciences Pharmaceutiques et Biologiques	Directeur : Mme la Professeure C. VINCIGUERRA
Institut des Sciences et Techniques de la Réadaptation	Directeur : M. le Professeur Y. MATILLON
Département de formation et Centre de Recherche en Biologie Humaine	Directeur : M. le Professeur P. FARGE

COMPOSANTES ET DEPARTEMENTS DE SCIENCES ET TECHNOLOGIE

Faculté des Sciences et Technologies	Directeur : M. le Professeur F. DE MARCHI
Département Biologie	Directeur : M. le Professeur F. FLEURY
Département Chimie Biochimie	Directeur : Mme le Professeur H. PARROT
Département GEP	Directeur : M. N. SIAUVE
Département Informatique	Directeur : M. le Professeur S. AKKOUCHE
Département Mathématiques	Directeur : M. le Professeur A. GOLDMAN
Département Mécanique	Directeur : M. le Professeur H. BEN HADID
Département Physique	Directeur : Mme S. FLECK
Département Sciences de la Terre	Directeur : Mme la Professeure I. DANIEL
UFR Sciences et Techniques des Activités Physiques et Sportives	Directeur : M. C. COLLIGNON
Observatoire des Sciences de l'Univers de Lyon	Directeur : M. B. GUIDERDONI
Polytech Lyon	Directeur : M. P. FOURNIER
Ecole Supérieure de Chimie Physique Electronique	Directeur : M. G. PIGNAULT
Institut Universitaire de Technologie de Lyon 1	Directeur : M. C. VITON
Institut Universitaire de Formation des Maîtres	Directeur : M. A. MOUGNIOTTE
Institut de Science Financière et d'Assurances	Administrateur provisoire : M. N. LEBOISNE

Résumé

La phylogénie permet de reconstituer l'histoire évolutive de séquences ainsi que des espèces qui les portent. Les récents progrès des méthodes de séquençage ont permis une inflation du nombre de séquences disponibles et donc du nombre d'arbres de gènes qu'il est possible de construire. La question qui se pose est alors d'optimiser la recherche d'informations dans ces arbres. Cette recherche doit être à la fois exhaustive et efficace. Pour ce faire, mon travail de thèse a consisté en l'écriture puis en l'utilisation d'un ensemble de programmes capables de parcourir et d'annoter les arbres phylogénétiques. Cet ensemble de programmes porte le nom de TPMS (*Tree Pattern Matching Suite*).

Le premier de ces programmes (`tpms_query`) permet d'effectuer l'interrogation de collections à l'aide d'un formalisme dédié. Les possibilités qu'il offre sont :

La détection de transferts horizontaux : Si un arbre de gènes présente une espèce branchée dans un arbre au milieu d'un groupe monophylétique d'espèces avec lesquelles elle n'est pas apparentée, on peut supposer qu'il s'agit d'un transfert horizontal, si ces organismes sont des procaryotes ou des eucaryotes unicellulaires.

La détection d'orthologie : Si une partie d'un arbre de gènes correspond exactement à l'arbre des espèces, on peut alors supposer que ces gènes sont un ensemble de gènes d'orthologues.

La validation de phylogénies connues : Quand l'arbre des espèces donne lieu à des débats, il peut est possible d'interroger une large collection d'arbres de gènes pour voir combien de familles de gènes correspondent à chaque hypothèse.

Un autre programme, `tpms_computations`, permet d'effectuer des opérations en parallèle sur tous les arbres, et propose notamment l'enracinement automatique des arbres *via* différents critères, ainsi que l'extraction de sous-arbres d'orthologues (séquence unique par espèce). Il propose aussi une méthode de détection automatique d'incongruences.

La thèse présente le contexte, les différents algorithmes à la base de ces programmes, ainsi que plusieurs utilisations qui en ont été faites.

Mots-clés : évolution moléculaire, bioinformatique, phylogénie, tree pattern matching, enracinement, transfert horizontal, orthologie, incongruence.

Abstract

Phylogeny allows to reconstruct evolutionnary history of sequences and species that carry them. Recent progress in sequencing methods produced a growing number of available sequences, and so of number of gene trees that one can build. One of the consecutive issues is to optimise the extraction of information from the trees. Such an extraction should be complete and efficient. To address this, my thesis consisted in writing and then using a suite of programs which aim to browse and annotate phylogenic trees. This program suite is named TPMS (*Tree Pattern Matching Suite*). It browses and annotates trees with several algorithms.

The first of them, `tpms_query` consists in querying collections using a dedicated formalism. This allows to:

Detect horizontal transfers If, in a gene tree, a species is nested in a monophyletic group of unrelated species, one can infer this is a horizontal transfer, if this organisms are prokaryotic (also concerning some unicellular eukaryotes).

Orthology detection If a part of a gene tree exactly matches to the species tree, one can suppose these genes are set of orthologues.

Validating known phylogenies When controversy exists concerning the species tree, it is possible to query a lange collection of gene trees to perform a count of families matching to each hypothesis.

Another program allows to perform parallel operations on all the trees, such as automating rooting of trees *via* different criterions. It also allows an automatic detection of incongruencies.

The thesis introduces the context, different algorithms which the programs are based on, and several using performed with it.

Keywords: evolutionnary biology, bioinformatics, phylogeny, tree pattern matching, rooting, horizontal gene transfer, orthology, incongruency.

Générique de remerciements

Par ordre arbirtraire, mais pas trop :

Le directeur de thèse omniscient Guy Perrière

Les donneurs d'idées géniales Vincent Daubin, Florent Lassalle, Jos Käfer

Ceux qui sont toujours là quand on en a besoin Élodie Gaden, Marie Cariou, Mathieu Groussin, mes parents Hubert et Patricia, mes frère et sœur Étienne et Anna, ma famille en général

Le conseiller spécial en phylogénie Mathieu Groussin

Les relectrices Élodie Gaden (l'agrégée de lettres), Héloïse Philippon (la doctorante candide)

Les supporters précoces Mathieu Groussin, Eugénie Pessia, Yann Lesecque, Floriane Plard

Les collaborateurs Jos Käfer, Eugénie Pessia, Florent Lassale, Mariona Ferrandiz, Aurélie Cohas

Les pourvoyeurs d'avenir Vincent Daubin, Laurent Guéguen

Les collègues actuels ou récents (dans le désordre le plus complet) Erika Kvikstad, Eugénie Pessia, Joanna Parmley, Murray Patterson, Bérénice Batut, Fanny Pouyet, Yann Lesecque, Héloïse Philippon, Magali Semeria, Florent Lassalle, Nicolas Rochette, Floriane Plard

Les collègues de la vieille époque Anamaria Necşulea, Joan Ho-Huu, Claire Guillet, Alexandra Popa, Jean-François Goût, Anne-Sophie Sertier, Anne Lagadic

Le coupable absent Jean Lobry[1]

Moyens techniques les pôles informatique, entretien, administratifs, et la direction des sports du LBBE

[1]La police scientifique est sur le coup.

L'inclassable Sylvain Mousset

Leçon de remerciements de thèse inégalables Vincent Daubin

Les donneurs de responsabilités d'enseignement Raquel Tavares, Vincent Lacroix, Laurent Guéguen, Dominique Mouchiroud

Plus généralement merci à tous ces gens de l'équipe BGE, et du LBBE, laboratoire particulièrement pourvu en personnes bienveillantes, sympathiques, et talentueuses.

Merci à tous ceux qui auraient participé de près ou de loin à l'accomplissement de cette thèse. Et désolé pour ceux qui ne sont pas sur cette page : c'est plus que vraisemblablement[2] un oubli.

[2]Voir section sur la vraisemblance.

Dédié à Henriette Bigot,
ma grand-mère de 88 ans,
qui, si elle n'avait pas eu quinze ans en 1939,
et n'avait pas été une jeune fille d'une famille modeste de cinq enfants,
aurait probablement fait de brillantes études supérieures,
quoi qu'elle puisse en penser.

Table des matières

1

Introduction

Le terme *bioinformatique*[1] est un mot-valise mettant en jeu la biologie et l'informatique. Dans ce mot français, le terme *informatique* peut, selon les acceptions, être un faux ami de l'anglais *informatics*, *informatique* se disant *computer science* dans cette langue. En effet, les anglophones distinguent *bioinformatics* de *biocomputing*. Tandis que le premier concerne l'étude de la bioinformation (séquences nucléiques et protéiques) — c'est du mot *information* que vient le terme *informatics* —, le second concerne plus généralement l'utilisation des outils informatiques pour répondre à des questions biologiques. La langue française ne consacre pas cette nuance et emploie indifféremment le terme bioinformatique pour l'une ou l'autre de ces approches. Il sera question ici de la bioinformatique des séquences, et en l'occurrence d'une approche évolutive de ces séquences.

La bioinformatique est née des progrès formidables et rapides dans les deux domaines durant la seconde moitié du siècle dernier : augmentation des performances, communication et reproduction des données sans coût du côté de l'informatique, et amélioration des techniques de séquençage pour la biologie.

Dans une approche naïve, on peut penser que l'informatique est un soutien, une base technique, des outils mis à la disposition de la biologie pour répondre à une question posée par la biologie. Le bioinformaticien est alors un technicien, un ingénieur

[1] *Bioinformatique* peut s'écrire avec ou sans trait-d'union entre *bio* et *informatique* : la langue française voudrait qu'on place ce trait-d'union pour des raisons de prononciation, mais l'usage plébicite l'inverse.

qui connaît des méthodes et les adapte aux problèmes de la biologie. Mais en réalité, le biologiste moléculaire et le bioinformaticien sont la même personne ou de très proches collègues. Dans cette optique-là, la bioinformatique n'a pas seulement un rôle de support, mais de théorisation et de questionnement de la biologie au travers de l'information génétique. Elle a des champs de réflexion qui lui sont propres. Elle tente de construire des modèles qui décrivent au mieux les processus évolutifs.

Bien que le terme *bioinformatique* soit apparu récemment, cette discipline complète les approches précédentes de théorisation de la biologie ou leur succède : la biométrie, et les biomathématiques. Cette dernière a commencé par traiter des problématiques de modélisation des interactions Proie-Prédateur (définie indépendemment par Lotka et Volterra [126]), de la population (Pierre-François Verhulst en 1838 [103]). Plus récemment, ce champ disciplinaire s'est intéressé aux problèmes émergents de phylogénie moléculaire, dont il est particulièrement question dans cette thèse.

L'apparition de la phylogénie moléculaire remonte aux années 1960 et correspond chronologiquement aux premiers séquençages protéiques puis nucléiques. Depuis, de nombreux progrès ont permis de tirer plus de signification des données et de comprendre les mécanismes d'évolution des séquences. D'autre part, les séquences sont de plus en plus nombreuses, tant en nombre brut qu'en terme d'espèces couvertes.

L'enjeu central de la phylogénie est de produire des arbres qui décrivent la parenté entre différents organismes ou différentes séquences. Il existe de nombreuses méthodes qui permettent de construire des arbres en partant de séquences. Toutes ont pour but d'inférer au mieux l'histoire évolutive portée par ces séquences et les progrès ne se sont pas interrompus depuis l'avènement de cette discipline.

Les arbres produits sont porteurs d'informations explicites : la position relative des entités analysées (séquences ou espèces), et très souvent, la distance qu'il y a entre ces séquences. Mais certains évènements évolutifs restent à interpréter comme par exemple les duplications et les transferts horizontaux. Donc en plus des énormes efforts effectués pour faire de bons arbres, pour en tirer toute la signification qu'ils portent, il faut parvenir à en extraire cette information implicite. Ce qui peut être fait par un humain en lisant quelques arbres ne peut pas l'être pour des milliers voire des dizaines de milliers d'arbres. Il faut trouver des moyens d'automatiser l'exploitation et l'analyse des données qui se trouvent cachées dans ces arbres. C'est dans ce contexte

que j'ai écrit TPMS[2]. Cette suite de programmes tourne autour du concept *pattern matching*. Trouver des motifs particuliers qui ont une signification biologique, et raciner les arbres sont deux piliers de cette suite.

Tout ceci a été pensé dans une optique d'efficacité des algorithmes utilisés. Interroger massivement des milliers d'arbres ne peut pas se faire sans penser que chaque opération qui sera exécutée des milliers voire des millions de fois doit être la plus efficace possible. Les informations auxquelles on accède souvent doivent être indexées, les résultats intermédiaires doivent être stockés s'ils sont amenés à être réutilisés. En plus de tenter de répondre aux questions bioinformatiques posées (enracinement automatique, recherche de motifs), le programme développé se veut une base logicielle pour traiter les arbres phylogénétiques après leur construction.

Après une présentation du contexte de la phylogénie moléculaire et des collections de familles de gènes, je présenterai l'outil développé. J'expliquerai la façon dont les arbres étudiés sont et traités, avec quelles fonctions ils sont explorés. Je présenterai ensuite plusieurs algorithmes pour raciner les arbres, et opérer une détection systématique des incongruences. Enfin, je montrerai les utilisations qui ont été faites de cet outil dont certaines ont été réalisées en collaboration.

[2] *Tree Pattern Matching Suite*, suite [logicielle] de recherche de motifs d'arbres.

2 Contexte bioinformatique

La bioinformatique est un champ disciplinaire large qui va de la prédiction des structures et des fonctions à partir de séquences, à la modélisation de réseaux protéiques, en passant évidemment par la phylogénie. Les sections qui suivent sont centrées sur cette dernière, bien qu'il sera question globalement d'évolution moléculaire.

2.1 Les arbres phylogénétiques

2.1.1 Introduction

La phylogénie est une des disciplines de la biologie évolutive. Elle consiste à établir les liens de parenté entre des espèces, vivantes ou disparues. Les espèces et leurs relations de parenté peuvent être représentées sous la forme d'un arbre phylogénétique. La phylogénie a longtemps utilisé des caractères morphologiques pour établir ces liens de parenté. Cette approche a l'avantage d'être simple et accessible à tout observateur averti, mais a plusieurs limites. La première d'entre elles est qu'une ressemblance phénotypique peut être la conséquence d'une convergence évolutive. Par exemple, la présence d'une aile pour voler chez l'oiseau et chez la chauve-souris sont issues de deux innovations évolutives indépendantes : elles ne peuvent donc pas servir de point de départ à l'établissement d'une phylogénie, car il s'agit de deux histoires distinctes. Une deuxième limite est que tous les organismes ne sont pas observables aussi directement que les plantes et les animaux : leur faible taille implique obligatoirement des disposi-

tifs de grossissement. La grande variété microbienne est difficile à décrire et à discerner sur les seuls critères morphologiques : on a alors un faible pouvoir résolutif, c'est-à-dire une faible capacité à discerner des espèces. On peut constater une troisième limite qui concerne les espèces disparues. Un moyen d'établir des caractéristiques de telles espèces est d'étudier les fossiles qu'elles ont pu laisser. Mais cette approche a comme limites que deux espèces bien différentes peuvent laisser des empruntes fossiles très proches, indiscernables, quand elles en laissent : de nombreuses espèces n'engendrent aucun fossile.

L'utilisation de l'information moléculaire a permis de surmonter ces difficultés, tout en présentant de nouveaux défis. Par information moléculaire, on entend les acides nucléiques que sont l'ADN et l'ARN, ainsi que les chaînes peptidiques constituées d'acides aminés. Utiliser les séquences biologiques présente plusieurs avantages sur les caractères morphologiques :

Subjectivité : Les critères morphologiques à observer doivent être choisis, leurs modalités possibles déterminées, et ces critères peuvent être très subjectifs, et reflètent essentiellement la façon dont l'expérimentateur perçoit les mécanismes évolutifs. Alors qu'*a contrario*, la phylogénie moléculaire n'implique que peu de choix subjectifs, si ce n'est, évidemment, le choix des séquences utilisées.

Manque de robustesse aux erreurs de mesure : Les critères morphologiques sont pour certains des mesures. Or, les mesures sont sujettes à des erreurs. Les séquences aussi sont sujettes à des erreurs, mais elles n'ont pas du tout les mêmes conséquences. Les séquences sont un encodage du vivant. Dans une séquence nucléique, chaque position ne peut prendre que quatre valeurs : A, C, G ou T. La théorie de l'information nous indique que plus le nombre d'états est faible, plus la probabilité de voir le signal dégradé est faible, mais plus le message est long[1] [113].

Limites de la méthode de reconstruction : Les arbres basés sur des caractères morphologiques sont le plus souvent reconstruits en utilisant une méthode de par-

[1]On peut grossièrement comparer l'encodage des caractères du vivant par des acides nucléiques à l'encodage de la musique et l'image, analogiques par essence, en numérique, par le codage binaire : seulement deux états sont possibles. Le passage au disque optique ou au fichier informatique a rendu la musique et la vidéo bien plus difficilement altérables que les cassettes ou les disques vinyles.

cimonie (même si des méthodes pour établir des distances à partir de caractères morphologiques existent [60]). Cette méthode a pour défaut de supposer que si un caractère présente un certain état dans deux espèces, alors ce caractère chez leur ancêtre commun présentait ce même état, ce qui est parfois faux[2].

Quoi qu'il en soit, la phylogénie utilisant des caractères morphologiques est encore utilisée : par exemple, début 2013, O'Leary et collaborateurs ont publié une étude [95] portant sur 4 541 caractères morphologiques, ainsi que 27 gènes. Cette étude tente de répondre à la controverse quant à savoir si les mammifères placentaires sont apparus avant ou après l'extinction Crétacé-Tertiaire[3]. Elle vient contredire les conclusions d'études basées sur des données moléculaires [84, 88, 116] qui concluaient à une apparition de ces placentaires *antérieure* à l'extinction Crétacé-Tertiaire.

Ajourd'hui, deux écoles de pensées s'affrontent ou du moins se défient. D'un côté, les biologistes moléculaires pensent que les séquences disponibles sont de bien meilleurs marqueurs que les mesures morphologiques, pour les raisons évoquées plus haut. De l'autre, des morphologistes arguent que l'observation des structures anatomiques sont bien plus facilement appréhendables et ancrées dans le réel que de cryptiques séquences analysées à l'aide de modèles mathématiques. Ici, nous nous intéresserons à la reconstruction d'histoires évolutives de séquences, et donc de phylogénie moléculaire, et donc à la production d'arbres phylogénétiques.

Construire — ou *re*construire, si on se place dans l'optique où l'arbre vrai existe et peut être approché déjà théoriquement — un arbre se fait en plusieurs étapes. Chacune de ces étapes doit être effectuée avec un esprit critique solide, et une bonne connaissance des options disponibles. Suivre un protocole standard sans en interroger le sens des paramètres présente un risque de passer à côté des hypothèses sous-jacentes, et donc de faire des erreurs.

2.1.2 Choix des séquences, concept d'homologie

La première étape d'une phylogénie moléculaire consiste à trouver des séquences homologues, c'est-à-dire des séquences qui ont une origine ancestrale commune. La phy-

[2]Voir 2.1.6.1.

[3]Il y a 65,5 millions d'années, un nombre très important d'espèces ont disparu. Cet évènement marque la fin de la période géologique *Crétacé* et le début de l'époque *Tertiaire* ou plus exactement *Paléogène*. Cet évènement est couramment désigné comme l'extinction des dinosaures.

logénie moléculaire vise à établir des relations d'apparentement de séquences, rassembler celles qui se ressemblent, voire établir des distances évolutives entre elles. Cela ne peut se faire qu'en observant les effets de l'évolution sur un marqueur qui à l'origine était unique. Le concept d'homologie est à la base de la phylogénie.

Il n'est pas possible de savoir, *a priori*, si des séquences sont homologues. On peut néanmoins en avoir une très forte présomption en se basant sur la similarité desdites séquences. On peut avoir une idée de la similarité de séquences après alignement (voir section 2.1.4).

Pour choisir les séquences à utiliser, il faut avoir une idée de l'échelle de temps que l'arbre va couvrir. En effet, les séquences n'évoluent pas toutes à la même vitesse ; choisir des séquences qui évoluent très vite pour chercher à comprendre une histoire ancienne peut aboutir à une *saturation du signal* phylogénétique. Les séquences qui évoluent vite pendant un temps long finissent par être tellement différentes qu'il est difficile de leur trouver des ressemblances.

Les conséquences de l'évolution sont moins visibles sur les séquences protéiques (le code génétique étant dégénéré[4]) que sur les séquences nucléiques correspondantes. Donc les séquences protéiques sont plus stables dans le temps, par nature. Les protéines sont contraintes : un changement d'un acide aminé peut modifier leur structure, et en conséquence peut potentiellement modifier leur fonction. Alors que le changement d'un nucléotide de la séquence codant pour une protéine peut très bien n'avoir aucune conséquence — dans le cas d'une mutation silencieuse[5]. Au sein des séquences d'acides nucléiques, il y a également de grandes différences de vitesses d'évolution. Les séquences non codantes évoluent plus vite que les séquences codantes, l'ADN mitochondrial plus lentement que l'ADN nucléique.

L'ADN viral a un taux d'évolution généralement très élevé et permet donc de résoudre des histoires récentes. Il a par exemple été utilisé par Biek et collaborateurs [10] pour démontrer la structuration en populations de cougars (*Puma concolor*) dont une partie étaient porteurs du virus d'immunodéficience félin (FIV). Comme leur répartition géographique actuelle était connue comme récente (une centaine d'années), il

[4]Il existe de un à six triplets différents pour coder chaque acide aminé d'une séquence protéique.

[5]Une mutation silencieuse est un changement de nucléotide dans une séquence condant pour une protéine qui n'implique aucune modification d'acide aminé dans la protéine codée. En effet, le code génétique étant dégénéré, un certain nombre d'acides aminés sont codés par plusieurs codons. On les appelle codons synonymes.

était inenvisageable de se baser sur leurs propres génomes. Les seules séquences dont ils étaient porteurs et qui avaient un taux d'évolution assez rapide pour que l'accumulation de mutations donne un signal significatif étaient celles du FIV. Les séquences de FIV ont été utilisées pour reconstruire une phylogénie qui a groupé les séquences en huit lignées, correspondant à des aires géographiques bien distinctes.

2.1.3 Orthologues *vs.* Paralogues

Il existe différents types d'homologues : les orthologues et les paralogues qui sont respectivement reliés aux concepts de spéciation et de duplication.

2.1.3.1 Définitions

Lorsque dans un jeu de gènes, aucune des séquences n'a subi un transfert ou une duplication, on a affaire à des gènes dits **orthologues**. Quand les évènements qui ont généré ces gènes sont exclusivement des spéciations, on parle de gènes orthologues, qu'Eugene Koonin définit comme « des gènes qui dérivent d'un gène ancestral unique dans le dernier ancêtre commun aux espèces considérées » [65]. Selon lui, la définition ne fait pas intervenir une similitude de fonction des gènes.

A contrario, quand des gènes ont comme lien un évènement de duplication, on parle de gènes **paralogues**. Les séquences n'ont pas à être situées dans le même génome. Peu importe la date de la duplication (des duplications ancestrales ont donné des gènes qu'on nomme paralogues).

Comme l'histoire des gènes orthologues correspond parfaitement à l'histoire des espèces, on cherche souvent à trouver des jeux de gènes orthologues. En effet, en connaissant l'histoire évolutive d'un marqueur unique par espèce, on peut en inférer directement l'histoire des espèces. Il existe principalement deux méthodes, la première basée sur la similarité, et la seconde qui utilise la phylogénie.

2.1.3.2 Annotation des gènes : orthologues ou paralogues ?

Plusieurs méthodes existent pour discerner les orthologues des paralogues [65]. *Symmetrical Bests Hits* est très simple d'approche, ce qui en fait l'une des méthodes les plus utilisées. Elle permet de trouver un ou plusieurs gènes orthologues à une séquence

d'intérêt, parmi un jeu de séquences, qui peut être une banque de données généraliste. Elle utilise la similarité : on cherche, parmi les n séquences de notre jeu, celle ou celles qui ressemblent le plus à la séquence d'intérêt (on utilise généralement BLAST pour cette opération, voir 2.3.2). À ce stade, on dispose de séquences dont la ressemblance au gène d'intérêt nous laisse à penser qu'il s'agit de gènes homologues. Dans une deuxième étape, on fait l'opération symétrique, c'est-à-dire faire une recherche BLAST des séquences trouvées précédemment et vérifier que le meilleur résultat est bien la séquence soumise.

Cette méthode fait alors l'hypothèse que les séquences protéiques des gènes orthologues sont plus similaires entre elles qu'elles le sont des autres gènes des génomes comparés. Cela semble cohérent avec l'hypothèse que des gènes orthologues remplissent la même fonction chez différentes espèces, et que donc leur évolution est contrainte par cette fonction : si un gène évolue trop et s'écarte de la séquence de ses orthologues, la fonction qu'il couvre risque de ne plus être remplie. On peut supposer qu'un écart aux autres séquences serait réprimé par une sélection purificatrice[6].

Le Symetrical Best Hits, ou *SymBet*, a été massivement utilisé pour inférer paralogues et orthologues chez *Haemophilus influenzae* à partir des annotations de gènes de *Escherichia coli* [121]. Cette technique a initialement été utilisée pour annoter la banque d'homologues COG [120].

Mais cette méthode présente quelques limites et ne donne pas toujours le meilleur résultat [65]. D'autres méthodes existent : elles se basent sur la phylogénie et sont abordées plus loin (voir 2.2.6).

2.1.4 Alignement des séquences

Les méthodes de phylogénie se basent pour la plupart sur la comparaison individuelle des sites des différentes séquences étudiées. Si l'évolution n'était faite que de substitutions, l'étape de l'alignement ne serait pas nécessaire : la séquence ancestrale et les séquences étudiées auraient toutes la même longueur. À une position donnée (n-ième acide aminé ou n-ième nucléotide en partant de la première position), le résidu présent dans chaque séquence étudiée serait issu du même résidu dans la séquence ancestrale.

[6]La sélection purificatrice est le phénomène par lequel si aléatoirement une séquence évolue de manière à générer une moins grande propension à la survie et à la reproduction de son hôte, alors cette séquence ne sera pas promue et tendra à disparaître. [76]

Or, les séquences subissent, au cours de leur évolution, des insertions et des délétions, ce qui décale les séquences et leur donne des longeurs qui peuvent être très différentes. De plus, il n'est pas possible de savoir directement à quel site d'une séquence correspond un site particulier d'une autre séquence homologue.

L'étape de l'alignement des séquences consiste à insérer des brèches au sein des différentes séquences afin que les blocs similaires se retrouvent superposés. On espère ainsi obtenir, dans une même colonne si les séquences sont en ligne, des sites homologues en regard. Ce type d'alignement est dit d'*alignement multiple* car il vise à aligner les sites appartenant à plus de deux séquences (la figure 2.1 donne un exemple d'alignement multiple de séquences protéiques). Il s'agit de minimiser les évènements du scénario qui explique les différences de position entre les séquences. Pour ce faire, chaque évènement, de substitution, d'insertion ou de délétion, est associé à une pénalité.

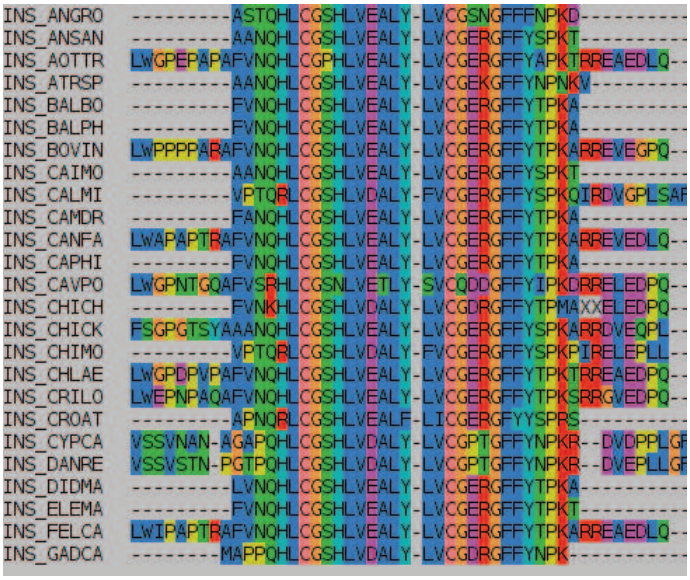


FIGURE 2.1 – **Alignement partiel de séquences protéiques d'insulines issues de différentes espèces de mammifères**, généré par MUSCLE, et visualisé à l'aide du programme SeaView [44]. Chaque lettre correspond à un acide aminé [21, 74]. Chaque couleur correspond à une famille d'acides aminés aux propriétés physico-chimiques identiques (caractère hydrophobe / hydrophile, propriétés acides / basiques, présence d'un cycle aromatique). Les tirets (-) représentent l'insertion, par le processus d'alignement, d'un vide au sein d'une séquence.

L'algorithme exact pour réaliser cette opération présente une complexité de $O(L^N)$

pour aligner N séquences de longueur L, ce qui rend la tâche irréalisable en un temps raisonnable, dès que le nombre de séquences dépasse une demi-douzaine. Comme souvent, dans ce cas, on utilise une heuristique, c'est-à-dire une méthode inexacte, plus rapide, mais dont le produit est une bonne approximation de la réalité. Dans la plupart des méthodes d'alignement multiple actuelles, l'heuristique utilisée s'appelle l'*alignement progressif* et a été introduit par Hogeweg et Hesper en 1984 [52]. L'alignement est découpé en sous-alignements qui sont réalisés en suivant un *arbre guide*, ce qui réduit la complexité du problème en $O(N^2)$. Il existe plusieurs implémentations : T-Coffee [92], MUSCLE [29, 30], Clustal [123, 67, 114].

2.1.5 Détermination des distances — Modèles d'évolution

Une fois que les séquences sont alignées entre elles, les sites homologues sont censés se trouver sur la même colonne. Dès lors, il est possible, en comparant la composition de chaque colonne — on pense donc que chaque colonne reflète l'histoire de l'évolution d'un résidu unique ancestral —, d'inférer une distance évolutive entre les séquences deux à deux. Le calcul de ces distances est réalisé au moyen de modèles d'évolution. Ces modèles sont utilisés par les méthodes de distances (*neighbor-joining*, moindres carrés, minimum d'évolution, classification ascendante hiérarchique), de maximum de vraisemblance ou bayésiennes. Toutes ces méthodes produisent des arbres ayant des longueurs de branche : une fois la topologie de l'arbre reconstruite, chaque branche a une longueur définie. Cela signifie qu'on est capable de déterminer la quantité d'évolution entre chaque séquence et sa séquence ancestrale dans l'arbre.

Il s'agit à ce stade d'établir les distances deux à deux entre toutes les séquences de l'étude. Ce calcul des distances peut, en première intention, se faire en comptant les différences entre deux séquences alignées. Cela s'appelle la *divergence observée*. Ce comptage n'est pas très informatif du fait des substitutions cachées. En effet, nous n'avons à notre disposition que les séquences actuelles, et nous pouvons supposer que lorsqu'on peut compter, entre deux séquences, v substitutions visibles, c'est qu'il y a eu r substitutions réelles au cours de l'histoire évolutive qui relie ces deux séquences. Or, en réalité, r ne peut qu'être supérieur à v : il peut y avoir eu sur un même site, plusieurs substitutions successives, et nous n'en verrions que la dernière.

Les modèles évolutifs sont utilisés pour estimer la quantité de ces substitutions ca-

chées. Ce sont des modèles de Markov [91] qui décrivent l'évolution d'un site dans une séquence au cours du temps. Les différents états de ces modèles sont les différents résidus possibles à chaque position sur une séquence (pour chaque site, 20 états possibles pour les séquences protéiques et 4 pour les séquences nucléiques), et les transitions correspondent aux différentes substitutions qui interviennent sur ces séquences au cours du temps. Les utiliser suppose que l'évolution d'une séquence est un processus markovien. Les modèles de Markov standards fonctionnent par étapes, mais ceux utilisés pour la phylogénie sont dits en temps continu : les étapes ont une durée infinitésimale.

Définir un modèle évolutif consiste tout simplement à établir la matrice de transition $P(t)$ d'un processus de Markov contenant des probabilités de transitions.

2.1.5.1 Modèles nucléiques

Pour les séquences nucléiques, cette matrice est de la forme :

$$P(t) = \begin{bmatrix} p_{AA}(t) & p_{AC}(t) & p_{AG}(t) & p_{AT}(t) \\ p_{CA}(t) & p_{CC}(t) & p_{CG}(t) & p_{CT}(t) \\ p_{GA}(t) & p_{GC}(t) & p_{GG}(t) & p_{GT}(t) \\ p_{TA}(t) & p_{TC}(t) & p_{TG}(t) & p_{TT}(t) \end{bmatrix}$$

avec $p_{ij}(t)$ la probabilité que, pour une position considérée, le nucléotide i soit substitué par un nucléotide j au cours du temps t, et $p_{ii}(t)$ la probabilité que ce nucléotide reste identique sur cette position.

La matrice $P(t)$ est elle-même obtenue à partir d'une autre matrice Q qui définit quant à elle les taux de substitutions instantanés, c'est-à-dire pour une séquence, le flux des substitutions. Elle est défini comme :

$$Q = \begin{bmatrix} q_{AA} & q_{AC} & q_{AG} & q_{AT} \\ q_{CA} & q_{CC} & q_{CG} & q_{CT} \\ q_{GA} & q_{GC} & q_{GG} & q_{GT} \\ q_{TA} & q_{TC} & q_{TG} & q_{TT} \end{bmatrix}$$

avec q_{ij} le taux de nucléotides de type i qui vont être substitués par un nucléo-

tide de type j au cours d'un temps infinitésimal dt. Cette matrice Q est à la base du modèle : elle décrit les modifications au sein d'une séquence selon les hypothèses qu'il fait.

Par définition, il est possible de passer de $P(t)$ à Q à l'aide de la relation :

$$P(t) = e^{Qt}$$

Il existe de nombreux modèles qui permettent de construire cette matrice. Le plus simple de ces modèles est celui de Jukes et Cantor [61], introduit en 1969. La matrice Q de ce modèle est ainsi définie :

$$Q = \begin{bmatrix} -3\alpha & \alpha & \alpha & \alpha \\ \alpha & -3\alpha & \alpha & \alpha \\ \alpha & \alpha & -3\alpha & \alpha \\ \alpha & \alpha & \alpha & -3\alpha \end{bmatrix}$$

Il fait l'hypothèse que les taux de substitution sont équiprobables ($q_{ij} = \alpha, \forall i \neq j$). Une conséquence de cette hypothèse est que les fréquences en bases à l'équilibre[7] sont égales entre elles, et égales à un quart : $\pi_A = \pi_C = \pi_G = \pi_T = \frac{1}{4}$. Il n'a qu'un seul paramètre, c'est le taux de substitution général α. Comme pour tout processus markovien, on peut obtenir les probabilités de transitions. Dans le cas du modèle Jukes et Cantor, les probabilités de transitions peuvent se calculer analytiquement. Elles sont les suivantes : $p_{ij}(t) = \frac{1}{4} - \frac{1}{4}e^{-4t\alpha}$ $(\forall i \neq j)$ et $p_{ii}(t) = \frac{1}{4} + \frac{3}{4}e^{-4t\alpha}$, à replacer dans la matrice $P(t)$ vue *supra*. On a donc bien obtenu, avec un seul paramètre, toutes les probabilités de transitions selon le temps.

Soit $p(t)$ la probabilité de substitution en un temps t, définie dans la matrice P *supra* par $p_{ij}(t) \forall i \neq j$. La probabilité p d'observer une substitution à un site (divergence observée) est définie par $p = 3p(2t)$, et la distance $d = 3\alpha 2t = 6\alpha t$ [8]. On peut

[7] Dans un processus de Markov, l'état d'équilibre est caractérisé par le fait que les fréquences ne changent plus. Dans le cas présent des modèles de distance, cela signifie que les fréquences des bases restent les mêmes.

[8] Explication des opérandes :

- $2t$ car la distance entre deux séquences est égale à la somme de la distance qui sépare la première de leur ancêtre commun, et de la seconde à ce même ancêtre, les deux séquences étant situées à

en déduire la distance en fonction du nombre de substitutions : $d = -\frac{3}{4}\ln\left(1-\frac{4}{3}p\right)$. *In fine* on est capable d'associer des distances évolutives à chaque paire de séquences, à partir de la divergence observée.

Il existe de nombreux autres modèles qui permettent de construire différentes versions de $P(t)$, ils ont plusieurs paramètres. Pour certains, on ne peut pas déterminer $P(t)$ de manière analytique, c'est-à-dire exacte, on ne peut le faire que numériquement, ce qui revient à faire une approximation. Les différents paramètres supposent des hypothèses différentes selon les modèles. Par exemple, le modèle de Kimura à deux paramètres [63] suppose que les transversions (passage d'une base purique à une base pyrimidique ou inversement) ne surviennent pas avec la même fréquence que les transitions (purine vers purine ou pyrimidine vers pyrimidine).

2.1.5.2 Modèles protéiques

Pour les séquences protéiques, il faut définir une matrice $P(t)$ de la même manière que précédemment, à la différence que cette matrice est de taille 20×20. Cela a pour conséquence qu'il y a un très grand nombre de paramètres impossibles à estimer à partir de deux séquences seulement. Il existe donc un certain nombre de matrices préétablies qui ont été déterminées en utilisant différents jeux de séquences protéiques. De la même manière, avec ces matrices de probabilités de transitions, on est capable de déterminer des distances évolutives.

Il existe différents modèles, les plus connus étant PAM [23], JTT [59], et WAG [129]. Ces modèles diffèrent par le nombre de séquences utilisées, les méthodes utilisées pour estimer les matrices, ainsi que les hypothèses biologiques sous-jacentes.

2.1.6 Reconstruction de la topologie — Méthodes classiques

L'étape centrale de la reconstruction d'un arbre est la reconstruction de sa topologie[9] : il s'agit, dans un premier temps, d'établir un arbre non raciné, c'est-à-dire un

un temps t de cet ancêtre ;

- 3α car il y a trois façons de changer d'état pour un nucléotide, étant donné qu'il y a en tout quatre états possibles.

[9]Il existe des méthodes où établissement des distances et reconstruction de la topologie sont deux étapes effectuées en simultané (*neighbor-joining*, UPGMA — Unweighted Pair Group Method with

graphe acyclique, avant de le raciner (voir 2.1.8) pour en faire ainsi un graphe acyclique orienté. Cette étape constiste en substance à relier entre elles les séquences analysées, ainsi que les nœuds internes représentant des séquences ancestrales, par des branches représentant une certaine quantité d'évolution (cette quantité d'évolution n'est pas toujours évaluée, selon la méthode utilisée).

La reconstruction topologique produit un arbre binaire : quel que soit le sens dans lequel on le parcourt, la rencontre d'un nœud aboutit sur deux branches qui aboutissent chacune sur un autre nœud. Cavalli-Sforza et Edwards ont établi [18] qu'il existait, pour n séquences, $\frac{(2n-3)!}{2^{n-3}(n-3)!}$ arbres non racinés possibles. Cela signifie qu'il y a par exemple grossièrement autant d'arbres possibles avec 20 séquences que de grains de sables dans le Sahara[10].

Il existe différentes méthodes pour reconstruire des arbres. Aujourd'hui, les plus communes sont au nombre de quatre : maximum de parcimonie, *neighbor-joining*, maximum de vraisemblance, et approche bayésienne.

2.1.6.1 Maximum de parcimonie

Comme souvent, dans la démarche scientifique, c'est le principe du rasoir d'Ockham[11] qui est utilisé : on cherche à établir le ou les scénarios qui impliquent le moins d'évènements évolutifs.

Le principe est le suivant : pour chaque site, pour une topologie donnée, connaissant les états actuels pour chaque séquence (c'est-à-dire les résidus présents sur un même site respectif de chaque séquence de l'alignement), on compte les substitutions nécessaires sur l'arbre pour obtenir cette topologie avec ces états. La figure 2.2 illustre le concept du comptage des substitutions nécessaires par site.

Une telle opération est facile à appréhender, mais elle doit également être automatisable. C'est Fitch en 1971 qui a formalisé [37] l'algorithme qui depuis porte son nom. Il s'agit d'un algorithme récursif, dont le cas de base est la feuille, et le cas récursif tout nœud interne (voir partie 3.1.5). Pour les besoins de la récursivité, une racine doit être placée dans l'arbre. Elle peut être placée sur n'importe quelle branche de l'arbre, elle

Arithmetic Mean—, Fitch-Margoliash).

[10] En prenant une superficie de 1000 km par 1000 km sur une profondeur de 50 m, on trouve grossièrement 10^{23} grains de sable de 0,1 mm de rayon [130].

[11] Initialement formulé par Aristote, puis Ptolémée [39] (vers 90 - vers 168) : « Nous considérons comme un bon principe d'expliquer le phénomène par les hypothèses les plus simples. »

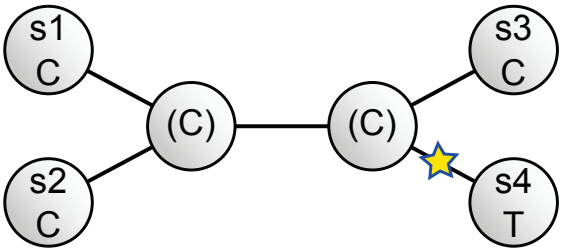


FIGURE 2.2 – **Exemple de comptage de substitutions nécessaire.** Quatre séquences sont ici étudiées. Sur un site donné, trois séquences, s_1, s_2 et s_3 portent le résidu C et la séquence s_4 le résidu T. L'arbre de cet exemple représente une des topologies possibles. Les nœuds internes portent, entre parenthèses, les résidus possibles à ces nœuds. L'étoile indique un évènement de substitution nécessaire pour cette topologie. Pour ces quatre séquences et pour le site en question, le nombre de substitutions nécessaires est 1.

n'a aucune signification phylogénétique. Des feuilles vers cette racine sont calculés, pour chaque sous-arbre dont le nœud p est la racine i) une liste d'états (résidus) possibles à ce nœud, qu'on note S_p ii) un nombre de changements nécessaires à ce nœud notés c_p.

À chaque nœud, on observe les états possibles aux nœuds fils. Deux cas possibles :

- si $S_q \cap S_r \neq \varnothing$ alors
  - $S_p \leftarrow S_q \cap S_r$
  - $c_p \leftarrow c_q + c_r$
- sinon
  - $S_p \leftarrow S_q \cup S_r$
  - $c_p \leftarrow c_q + c_r + 1$

La figure 2.3 montre un exemple de réalisation de l'algorithme de Fitch pour le calcul récursif du nombre de substitutions nécessaires pour une topologie donnée.

Cette opération est effectuée pour chaque site de l'alignement : la ou les topologies nécessitant le nombre de substitutions le plus faible est/sont choisie(s). Cette méthode peut produire plusieurs arbres qui nécessitent chacun le même nombre minimum d'évènements de substitution. Ces arbres sont dits *équiparcimonieux*. Une solution

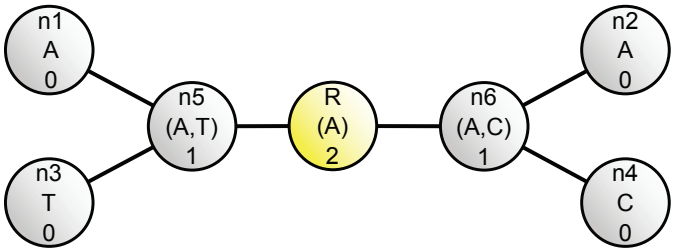


FIGURE 2.3 – **Calcul du nombre de substitutions nécessaires pour une topologie donnée sur un site donné, en utilisant l'algorithme de Fitch.** On étudie ici la phylogénie de quatre séquences. Sur le site considéré, les séquences n_1 et n_2 portent le résidu A, la séquence n_3 le résidu T et n_4 porte C. Le nombre sur la troisième ligne indique le nombre de substitutions nécessaire, et sur la deuxième ligne des nœuds internes sont reportés les états possibles. Le nœud annoté R (en jaune) est la racine virtuelle. Le nombre de substitutions calculées pour cette racine, et donc pour cette topologie, est de 2.

peut être de donner à l'utilisateur un arbre consensus qui synthétise un branchement conflictuel entre deux topologies équiparcimonieuses par une multifurcation.

La parcimonie est une méthode facilement compréhensible, et elle produit des arbres acceptables dans une première approche. Mais elle est naïve en considérant que si deux espèces ont un caractère identique, alors leur ancêtre commun présentait ce même caractère identique. Cela ne fonctionne pas dans certains cas, notamment quand certaines espèces ont un taux d'évolution très différent de la majorité. De même, par conception, le raisonnement par parcimonie n'est pas capable de différencier homoplasie et homologie. En effet, quand deux caractères sont identiques chez deux espèces, on ne peut pas savoir si c'est parce que ce caractère a été hérité de leur ancêtre commun, ou bien si ce caractère est identique pour d'autres raisons (convergence, parallélisme, ou réversion). Comme la parcimonie ne fait qu'observer les états changeants, elle peut inférer une identité à tort. La parcimonie est un raisonnement d'économie. L'étape suivante fut d'élaborer des modèles de l'évolution des séquences. Les méthodes suivantes, *neighbor-joining*, maximum de vraisemblance, et approche bayesienne utilisent des modèles évolutifs.

2.1.6.2 *Neighbor-joining*

Le *neighbor-joining* est une méthode, appartenant à la famille des méthodes de distances, qui permet d'approximer le minimum d'évolution [110, 109, 42]. L'arbre du minimum d'évolution est celui des arbres possibles dont la somme des longueurs de branches est la plus faible. Pour l'obtenir, il faut, pour toutes les topologies possibles, déterminer les longueurs des branches par la méthode des moindres carrés, et ne conserver que l'arbre ou les arbres dont la somme est la plus faible. Cette tâche n'étant pas possible dans un temps raisonnable lorsque le nombre de taxons à traiter dépasse la dizaine, le *neighbor-joining* en est la meilleure approximation.

Dans un premier temps, la méthode utilise une matrice de distance entre les séquences deux à deux. Cette distance est le plus souvent obtenue grâce à un modèle d'évolution (voir 2.1.5). Une fois que les distances sont établies entre les séquences, la topologie doit être construite. Le procédé consiste à partir d'une topologie en étoile, c'est-à-dire un seul nœud interne relié à toutes les feuilles, puis d'identifier les deux séquences qui, une fois regroupées permettront d'obtenir l'arbre dont la somme des longueurs de branches est la plus faible. Le nouveau nœud qui rejoint ces deux séquences est lui-même relié au nœud central. Petit à petit, une topologie binaire se construit en reliant les nœuds et les feuilles les plus proches. La figure 2.4 donne un exemple d'une telle construction.

À chaque étape de ce processus, la distance entre deux séquences dans l'arbre ne peux pas refléter exactement la distance obtenue lors de la détermination des paires de distances séquence à séquence. L'algorithme doit faire en sorte que la topologie minimise les écarts à ces distances théoriques.

2.1.6.3 Maximum de vraisemblance

Le maximum de vraisemblance est, en statistique, une méthode permettant d'estimer les paramètres d'une distribution. Cette méthode a été conçue par Fisher [3] en 1912. Elle part de mesures de valeurs que prend une variable aléatoire, ainsi que d'une loi de probabilité donnée, pour déterminer avec quels paramètres cette loi est la plus susceptible de produire les valeurs rencontrées dans l'échantillon. Pour chaque loi de distribution, il peut y avoir une, plusieurs ou zéro fonctions de vraisemblance, c'est-à-dire une fonction qui associe la probabilité d'obtenir les données observées à un ensemble

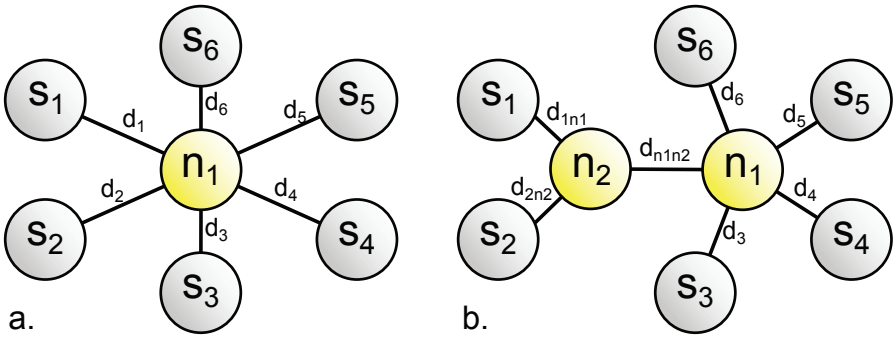


FIGURE 2.4 – ***Neighbor-joining* sur un arbre à six feuilles a.** L'arbre initial est en étoile : les distances sont établies de telle sorte qu'elles reflètent les distances deux à deux calculées préalablement. **b.** Première étape du *neighbor-joining* : en reliant par un nouveau nœud s_1 et s_2, on obtient l'arbre qui présente la somme des longueurs de branches la plus faible, s_1 et s_2 ayant été choisis à cet effet. On recalcule alors les distances évolutives d'une part, et d'autre part, on calcule la longueur topologique de la branche. Sauf cas très particulier de matrice de distance, la distance topologique (somme des branches entre deux feuilles) ne peut pas être égale à la distance évolutive. Néanmoins, elle s'en approche fortement.

de paramètres.

C'est plus tard que Neyman [90] puis Felsenstein [34] ont appliqué ce raisonnement dans le cadre de la phylogénie moléculaire. La fonction de vraisemblance, en phylogénie, associe une probabilité d'obtenir les séquences, autrement dit les différents sites, en fonction de paramètres. Les paramètres (Θ) sont de deux types :

- discret : la topologie (τ),
- continus : les longueurs de branche (b), les paramètres des modèles d'évolution utilisés (θ).

La vraisemblance (L, pour l'anglais *likelihood*) peut alors être exprimée comme le produit des vraisemblances à chaque site ($S^{(i)}$ étant l'état du site à la position i dans l'alignement de longueur l) :

$$L(\Theta) = P(S \mid \Theta) = \prod_{i=1}^{l} P(S^{(i)} \mid \Theta) \tag{2.1}$$

Le processus consiste alors à faire varier ces différents paramètres de telle sorte

qu'on augmente la vraisemblance. Il s'agit d'explorer d'une part les topologies possibles, et d'autre part, les paramètres des modèles d'évolution et les longueurs des branches de l'arbre. Idéalement, on voudrait pouvoir évaluer toutes les topologies possibles de l'arbre. Mais en vertu de l'immense nombre d'arbres possibles, ceci n'est pas envisageable. Comme souvent en reconstruction phylogénétique, on utilise des heuristiques, censées approcher le résultat optimal.

L'algorithme de maximum de vraisemblance va donc évaluer les solutions les plus probables. Ce processus d'exploration des solutions possibles reste à la discrétion de chaque implémentation de cette méthode, mais généralement, il s'agit de partir d'un arbre construit avec une méthode de distances, et d'en faire varier la topologie avec les mouvements suivants :

- NNI, *Nearest Neighbor Interchange* : il s'agit d'échanger, dans un branchement, un nœud avec son oncle, comme illustré sur la figure 2.5.
- SPR, *Subtree Prune and Regraft* : un sous-arbre est élagué, c'est-à-dire retiré de l'arbre et branché ailleurs dans l'arbre.

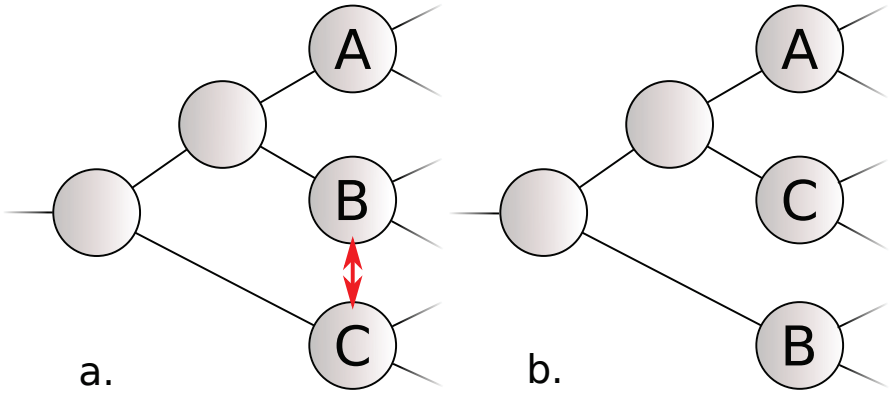


FIGURE 2.5 – ***Nearest Neighbor Interchange* (NNI).** Une partie d'un arbre est représentée. **a.** État initial de l'arbre. **b.** On a échangé les nœuds (et leurs sous-arbres respectifs) B et C.

À chaque mouvement, la méthode évalue s'il y a eu un gain de vraisemblance, et éventuellement, selon l'implémentation, recalcule les nouveaux paramètres qui maximisent la vraisemblance de cette nouvelle topologie. Le processus d'exploration se termine quand il n'est plus possible de faire augmenter la vraisemblance en pratiquant

des mouvements sur l'arbre. Alors, la topologie ayant la meilleure vraisemblance est conservée, et l'ensemble des paramètres Θ sont recalculés. On dispose alors de la topologie qui présente la meilleure vraisemblance[12].

La méthode du maximum de vraisemblance est actuellement le standard dans les méthodes de reconstruction d'arbres. Elle a été implémentée dans de nombreux programmes : PhyML [48], RAxML [117], PAML [134], PHYLIP [36], TreeFinder [58], FastTree [101].

2.1.6.4 Méthode bayésienne

En phylogénie, les méthodes bayésiennes se basent sur le théorème de Bayes, énoncé par Thomas Bayes et publié deux ans après sa mort, en 1763 [8]. Le théorème de Bayes vise à préciser la probabilité d'un évènement lorsqu'on dispose d'informations supplémentaires qui peuvent avoir une influence sur cette probabilité. Le théorème de Bayes s'énonce ainsi :

$$\mathbb{P}(A \mid B) = \frac{\mathbb{P}(B \mid A) \times \mathbb{P}(A)}{\mathbb{P}(B)} \tag{2.2}$$

avec A et B des évènements, $\mathbb{P}(A)$ la probabilité associée à l'évènement A, et $\mathbb{P}(A \mid B)$ la probabilité de A sachant que l'évènement B est réalisé. On parle alors de probabilité *postérieure*, car la probabilité $\mathbb{P}(A \mid B)$ est évaluée postérieurement à la connaissance que l'évènement B est réalisé.

Les implications de ce théorème sont difficiles à intuiter quand on n'en a pas l'habitude. Pour mieux en saisir l'idée, imaginons une expérience très simple : un expérimentateur doit tirer une boule et il a à sa disposition deux urnes. La première est très majoritairement remplie de boules noires (9/10), tandis que la seconde est remplie à 9/10 avec des boules blanches. L'expérimentateur tire à pile ou face pour savoir dans quelle urne il doit tirer une boule. La question qu'on se pose est de savoir dans quelle urne la boule a été tirée (on note A l'évènement *tirer dans la première urne*). Si l'expérimentateur tient la boule qu'il vient de tirer dans sa main, la seule chose qu'on peut déterminer sont les probabilités *a priori* : la probabilité que la boule ait été tirée dans la première urne est de 0,5, tout comme la probabilité que la boule ait été tirée dans la seconde. Si alors l'expérimentateur ouvre sa main et montre une boule noire (on

[12] En réalité, on n'en a aucune certitude mathématique si on utilise une heuristique, mais on a de bonnes chances de s'en approcher.

note N l'évènement *tirer une boule noire*), on a une information supplémentaire, et on peut dire que la probabilité qu'il ait tiré cette boule dans la première urne augmente franchement, à la défaveur de la probabilité qu'elle ait été tirée dans la seconde. C'est la probabilité *a posteriori*, qu'on calcule avec le théorème de Bayes. La probabilité que la boule ait été tirée dans la première urne *sachant que* la boule tirée est noire se calcule ainsi :

$$\mathbb{P}(A \mid N) = \frac{0,9 \times 0,5}{0,5} = 0,9 \tag{2.3}$$

On a donc déterminé que, étant donné l'information que la boule est noire, la probabilité de l'avoir tirée dans la première urne est de 0,9.

L'utilisation de ces concepts pour la phylogénie moléculaire est beaucoup plus récente puisque les logiciels qui permettent de faire de l'inférence bayésienne ne datent que des années 2000, essentiellement en raison des nombreux calculs nécessaires à cette approche, difficilement envisageables en routine sur des ordinateurs des années 1990. Les plus utilisés sont MrBayes [55, 106] et PhyloBayes [68]. L'approche bayésienne avait néanmoins été utilisée pour des applications plus restreintes dans la seconde moitié des années 1990 [135, 124].

Contrairement au maximum de vraisemblance où on cherchait à estimer la probabilité des séquences S en sachant les paramètres Θ (τ la topologie, b les longueurs de branches, et θ les paramètres du modèle d'évolution), ici on cherche à connaître la probabilité que l'arbre soit vrai avec de tels paramètres, connaissant les séquences :

$$f(\tau, b, \theta \mid S) = \frac{f(S \mid \tau, b, \theta) \times f(\tau, b, \theta)}{f(S)} \tag{2.4}$$

Cette fonction de probabilité postérieure contient la fonction de vraisemblance ($f(S \mid \tau, b, \theta)$), mais elle ne fonctionne pas de la même façon. La différence de fonctionnement réside avant tout dans la forme du résultat et de la méthode d'exploration.

Le résultat d'une inférence bayésienne consiste en une distribution de probabilité : chaque paramètre est représenté par un ensemble de valeurs possibles. Cela permet, à l'issue du processus, de déterminer la valeur moyenne d'un paramètre ainsi que son intervalle de confiance.

Autre point de différence avec la méthode du maximum de vraisemblance : l'ex-

ploration de l'espace des paramètres possibles. En maximum de vraisemblance, cette exploration est déterministe : on part d'une topologie donnée, et à force de changements, on se dirige vers une solution unique. Avec Bayes, on tire les valeurs des paramètres dans une distribution appelée *prior*. Cette distribution peut être une distribution uniforme (donc non informative), ou prendre une forme déterminée par l'expérimentateur s'il a une idée précise du profil de ses paramètres. L'exploration des paramètres ne se termine qu'une fois qu'un certain nombre d'itérations (défini par l'utilisateur) a été atteint. Elle se fait généralement par MCMC (*Markov Chain, Monte Carlo*), souvent combiné avec la méthode de Metropolis-Hastings [85, 51] : il s'agit de parcourir l'espace des paramètres de manière aléatoire, de proche en proche, en faisant varier un paramètre à la fois (parmi τ, b, et θ). Pour ce qui concerne la topologie, on la fait varier comme en maximum de vraisemblance, à l'aide de NNI et de SPR (voir *supra*). À chaque itération, on évalue le gain en terme de probabilité que représente le changement de paramètre. S'il y a un gain, on conserve le jeu de paramètres ayant abouti à l'augmentation. Si par contre il y a une baisse de probabilité, on effectue un tirage aléatoire qui va déterminer si on garde ce jeu de paramètres : plus la baisse de probabilités est forte, moins le jeu a de chances d'être conservé. Il est donc possible de conserver un jeu de paramètres qui donne une probabilité moins bonne. De proche en proche, l'espace des paramètres est ainsi parcouru, et le résultat de ce parcours, quand le nombre d'itérations tend vers l'infini, représente la distribution des probabilités postérieures.

2.1.7 Évolution de ces méthodes, modèles non homogènes

Les modèles évoqués dans la section précédente constituent un standard implémenté dans beaucoup d'outils depuis de nombreuses années maintenant. Mais ils ont des défauts : ils considèrent que l'évolution des séquences se fait selon le même processus et avec les mêmes vitesses d'évolution quel que soit le site, et quel que soit le clade considéré. En réalité, par exemple concernant les séquences codantes, les sites n'évoluent pas tous de la même façon, selon la partie de la protéine concernée. Ceci est également vrai pour les ARNr. De même, au fil du temps et selon les organismes concernés, les séquences peuvent voir leur processus d'évolution varier.

Ces idées ont donné lieu à des développements dès les années 1990 qui se sont

intensivement poursuivis ces dix dernières années : d'abord une modification des modèles homogènes pour rendre les taux d'évolution non-homogènes, puis la conception des modèles non homogènes.

Malgré les progrès apportés par ces développements, ils ne sont pas systématiquement utilisés dans les reconstructions phylogénétiques, du fait d'une augmentation considérable du nombre de calculs à réaliser. Par exemple, reconstruire des milliers d'arbres d'un large jeu de données de collections de familles de gènes homologues en utilisant des modèles non homogènes nécessiterait un temps de calcul inenvisageable.

2.1.7.1 Variation du taux d'évolution

La première solution consiste à utiliser des modèles homogènes, mais permettre de faire varier le taux d'évolution tout en utilisant un même processus évolutif quel que soit le site et ceci pour l'ensemble de l'arbre.

Loi Γ et variation de la vitesse entre sites : Il a été observé que les sites le long d'une molécule n'évoluaient pas tous à la même vitesse. Suite à cette observation, il a été proposé de modéliser cette variation de taux d'étolution à l'aide d'une distribution Γ [132]. Cette distribution Γ n'a aucune signification biologique mais est utilisée comme un outil mathématique permettant de générer facilement des taux d'évolution aux valeurs très différentes. La distribution Γ est caractérisée par deux paramètres, α la forme et β l'échelle et la moyenne de cette distribution, à savoir ici la moyenne des taux d'évolution est égale à α/β. Classiquement, on fixe $\alpha = \beta$ et le taux moyen est alors égal à 1. Pour des raisons de temps de calcul, la distribution Γ est ensuite discrétisée en n catégories, ce qui permet d'obtenir une valeur de taux de substitution pour chacune des catégories. L'utilisateur détermine *a priori* le nombre n de catégories de taux d'évolution pour le jeu de données concerné. L'algorithme va optimiser la valeur du paramètre α afin de déterminer ensuite pour chaque catégorie n son taux d'évolution. Le modèle d'évolution est alors ce que l'on appelle un modèle mixte, ou de mélange : la vraisemblance de chaque site est égale à la moyenne pondérée des vraisemblances calculées à partir de chacune des vitesses d'évolutions des n catégories. La pondération s'effectue par la probabilité pour le site d'appartenir à chaque catégorie. Ce nombre de catérogies est généralement faible : en routine, on en utilise souvent quatre.

Modèles hétérotaches : On constate également, que chaque site peut voir sa vitesse d'évolution varier au cours du temps. Lopez *et al.* [78] ont démontré que 95% des sites variables de 2000 séquences de cytochrome B présentaient une variation dans leur taux de substitution au cours du temps.

Des modèles ont été proposés pour répondre à cette problématique. Par exemple, le modèle de Galtier [40] permet au taux spécifique d'un site de varier selon les lignées, en estimant la proportion de sites sujet à des changements de taux au cours du temps et le taux de ce changement, le tout en maximum de vraisemblance.

2.1.7.2 Modèles non homogènes

Il est possible d'aller plus loin que la simple variation du taux d'évolution : faire varier la matrice Q des taux instantanés de substitutions elle-même, soit le long de la molécule, soit au cours du temps. Ainsi, respectivement différents sites ou différentes sous-parties de l'arbre utilisent différentes matrices Q.

Modèles non-homogènes en sites : Le modèle CAT, implémenté dans le logiciel PhyloBayes [68] est un des modèles non-homogène et il utilise l'approche Bayésienne. Il fait l'hypothèse que les différents acides aminés, le long d'une séquence protéique, ne subissent pas les mêmes contraintes : certains sites, pour des raisons fonctionnelles, voient leurs propriétés biochimiques (hydrophobicité, aromaticité, propriétés acido-basique par exemple) conservées. Par conséquent, chaque site privilégie certains types d'acides aminés, et peut avoir, dans le modèle CAT, sa propre matrice de fréquences d'équilibres Π. La matrice Π intervient dans la définition dans la matrice Q de la manière suivante : $Q = S \times \Pi$.

Le modèle optimise donc n profiles, c'est à dire n vecteurs de fréquences d'équilibre Π, permettant d'obtenir une matrice de substitutions Q_i ($1 \leq i \leq n$) par l'opération $Q_i = S_{F_{81}} \times \Pi_i$, avec $n \leq l$, avec l le nombre de sites pour la séquence étudiée. $S_{F_{81}}$ est une matrice d'échangeabilité dite *plate* [34]. Le nombre de profiles n est optimisé par approche bayésienne, et les Π_i sont déterminés d'après les séquences. L'évolution de chaque site s_j est modélisée par un profile Π_i : on peut donc pour chacun calculer une vraisemblance L_{s_j} utilisant Π_i. L'estimation des vraisemblances pour chaque site se fait en utilisant le même profile tout au long de l'arbre.

Le *et al.* proposent une version modifiée de PhyML [71] qui introduit des *modèles de mélanges de matrices de substitutions pour acides aminés*, utilisant le maximum de vraisemblance. Chacun de ces modèles propose plusieurs sous-modèles Q_i dont le nombre est prédéfini de manière empirique, chaque sous-modèle correspondant à des sites ayant des propriétés conformationnelles protéiques différentes, comme par exemple le fait pour un acide aminé de se trouver dans un feuillet bêta ou une hélice alpha. Les sous-modèles sont optimisés par le programme. Chaque site ς_j voit sa vraisemblance Q_j calculée de la façon suivante $L_{s_j} = \sum_k \alpha_k L_{k,s_j}$, avec α_k, la pondération de la catégorie k du mélange, optimisé par maximum de vraisemblance.

Il existe également des *modèles de mélange de codons*, comme implémenté dans CodeML [133]. Par exemple, il est possible de rendre non homogène le d_N/d_S le long des sites. Le d_N/d_S est le ratio du taux de substitutions non-synonymes divisé par le taux de substitutions synonymes. Ce ratio permet d'évaluer les pressions de sélection que subit une séquence : s'il est égal à 1, la sélection ne subit pas de pression ; s'il est très supérieur à 1, c'est que la séquence est soumise à une sélection diversifiante ; s'il est très inférieur à 1, c'est que la séquence est soumise à une sélection purificatrice. Il est alors possible, grâce à ce type de modèles de déterminer a posteriori pour chaque type l'appartenance à un type de catégorie de d_N/d_S, et d'avoir une idée de la pression de sélection qui s'exerce le long de la molécule.

Modèles non-homogènes en temps Les modèles non homogènes en temps peuvent par exemple faire l'hypothèse que la composition varie dans le temps, c'est-à-dire entre les lignées de l'arbre. Cela est utile par exemple quand les séquences en présence ont des compositions visiblement différentes : cette hétérogénéité est la conséquence de processus d'évolution changeants dans le temps, changements qui sont pris en compte par ce type de modèles.

Comme précédemment, on dispose de plusieurs matrices Q_i, mais dans le cas des modèles non homogènes en temps, c'est chaque branche qui peut se voir associée à une matrice Q_i, spécifique de par son vecteur de fréquences d'équilibres Π_i (Π est une matrice diagonale dont on utilise le vecteur correspondant comme vecteur de de fréquences d'équilibre). On distingue deux types de modèles non-homogènes en temps. Dans le type 1 (comme implémenté dans nhPhyML [15] pour les séquences nucléiques, et par le modèle COaLA [47] pour les séquences protéiques), chaque branche

est associée à une matrice Q_i. Dans le type 2 (comme décrit par Blanquart *et al.* [13]), plusieurs branches peuvent partager une même matrice Q_i : les changements se font en suivant des *breakpoints* qui ne sont donc pas systématiquement situés sur les nœuds internes de l'arbre, contrairement aux modèles de type 1.

2.1.8 Enracinement

La racine d'un arbre phylogénétique désigne un point dans une des branches où se situe l'ancêtre commun à toutes les séquences présentes dans l'arbre. Dans un arbre à n feuilles, il y a autant de positions de racines possibles que de branches, soit $2n - 3$. Les méthodes de reconstruction d'arbres donnent une information sur la topologie de l'arbre, et très souvent sur les distances évolutives séparant les différentes séquences. Mais ces méthodes ne peuvent pas donner d'information sur l'emplacement de la racine. Le racinement (ou enracinement) permet de donner un sens de lecture à l'arbre : le temps s'écoule de la racine vers les feuilles. Dans un arbre non raciné, on peut seulement savoir quelles séquences sont groupées ensemble.

Il existe plusieurs méthodes pour raciner un arbre. Les plus utilisées sont la méthode du barycentre, le groupe externe, et parfois la duplication ancestrale qui sont décrites ci-dessous. Certains modèles (comme les modèles irréversibles) ou certaines méthodes de reconstruction (UPGMA) produisent directement des arbres racinés.

2.1.8.1 Barycentre

La méthode du racinement par le barycentre, aussi appelée *midpoint rooting*, repose sur l'hypothèse de l'horloge moléculaire [139], selon laquelle les séquences évoluent à la même vitesse, continuellement au cours du temps, depuis la séquence de leur ancêtre commun. En acceptant cette hypothèse, on s'attend à trouver le point ancestral de l'arbre le plus éloigné possible de toutes les feuilles. Le point choisi est celui qui se trouve au milieu des deux nœuds les plus internes de l'arbre. Le problème est que cette hypothèse est néanmoins souvent fausse [75], les séquences évoluant la plupart du temps à des vitesses différentes.

2.1.8.2 Le groupe externe

La méthode du groupe externe (ou *outgroup*) nécessite d'avoir, dans le jeu de données, une ou plusieurs séquences qu'on sait *a priori* ne pas appartenir à une ou des espèces faisant partie du groupe étudié. Ce groupe de séquences est alignée avec les autres, et l'arbre est reconstruit en incluant les données du groupe externe. Ce dernier figure donc sur l'arbre généré par la méthode. La racine étant alors placée entre le groupe externe et le reste de l'arbre.

Par exemple, si l'arbre à reconstruire porte sur des séquences de bactéries de l'ordre *Bacillus*, on peut inclure dans les données une séquence homologue provenant d'un individu des *Lactobacillus*, et placer la racine entre la séquence *Lactobacillus* et le reste de l'arbre.

2.1.8.3 Duplication ancestrale

Dans certains cas, l'arbre contient des paralogues, et la duplication à leur origine est survenue chez l'ancêtre commun, ou avant son apparition. Si les éventuelles pertes sélectives ne sont pas trop importantes, on peut facilement identifier la racine. Il s'agit de la branche qui répartit des séquences paralogues appartenant aux mêmes espèces.

Par exemple, les séquences des facteurs d'élongation EF-1α/EF-tu sont paralogues aux séquences EF-2/EF-G. La duplication a eu lieu avant la divergence des trois domaines que sont les archées, les bactéries et les eucaryotes. La figure 2.6 représente un arbre obtenu à partir de différentes séquences de ce marqueur.

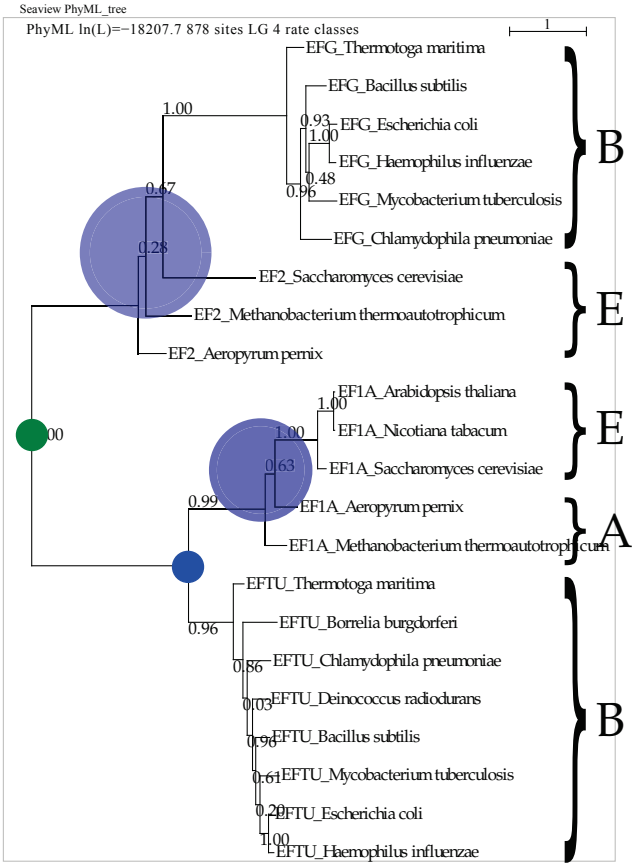


FIGURE 2.6 – **Exemple de racinement par duplication.** Arbre de séquences de facteurs d'élongation obtenu par maximum de vraisemblance. Différentes espèces des trois domaines sont impliquées : A pour archées, B pour bactéries et E pour eucaryotes. Le cercle vert représente un évènement de duplication. Les cercles bleus représentent des spéciations. Le cercle s'étale parfois sur plusieurs nœuds quand l'évènement de spéciation est mal inféré par la reconstruction, peu soutenu par les valeurs d'aLRT (voir section 2.1.9).

2.1.9 Soutien statistique

En phylogénie moléculaire, le soutien statistique vise à quantifier la confiance qu'on peut accorder à une branche d'une topologie. Ce soutien est déterminé par les données utilisées pour construire l'arbre. Il ne permet donc pas de connaître la probabilité qu'un branchement soit vrai, mais simplement à quel point on peut avoir confiance

dans chacune des branches de l'arbre généré par la méthode.

Ce soutien statistique est exprimé sous forme d'une probabilité, un nombre décimal entre 0 et 1, ou bien d'un pourcentage, un entier entre 0 et 100. Il donne la confiance dans ce que représente la branche qui le porte, en d'autres mots, la bipartition générée par cette branche[13]. Hormis le *bootstrap* décrit ci-dessous, il existe d'autres mesures qui pour certaines sont liées directement à la méthode de reconstruction employée comme l'aLRT [7] pour le maximum de vraisemblance, et les probabilités postérieures pour l'approche bayésienne.

2.1.9.1 Bootstrap

Le *bootstrap* est une méthode développée par Bradley Efron en 1979 [31], qui vise à estimer certains paramètres de distributions statistiques, et d'en donner un intervalle de confiance. Cette méthode ne requiert que les données sur lesquelles on travaille, et elle consiste à en refaire un échantillonnage, en tirant aléatoirement avec remise parmi les individus de ces données. Cette expérience de rééchantillonnage est réalisée un grand nombre de fois, et permet de révéler les éventuelles variations au sein de l'échantillon, selon les individus faisant partie du rééchantillonnage.

En phylogénie, le *bootstrap* est couramment employé pour évaluer le soutien statistique des branches. La méthode a été imaginée par Felsenstein en 1985 [35]. Il s'agit, une fois qu'un arbre a été construit, pour évaluer cet arbre, de génerer N alignements virtuels en pratiquant des échantillonnages aléatoires avec remise dans des sites de l'alignement qui a servi à construire l'arbre à évaluer. Ces N alignements sont utilisés pour générer les N arbres correspondants, avec la même méthode, le même modèle et les mêmes paramètres.

Dans l'arbre final, chaque branche produit une bipartition des feuilles. Il s'agit de vérifier, dans les arbres produits par le processus de *bootstrap*, combien d'entre eux contiennent une branche qui génère la même bipartition. On obtient ainsi une fraction d'arbres, qu'il est d'usage de représenter en pourcentage, autrement dit un entier de 1 à 100.

[13] En effet, chaque branche réalise une bipartition des feuilles de l'arbre : soit une feuille est d'un côté de cette branche, soit elle est de l'autre côté. Cela forme ainsi deux ensembles distincts de feuilles.

2.1.9.2 Interprétation du soutien statistique

Le soutien statistique des branches d'un arbre ne permet d'aucune manière d'émettre un jugement sur la qualité de la méthode, car il est donné par la méthode elle-même. Il permet simplement d'évaluer, selon la méthode de reconstruction utilisée, à quel point les données utilisées plaident pour l'existence de chaque branche.

Le cas du groupe N du virus VIH-1 est très parlant à ce sujet. Le VIH, le Virus de l'Immunodéficience Humaine est responsable du SIDA dont la compréhension est un immense enjeu de la médecine et de la biologie depuis la fin du XX^{e} siècle. Son origine est un des sujets de recherche et de controverses. Ce virus est classé en différents groupes et sous-groupes, qui sont différenciables par leurs génomes.

En 1999, Gao *et al.* [41] ont étudié la phylogénie d'un concaténat de gènes de VIH-1, et des concaténats de gènes VIS (virus homologue du VIH chez le chimpanzé) pour déterminer l'origine du groupe N. Un concaténat est un regroupement d'alignements sous la forme d'un super-alignement unique. Le but est d'augmenter le signal phylogénétique en combinant des données qu'on suppose avoir la même histoire. Accumuler des séquences ayant une histoire commune revient à augmenter la probabilité de la phylogénie résultante : on s'attend à augmenter le soutien statistique avec le nombre de séquences.

L'arbre obtenu groupe le génome du groupe N s'associe avec d'autres séquences que celles du VIS de chimpanzé, mais avec un *bootstrap* faible, toujours inférieur à 90%. Les auteurs ont découvert que le groupe N était en fait une mozaïque de deux virus : le VIH du groupe M et le VIS du groupe US porté par des chimpanzés *Pan troglodytes troglodytes*. Cela est confirmé par la reconstruction d'un arbre sur la première moitié du concaténat (contenant les gènes *gag*, *pol* et *vif*) d'une part, et d'un autre arbre sur la fin du concaténat (contenant les gènes *env* et *nef*) d'autre part. Ces deux arbres groupent les séquences différemment (respectivement N avec VIS groupe US, et N avec VIH groupe M). Les arbres sont représentés sur la figure 2.7

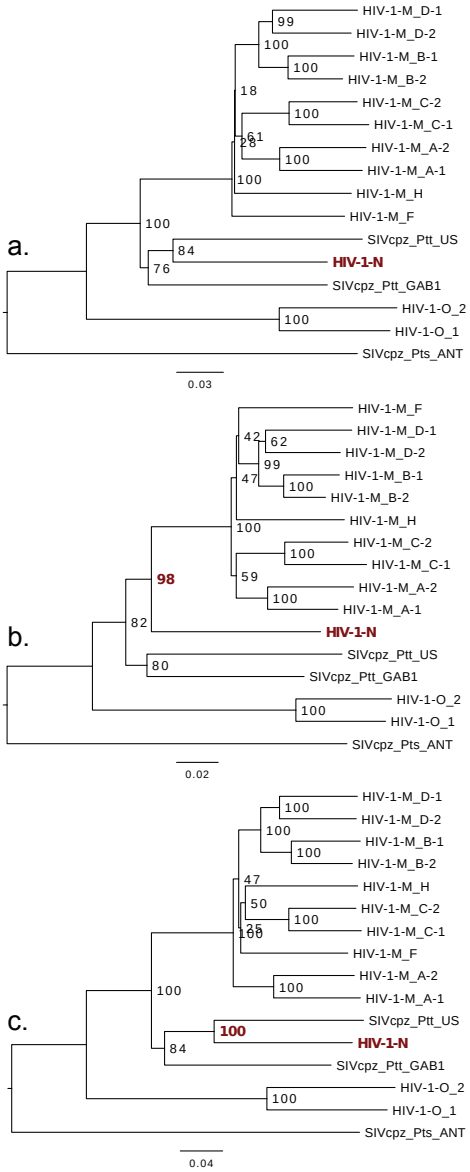


FIGURE 2.7 – **Arbres de concaténats de gènes du VIH et VIS**, d'après [41], réalisés avec une distance de Poisson. **a.** Arbre réalisé sur la totalité du concaténat. Le VIH-1 N est groupé avec des séquences de SIV de Chimpanzés *Pan troglodytes troglodytes*, mais avec un faible soutien. **b.** Arbre réalisé sur les 1400 premiers sites de l'alignement du concaténat. Le VIH-1 N est groupé avec les séquences de VIH-1 M, avec un fort soutien : 98%. **c.** Arbre réalisé sur sites situés après le 1400eme dans l'alignement du concaténat. Le VIH-1 N est groupé avec les séquences de VIS US de chimpanzés *Pan troglodytes troglodytes*, avec un fort soutien : 100%.

Dans cet exemple, le faible soutien statistique observé dans le premier arbre est dû au conflit des informations contenues dans les séquences. Chaque échantillonnage aléatoire avec remise pratiqué pendant le processus de *bootstrap* peut, par hasard, sélectionner majoritairement des sites situés au début ou à la fin de l'alignement.

2.2 Arbres d'espèces contre arbres de gènes

2.2.1 Introduction

Une famille de gènes est un ensemble de séquences codantes homologues. Au cours de l'évolution, une séquence subit de nombreuses modifications : insertions, délétions, substitutions. Elle peut, au sein d'une espèce, être dupliquée ; elle pet également être transférée dans une autre espèce. Tous ces évènements évolutifs qui jalonnent l'existence de la séquence ancestrale vont constituer une histoire unique. Reconstituer la phylogénie des séquences d'aujourd'hui peut donner un aperçu de cette histoire. Malheureusement, plusieurs facteurs limitent cette reconstitution. Le premier est qu'il est difficile de reconstituer une phylogénie avec certitude. Et même cette étape passée, le second facteur est que nous ne disposons que des séquences actuelles : comment savoir quand a eu lieu une duplication ? comment faire la différence entre la perte d'un gène et le fait que nous ne l'ayons tout simplement pas encore trouvé, parce que nous disposons de trop peu de séquences[14] ?

Bien souvent, ces évènements sont inférés à l'aune de la différence d'un arbre de gène et d'un arbre d'espèces qu'on suppose vrai, ou du moins qu'on utilise en référence. Toute différence de topologie soutenue est expliquée par un évènement, c'est la *réconciliation*.

2.2.2 Incongruence entre arbre de gènes et arbre d'espèces

Une hypothèse forte de la phylogénie est qu'on peut connaître les liens de parenté entre les espèces en analysant les liens de parenté entre des séquences portées par ces espèces. Les organismes sont alors des véhicules de gènes, et leurs histoires se

[14] Dans le cas de génomes complets, la question ne se pose pas, car on est certain de disposer de toutes les séquences présentes dans les espèces.

confondent. Pourtant – au regard de la simplicité d'une telle approche – cette hypothèse est souvent fausse. On observe alors une incongruence, c'est-à-dire une différence de topologie, entre l'arbre d'une famille de gènes et un arbre d'espèces, tel qu'exposé sur la figure 2.8.

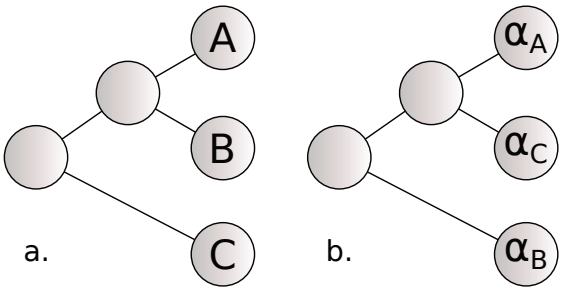


FIGURE 2.8 – Un exemple d'inconguence. (**a**) L'arbre des espèces *A*, *B*, et *C*. *A* et *B* sont frères tandis que leur ancêtre commun est frère de *C*. (**b**) L'arbre des gènes de la famille α. La lettre en indice indique l'appartenance du gène à une espèce. On voit ici que les gènes des espèces *A* et *C* sont frères et que leur ancêtre commun est frère avec le gène de l'espèce *B*.

Il y a plusieurs raisons qui peuvent conduire à des incongruences entre un arbre d'espèces et un arbre de gènes. Dans la mesure où l'on considère qu'il n'y a pas d'erreur dans l'arbre des espèces, on peut imaginer deux scénarios qui conduisent à une incongruence : le transfert horizontal, la duplication suivie de pertes, et le tri de lignées incomplet, processus qui sont expliqués ci-après.

2.2.3 Transferts horizontaux

2.2.3.1 Introduction

Historiquement, la transmission des caractères génétiques a été observée et théorisée chez des organismes observables à l'œil nu, essentiellement les organismes d'intérêt agronomiques comme les plantes et les animaux. Gregor Mendel publie en 1866 [45] le résultat de ses travaux sur la transmission héréditaire de caractéristiques phénotypiques chez le pois. Sans avoir aucun indice ni présomption sur la nature du support de l'information génétique, il théorise le fait que des caractères sont transmis à un individu par ses parents, de génération en génération. Cette conception, reconnue par la

suite comme la *génétique mendelienne*, est restée pendant longtemps la seule approche de la génétique.

Puis les techniques d'observations se sont faites de plus en plus pointues. On a pu observer les organismes microscopiques — bactéries, archées, eucaryotes unicellulaires — et comprendre que les bactéries étaient capables d'échanger de l'information génétique. Pas seulement de manière verticale, d'une génération à l'autre comme on l'avait déjà modélisé, mais de manière horizontale, c'est-à-dire entre individus contemporains. Cet échange d'information génétique se fait par transfert de matériel génétique, c'est-à-dire d'ADN. Une fois que de l'ADN a pénétré dans une cellule bactérienne, et s'il est codant, il peut être utilisé par la machinerie cellulaire pour produire la protéine correspondante. L'information génétique peut alors être intégrée dans le génome de son hôte. Si cette nouvelle caractéristique apporte un avantage sélectif à l'individu porteur de ce nouveau gène, il va se multiplier, et potentiellement se généraliser dans l'espèce porteuse.

Il existe plusieurs mécanismes pour faire rentrer de l'ADN exogène dans une cellule bactérienne. Trois mécanismes sont clairement identifiés :

La transformation : découverte en 1928 par Frederick Griffith [46]. Un brin d'ADN est transporté depuis le milieu de vie jusqu'à l'intérieur de la cellule. Seules les cellules dites *compétentes* sont naturellement capables d'intégrer de l'ADN de cette manière. Il est aussi possible de perméabiliser artificiellement une membrane plasmique bactérienne pour que les cellules puissent internaliser l'ADN. Une fois dans la cellule, l'ADN peut, si le génome hôte présente des régions homologues avec lui, recombiner. Il est également possible que cet ADN soit intégré dans le génome hôte *via* des sites d'insertion particuliers.

La conjugaison : décrite en 1946 par Lederberg et Tatum [73]. Deux bactéries échangent, par l'intermédiaire d'un canal appelé *pilus*, un plasmide[15].

La transduction : découverte par Zinder et Lederberg lors d'expériences sur la conjugaison [138]. Un bactériophage (virus contaminant des bactéries) infecte une bactérie en injectant son ADN viral. Cet ADN commande la production de

[15]Le plasmide est, chez la bactérie — et la levure —, est une molécule d'ADN qui n'est pas chromosomique, dont la réplication est autonome et qui n'est pas essentielle à la survie de l'organisme. [72]

nouveaux phages qui sont expulsés quand la cellule hôte meurt. Par hasard, certains de ces phages peuvent avoir encapsidé de l'ADN de la bactérie dans laquelle ils ont crû. En infectant une nouvelle cellule, cet ADN bactérien se retrouve injecté dans une autre bactérie. Si la bactérie receveuse survit à l'infection du phage, l'ADN pourra éventuellement suivre le processus d'intégration dans son génome.

2.2.3.2 Effet sur les arbres phylogénétiques

La figure 2.9 montre l'effet d'un transfert horizontal sur un arbre de gènes.

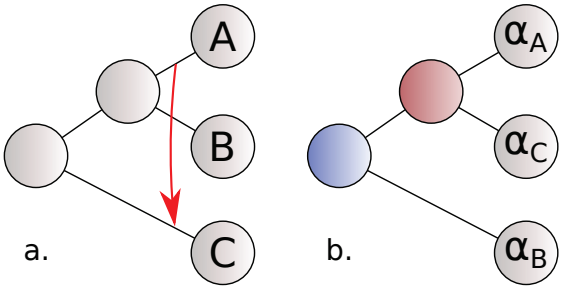


FIGURE 2.9 – **Exemple de transfert horizontal. (a)** Sur l'arbre des espèces, la flèche rouge figure un transfert horizontal du gène α de l'ancêtre de l'espèce A à l'ancêtre de l'espèce C. **(b)** Le gène α de l'espèce C est ainsi groupé avec celui de l'espèce A. L'évènement associé au nœud en rouge est ce tranfert horizontal, alors que l'évenement associé au nœud en bleu est une spéciation.

2.2.3.3 Détection des transferts

Plusieurs méthodes bioinformatiques permettent d'évaluer et de détecter les transferts horizontaux. Elles sont classées en deux catégories : paramétriques et non-paramétriques.

Les méthodes paramétriques sont basées sur le fait que les séquences appartenant à une espèce ont des caractéristiques communes. Les séquences transférées ont, quant à elles, des caractéristiques propres à leur hôte d'origine, c'est-à-dire le donneur. Un usage biaisé des codons (mesuré par des indices tels que le CAI (*Codon Adaptation Index*) ou un χ^2 d'usage du code), ainsi que le taux de GC, sont des caractéristiques mesurables tout au long d'un génome. Les séquences qui présenteraient localement des

différences significatives avec le reste du génome pourraient être considées comme ne faisant pas partie du génome d'origine : on les considère donc comme des transferts horizontaux [70]. Un des défauts intrinsèques à ces méthodes est qu'elles génèrent beaucoup de faux négatifs. En effet, les séquences assimilées subissent les mêmes contraintes évolutives que le reste du génome de l'espèce receveuse. Elles finissent donc, avec le temps, par présenter les mêmes caractéristiques que les séquences autochtones, et deviennent indifférentiables de ces dernières.

En 2001, Ochman et collaborateurs ont déterminé, à l'aide de méthodes paramétriques, que 12,8% du génome de *Escherichia coli K12* était putativement issu de transferts horizontaux [93]. Ils démontrent entre autre que de nombreux génomes bactériens et archéens contiennent des séquences issues d'autres procaryotes.

Les méthodes non paramétriques quant à elles, utilisent la phylogénie. Elles sont basées pour beaucoup sur la réconciliation (voir *infra*) de l'arbre des gènes avec l'arbre des espèces. Voici quelques méthodes basées sur la phylogénie pour la détection de transferts : Prunier [1], ODT [119].

2.2.4 Duplication suivie de pertes

2.2.4.1 Introduction

On appelle *duplication* l'évènement qui consiste à passer d'une version unique d'un gène au sein d'un hôte à deux versions de ce gène. Dans une revue publiée en 2004 [122], Taylor et Raes soulignent que le concept de duplication est relativement ancien, même s'il est alors décorrélé de la définition actuelle d'un gène. En effet, Darwin lui-même, dans son célèbre ouvrage *L'Origine des espèces* [22], écrivait :

> Nous avons précédemment vu que des éléments plusieurs fois répétés sont éminemment susceptibles de varier en nombre et en structure ; par conséquent, il est fort probable que la sélection naturelle, au cours de la longue succession de modifications, doit s'être saisie d'un certain nombre d'éléments originairement répétés de nombreuses fois, et les a adapté à des fins des plus diverses.[16]

[16] Texte original : « We have formerly seen that parts many times repeated are eminently liable to

Ce qu'écrit Darwin s'est avéré vrai pour les gènes dupliqués : les duplications sont souvent source de variabilité et d'innovations. En effet, quand un gène est en version unique, la moindre modification peut avoir des conséquences sévères sur l'organisme ; alors que lorsqu'il y a au moins deux copies d'un gène, l'une des deux peut évoluer sans conséquence pour la fonction supportée par la première. Les deux copies peuvent alors avoir des destins différents :

sous-fonctionnalisation : chacune des deux copies perd une partie de ses fonctions, et fonctionne de manière complémentaire. Au lieu d'avoir un gène qui assure une fonction, on a deux gènes qui assurent chacun une partie de la fonction ;

néo-fonctionnalisation : un des deux gènes garde la fonction initiale, l'autre change de fonction ;

perte : un des deux gènes est perdu

C'est ce dernier cas de la perte du gène qui pose problème en phylogénie : il n'y a pas moyen, *a priori*, de faire la différence entre une séquence qui n'existe plus et une séquence qui n'a jamais existé. On peut parfois être face à un problème de paralogie cachée. Si cette paralogie cachée n'est pas détectée, elle peut amener à une interprétation fausse des arbres de gènes.

2.2.4.2 Effets sur les arbres phylogénétiques

2.2.5 Tris de lignée incomplets

Ce phénomène ressemble un peu, par son processus, à la duplication suivie de perte expliquée ci-dessus. Un gène peut avoir plusieurs allèles, qui sont autant de versions différentes de ce gène, disponibles dans une population ou plus largement dans une espèce. Si certains allèles viennent à disparaître sélectivement dans plusieurs espèces, les séquences restantes incriminées peuvent être vues à tort comme l'unique version du gène de l'espèce donnée, et ces séquences peuvent être regroupées suite au fait qu'elles ont

vary in number and structure ; consequently it is quite probable that natural selection, during the long-continued course of modification, should have seized on a certain number of the primordially similar elements, many times repeated, and have adapted them to the most diverse purposes. » — traduction personnelle.

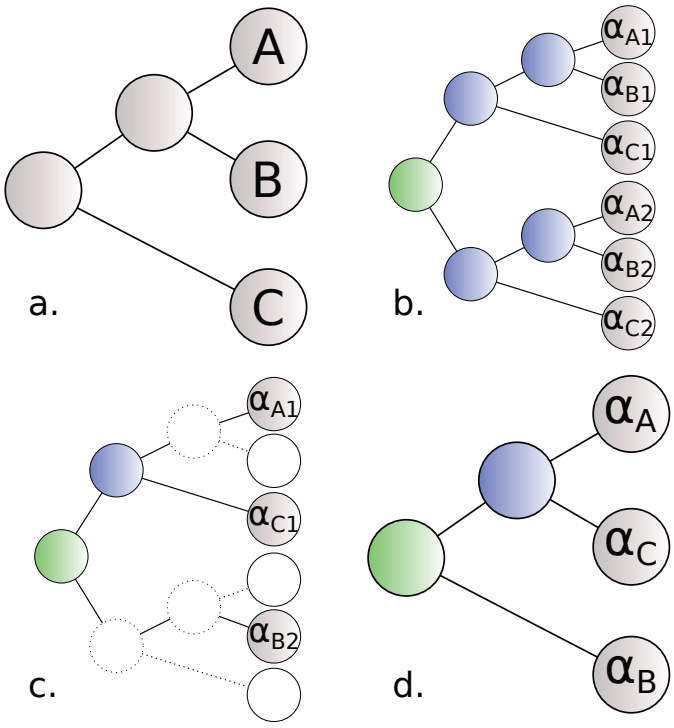


FIGURE 2.10 – **Exemple d'une duplication suivie de pertes générant une paralogie cachée. (a)** L'arbre des espèces. **(b)** Dans l'arbre de gènes, une duplication ancestrale, figurée en vert se produit. α_{A1}, α_{B1} et α_{C1} sont des gènes paralogues des gènes α_{A2}, α_{B2} et α_{C2}. Les nœuds représentés en bleu sont des nœuds de spéciation dont l'histoire est identique à celle décrite par l'arbre des espèces en (a). **(c)** Des pertes sélectives interviennent sur les gènes α_{B1}, α_{A2}, et α_{C2}. **(d)** La phylogénie résultant de ces pertes est incongruente avec l'histoire des espèces en (a). La paralogie est dite *cachée*, car chaque séquence n'est présente qu'en une seule copie, et on ne peut donc pas déduire qu'il s'agit de paralogie.

conservé un allèle unique commun (qui a néanmoins divergé). La figure 2.11 illustre ce phénomène.

2.2.6 Réconciliation

Les arbres de gènes et d'espèces peuvent donc, et c'est très souvent le cas, ne pas être congruents. Ces différences ne permettent pas de déduire à coup sûr l'arbre des espèces d'un arbre de gène. Mais elles sont révélatrices d'évènements évolutifs. Confronter un arbre de gène et un arbre d'espèces permet d'inférer ces évènements, et cela s'appelle

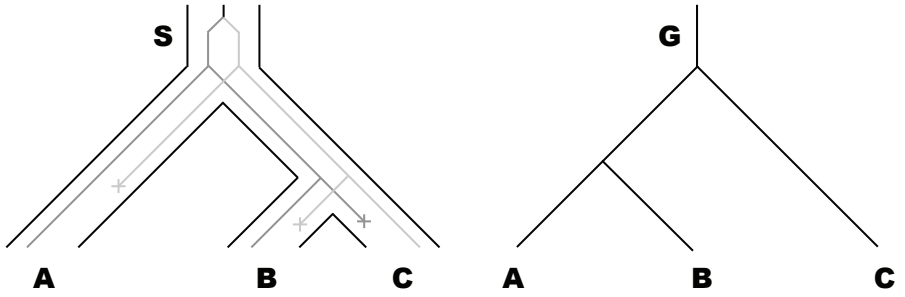


FIGURE 2.11 – **Tri de lignées incomplet**, tiré de [98]. **S** est l'arbre des espèces A, B, et C, dans lequel deux allèles (gris clair et gris foncé) d'un gène évoluent depuis l'ancêtre commun à ces trois espèces. L'allèle gris clair s'éteint chez A et B, tandis que l'allèle gris foncé s'éteint chez C. L'arbre **G** du gène résultant groupe alors les espèces A et B, puisque leurs séquences sont plus proches (allèles gris foncé) : il n'est pas conforme à l'arbre des espèces.

la réconciliation. Il s'agit d'établir un scénario qui explique les différences constatées par une succession d'évènements évolutifs

2.2.6.1 Détection de paralogie

La réconciliation peut être utilisée par de nombreuses méthodes pour différencier les orthologues des paralogues [87, 96, 136, 33]. Toutes fonctionnent sur le même principe : à chaque fois qu'une partie de l'arbre est incongruente, on présume que c'est à cause d'une duplication suivie de pertes. On procède par parcimonie (ou en pondérant les évènements de duplications et de pertes), et on évalue le scénario qui suppose le moins d'évènements de duplications et de pertes. Sur la figure 2.10, cela revient à disposir de l'arbre de gènes en **(d)** et l'arbre d'espèces en **(a)** et en déduire le scénario décrit par **(b)** et **(c)**.

2.2.6.2 Détection de transferts

Les transferts horizontaux peuvent également être inférés par réconciliation. Par exemple, Prunier [1] part d'un arbre d'espèces, et effectue des opération d'élagage (*prune*) et de greffe (*regraft*) de parties de l'arbre jusqu'à ce que l'arbre soit congruent avec l'arbre d'espèces. Cette méthode permet d'obtenir une succession d'évènements de transferts horizontaux : chaque déplacement de sous-arbre est vu comme le transfert du gène d'intérêt à ce sous-arbre.

2.2.6.3 Réconciliation complète

Des méthodes récentes (par exemple ALE [119]) permettent de faire une réconciliation complète, c'est-à-dire d'expliquer les incongruences par des duplications suivies de pertes et également des transferts horizontaux.

2.3 Collections de familles de gènes

2.3.1 Introduction

Nous disposons de très nombreuses séquences. En plus des banques généralistes accessibles à toute la communauté, comme GenBank [9] et SwissProt [79], il existe de nombreuses banques spécialisées. Et avec l'avènement du séquençage à prix très bas, de nombreuses équipes de recherche dans le monde disposent de séquences qui ne sont pas rendues publiques.

Parmi ces séquences, beaucoup codent pour des gènes. Pour mieux comprendre les histoires de ces gènes, il faut en faire des arbres, et donc les regrouper en familles de gènes homologues.

2.3.2 Recherche de similarité : BLAST

Il existe un moyen de trouver des séquences qui se ressemblent et de quantifier cette ressemblance. Les méthodes qui proposent de le faire sont basées sur l'alignement. Contrairement à l'alignement *multiple*[17], le procédé dont il est question ici consiste à n'aligner que deux séquences à la fois, et à quantifier l'identité de ces séquences.

BLAST [5, 17] est une méthode d'alignement local publiée en 1990. Elle répond à la nécessité de trouver une heuristique qui permet d'éviter d'avoir à faire un alignement exhaustif de tous les sites de toutes les séquences. Avant BLAST, l'algorithme de Smith et Waterman (implémenté dans SSEARCH [115]) permettait déjà de faire un alignement exact, mais pas complet, en utilisant quelques astuces de programmation dynamique. Cependant, BLAST est bien plus performant et permet une montée en échelle très satisfaisante, puisque malgré des banques de données à la croissance exponentielle, cette méthode rencontre toujours une utilisation massive et satisfaisante.

[17] Voir 2.1.4.

Afin de finaliser la construction de la banque, et avant de pouvoir faire une première requête avec BLAST, on doit amorcer un processus d'indexation de mots. Il s'agit pour BLAST de créer une liste organisée, c'est-à-dire dans laquelle on pourra faire des recherches rapides par la suite, de *k*-mères, petits morceaux de séquences de longueur *k*, extraits de celles dans lesquelles on veut pouvoir faire des recherches de gènes similaires. Cette indexation faite, l'algorithme est capable de retrouver, dans l'ensemble des séquences cibles, celle(s) qui est (sont) la (les) plus proche(s) d'une séquence requête. Des *k*-mères sont extraits de la requête, puis une recherche de ces mots est effectuée dans l'index préalablement créé.

On dispose, à ce stade de la recherche, de positions dans les séquences cibles qui sont identiques à certaines positions de la séquence requête. C'est pour cette raison que BLAST est dit un alignement *local* : il commence par trouver des régions identiques entre les séquences. BLAST nomme ces régions des graines (*seeds*), et c'est à partir d'elles que l'alignement commence. Ensuite, l'algorithme va essayer d'étendre cet alignement à gauche et à droite de la graine. Une matrice de pénalités est nécessaire : elle permet à BLAST de quantifier la similarité des séquences. L'extension de l'alignement est interrompue si une des séquences arrive à son extrémité, ou si la combinaison des pénalités laisse à penser qu'il est peu vraisemblable que l'on continuera à trouver de la similarité en poursuivant l'extension.

Au terme de ce processus, une liste de segments homologues pour chaque séquence est retournée. Il s'agit des segments les plus ressemblants à l'intérieur d'une séquence. Pour chacune, BLAST fournit plusieurs informations dont le score et la *Expect-value* (*E-value*). Le score est tout simplement informatif de la similarité des deux séquences. Le meilleur score possible correspond à l'alignement de deux séquences parfaitement identiques. La *E-value*, quant à elle, donne une information sur la fiabilité de cet alignement. Cette valeur est le nombre d'alignements de même score que l'alignement en question que l'on s'attend à trouver par hasard dans une collection de séquences de la taille de la banque dans laquelle on a fait la requête. Cette valeur est d'autant plus petite que l'alignement est grand et son score élevé, et que la banque de séquences est de petite taille. Plus la *E-value* est faible, plus l'alignement aura de sens et sera perçu comme significatif.

2.3.3 Constitution des familles

L'opération consiste à rassembler les séquences qui sont suffisamment ressemblantes pour qu'aucun doute ne puisse exister quant à leur homologie. Il faut pour cela créer une banque contenant toutes les séquences à répartir en familles. Pour chacune, on effectue une recherche de similarité *via* BLAST. On nomme cette opération *BLAST tout contre tout*, car chaque séquence est virtuellement comparée à toutes celles du jeu de séquences.

L'étape suivante est d'établir des groupes de gènes similaires (ou *clusters*). Il existe de nombreux programmes permettant d'effectuer de tels regroupements de séquences : SiLiX [86], hcluster_sg [107], MC-UPGMA [77], et MCL [32]. Le but de ces programmes est de créer des familles de protéines qui sont homologues sur toutes leur longueur, contrairement à ce que l'on peut trouver dans des collections de domaines comme Pfam [102] ou ProDom [112] qui ont pour but d'identifier des domaines protéiques[18].

Le programme SiLiX est dédié à la construction des banques de familles de gènes utilisées dans notre laboratoire (HOGENOM, HOVERGEN, HOMOLENS). Ce programme travaille à l'échelle de la protéine entière (ou du gène entier), il s'agit simplement de faire du *single-linkage*, c'est-à-dire que si la protéine A ressemble à la protéine B, et que la protéine B ressemble à la protéine C, alors on met dans la même famille A, B, et C, sans vérifier que A et C partagent un fort niveau d'identité. Mais un problème peut alors survenir : il se pourrait que A et B soient similaire sur une partie de leur séquence, et que B et C soient similaires sur une autre partie de la séquence, comme montré sur la figure 2.12. La solution trouvée est d'exiger un seuil mimimum de couverture : la similarité des séquences doit couvrir une certaine partie des protéines candidates au regroupement.

Il existe plusieurs collections de familles de gènes basées sur la phylogénie : HOGENOM, HOVERGEN et HOMOLENS [97] ; Ensembl/Compara [38] ; PhylomeDB [56] ; OMA [4] ; eggNOG [100] ; ProtClust [64].

[18] Ces banques sont basées sur l'hypothèse que les protéines sont modulaires, et que l'évolution se fait au niveau de chaque domaine.

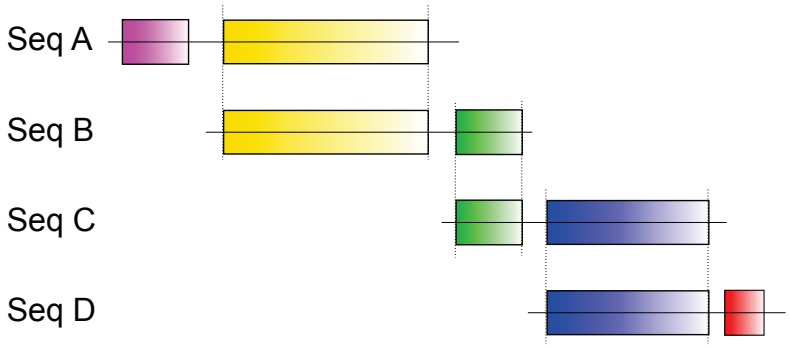


FIGURE 2.12 – **Contrainte de couverture en clustering *simple lien***, adapté de Miele et collaborateurs [86]. Les quatre protéines contiennent des domaines homologues (représentés par les rectangles de couleur). Pour éviter de placer dans une même famille les protéines qui ne partagent aucune homologie (par exemple A et D), les alignements deux à deux de séquences sont pris en compte dans le cluster seulement s'ils couvrent un seuil minimum de la longueur de chacune des deux protéines. Ce seuil doit être assez haut pour exclure les cas tels que l'alignement $\{B,C\}$, qui pourrait conduire au groupement de A et D.

2.3.3.1 Représentation formelle des arbres phylogénétiques

Les structures arborescentes sont courantes en informatique, et il existe de nombreuses façons de les représenter en vue de leur stockage. Aujourd'hui, un des grands standards pour ce type de données est le XML.

Mais historiquement, c'est un format en texte brut (*plain text*) qui s'est imposé : le format Newick [36] (autrement appelé *New Hampshire*). Le principe est simple et permet à quiconque, avec un éditeur de texte basique, de composer un arbre. Les noms des feuilles sont écrits en toutes lettres, et groupés par des parenthèses. Dans un même groupe, il peut y avoir plusieurs feuilles, séparées par des virgules. Un ensemble de caractères entre deux parenthèses représente donc un nœud et le sous-arbre qui y est afférent. Une distance au nœud père peut être précisée sur tous les nœuds en utilisant le symbole *deux points*, suivi de ladite distance. Le texte des nœuds internes doit être, le cas échéant, écrit après la parenthèse qui termine le groupe défini par ce nœud : cela peut être son nom ou bien un soutien statistique qui lui est associé.

Un exemple simple du format Newick pour les quatre espèces Homme, Chimpanzé, Rat et Souris, illustré sur la figure 2.13, peut être :

```
((Homo sapiens, Pan troglodytes), (Mus musculus, Rattus rattus));
```

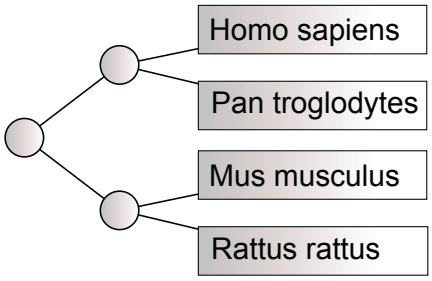


FIGURE 2.13 – Arbre codé par la chaîne `((Homo sapiens, Pan troglodytes), (Mus musculus, Rattus rattus));` au format Newick.

Plus récemment, des formats XML ont été proposés. XML, *Extensible Markup Language*, est un langage de structuration des données par balises. XML définit une base mais n'est pas utilisée en tant que telle. Des dérivés, appelés *dialectes*, définissent leur propre grammaire et leur propre vocabulaire. L'exemple le plus connu de dialecte XML est sans doute xHTML, langage utilisé pour programmer l'intégralité[19] des pages web.

Deux dialectes concernent directement la phylogénie : phyloXML [50] et NeXML [127]. Ils ont des avantages inhérents au fait qu'ils sont des dialectes XML. Le chargement du fichier XML est assuré par des bibliothèques éprouvées et très répandues[20], présentes dans de nombreux langages de programmation, alors que chaque programme de phylogénie implémente généralement son propre analyseur Newick. Les différents dialectes XML, grâce à leurs définitions très strictes, permettent par ailleurs de s'assurer qu'un document (ici un arbre phylogénétique) est bien valide, simplement en le soumettant à une bibliothèque XML, accompagné de sa définition. L'immense inconvénient des langages XML est que les fichiers ainsi générés perdent en lisibilité et augmentent en taille. Mais cet excès de complexité est compensé par la quantité d'informations qui peuvent être ajoutées, et ceci sans risque de générer un arbre non valide. Quelques programmes utilisent ces formats XML : citons Archeopterix [50], BioPython [20] pour phyloXML, et une dizaine d'autres programmes[21] utilisant NeXML.

[19] En réalité, environ 20% des sites web utilisent le langage html, mais on peut le considérer comme un ancêtre très proche du xHTML.

[20] libxml2 sous Linux (utilisables par des programmes C, C++, Python, PHP, Ruby), MSXML pour Microsoft, Xerces pour les programmes Java, pour ne citer que les plus courants.

[21] Voir `http://www.nexml.org/`.

3 Méthodes développées

Nous avons vu dans le chapitre précédent qu'il est possible, à partir d'un grand ensemble de séquences protéiques et/ou nucléiques, de construire des collections de familles de gènes homologues. Pour chaque famille, on peut construire un arbre phylogénétique qui reconstitue l'histoire des séquences actuelles jusqu'à la séquence ancestrale.

La suite de programmes présentée dans cette thèse, TPMS (*Tree Pattern Matching Suite*), vise à construire une base logicielle pour l'analyse systématique de ces arbres, après leur construction. Elle est constituée de trois exécutables codés en C++. Je donnerai dans un premier temps des détails sur l'implémentation de la suite, sa structure, et dans quel contexte logiciel elle s'inscrit. Ensuite, je décrirai les algorithmes de base utilisés par le programme pour l'association de nœuds à des taxons, et pour le *pattern matching*. Enfin, je décrirai les méthodes développées pour l'enracinement automatique, la recherche de motifs, et la détection automatique d'incongruences.

3.1 Implémentation et concepts utilisés

3.1.1 Notation

Pour expliquer les fonctions développées, j'utilise une notation algorithmique. Elle se base sur les structures simples et classiques de la programmation : boucles `pour tous`, `tant que`, conditions `si`, `sinon`, `sinon si`. Les types des variables sont données en in-

troduction. Le point signifie une action sur un objet, et les actions sont écrites en français. Ainsi, si `chaîneExemple` est une chaîne de texte, `chaîneExemple.dernierCaractere()` retourne le dernier caractère de la chaîne en question.

D'autre part, bien que la suite de programmes soit articulée en classes, cette notion est peu pratique à représenter dans un document hors du contexte du programme. Ainsi, si la classe `Taxon` contient une méthode pour obtenir le taxon père, cette méthode sera décrite par une fonction qui prend en premier argument un objet taxon, de la même manière qu'en langage Python[1], le premier argument d'une méthode d'une classe est `self` qui désigne une instance de la classe elle-même. Cette représentation ne nuit pas à la description de l'algorithme.

3.1.2 Programmation orientée objet

3.1.2.1 Principe

TPMS est structuré sous forme de classes, suivant le paradigme de la programmation orientée objet (POO). Cette façon de programmer, imaginée dans les années 1960–70, puis fortement popularisée au milieu des années 1990 avec les langages C++ puis Java, change radicalement la logique du programme en permettant au programmeur de réunir des concepts communs dans des *classes*. Ces classes correspondent le plus souvent à un objet de la vie réelle ou à un concept traité par le programme. À titre d'exemple, la classe la plus utilisée dans TPMS est le nœud d'arbre. Autour de ce concept, on réunit des variables qui sont appelées des *attributs* comme son nom, son éventuel père, ses éventuels fils, la distance qui le sépare de son père, le soutien statistique de la branche qui mène à lui. Généralement, ces attributs ne sont pas accessibles aux autres classes, on parle alors d'*encapsulation* : chaque classe est à même de gérer ses propres attributs, et tout changement depuis l' extérieur suppose un contrôle des données. On réunit aussi des fonctions qui sont appelées *méthodes*, par exemple ajouter ou supprimer un fils, connaître le père, ou changer la distance qui le sépare du père.

La classe désigne le concept, mais quand on crée puis travaille sur un nœud en particulier, on parle d'un *objet* de la classe nœud. Passer du concept générique de la classe à un exemplaire particulier de cette classe s'appelle *l'instanciation*.

[1] `http://docs.python.org/2/tutorial/classes.html#random-remarks`

Pour ce qui concerne les langages compilés, la POO ne change rien à l'exécution du programme, à son efficacité ou sa fiabilité[2], et ceci par essence. En effet, un programme écrit en POO est, au moment de sa compilation, converti en une suite d'instructions linéaires, car la seule façon dont fonctionne un processeur est de dépiler une liste d'instructions et éventuellement de bifurquer à un autre endroit du code au moment d'un branchement conditionnel (*if*...*else*). Il est d'ailleurs théoriquement possible de produire deux programmes binaires identiques, l'un ayant été écrit en POO, l'autre en programmation procédurale, plus classique. Les améliorations se situent au niveau de la cohérence, de la facilité à programmer, de la maintenabilité, de la lisibilité, de la possibilité de coopération entre plusieurs programmeurs, et donc de la réutilisabilité du code.

3.1.2.2 Classes de TPMS

TPMS est découpé en dix classes dont huit sont propres à la gestion de collections de familles de gènes, et deux dédiées au fonctionnement interactif du programme (affichage de barres de défilements, et gestion des paramètres d'entrée au programme.). Ci-dessous sont listés les noms et les prérogatives de chaque classe.

DataBase : Chargement de familles de gènes, contient toutes les familles de la collection d'arbres. Contient le code pour l'exécution parallèle de fonctions sur chaque famille. Contient l'arbre des espèces.

Family : Contient une famille, son nom, son arbre, la liste des espèces, toutes les fonctions de racinement, de mapping, de détection d'incongruences.

Pattern : Concerne les motifs d'arbres, recherche de motifs dans les arbres binaires.

NodeConstraints : Un nœud d'arbre requête et les contraintes qui y sont attachées : restrictions sur les espèces autorisées, le type du nœud, le lien direct avec le nœud fils[3]...

[2]Néanmoins, la programmation orientée objet permet la mise en œuvre de techniques de programmation (encapsulation, cohérence des classes), ce qui représente un réel progrès et un confort certain pour le programmeur.

[3]Voir 3.3

CandidateNode : Concerne la reconstruction *a posteriori* des arbres qui correspondent au motif. Une instance de cette classe est potentiellement un nœud de l'arbre de gènes qui correspond à un nœud de l'arbre requête[4].

TreeTools : Contient essentiellement des méthodes statiques[5], cette classe est avant tout une boîte à outils de phylogénie : lecture du Newick étendu, binarisation des arbres et modification des arbres. Idéalement, ces fonctions devraient être intégrées aux fonctions de base de Bio++ (voir plus bas).

Taxon : Gestion des relations entre espèces et groupes taxonomiques. Indexation de tous les descendants et ancêtres d'un groupe taxonomique. Contient des fonctions telles que trouver le dernier ancêtre commun de deux espèces.

Query : Interprétation d'une chaîne de requête de *tree pattern matching*.

Waiter : Gestion de l'affichage des barres de progression (en mode console), gère l'avancement en parallèle de plusieurs fonctions.

CmdLineArgs : Gestion avancée des commandes entrées par l'utilisateur lors du lancement d'un des programmes TPMS.

La figure 3.1 récapitule les classes de TPMS ainsi que leurs dépendances. On peut y voir de nombreuses flèches à double sens, et parfois des cycles : il s'agit de dépendances circulaires. Par exemple, une instance de la classe `DataBase` contient des listes de taxons de la classe `Taxon`, donc la définition de la première requiert la définition de la seconde. Mais chaque objet de la classe `Taxon` se réfère à un nœud précis dans l'arbre d'espèces qui est stocké dans l'objet de la classe `DataBase`, et requiert donc, pour sa définition, que cette dernière soit décrite.

[4] Voir 3.3.4.

[5] Dans une classe, une méthode statique est une fonction qui n'est pas instanciée avec l'objet, parce qu'on estime qu'elle n'est pas conceptuellement liée à un exemplaire d'objet en particulier. Par exemple, on implémente une fonction pour supprimer un nœud et les nœuds du sous-arbre associé. Cette fonction n'est pas spécialement liée à une famille, ou à un autre objet des classes disponibles dans TPMS, donc elle a été placée dans `TreeTools`.

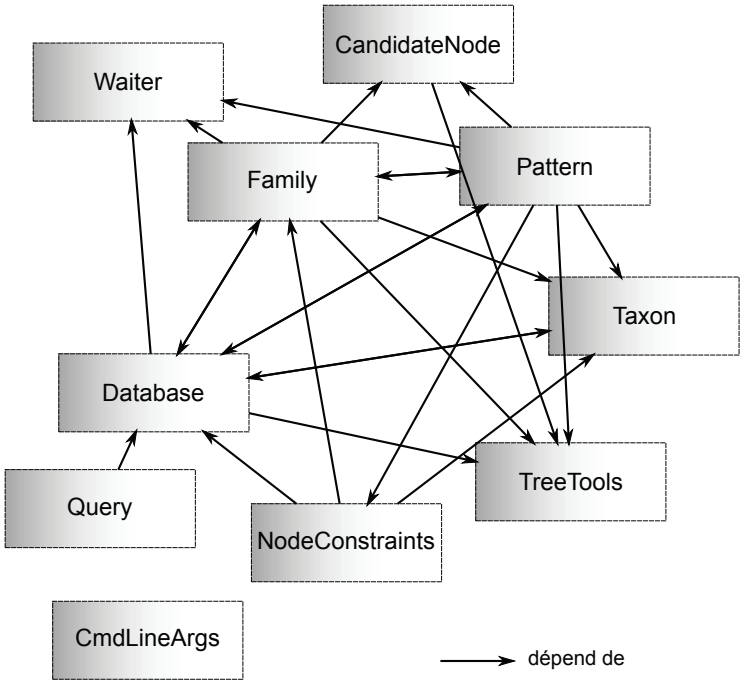


FIGURE 3.1 – **Les classes de TPMS et leurs dépendances**. On voit ici de nombreuses flèches à double sens, ainsi que des phénomènes de cycles conduisant à des dépendances circulaires. Aucune classe de TPMS ne dépend de `CmdLineArgs`, elle n'est utilisée que par les exécutables de la suite.

3.1.3 Utilisation de bibliothèques externes

Le langage C++, agrémenté de la bibliothèque standard, aurait pu suffire à lui seul pour programmer TPMS. Mais certains objets qu'il aurait alors fallu programmer ont déjà été conçus dans d'autres bibliothèques, et vu qu'il convient de ne pas réinventer la roue, TPMS les utilise. Cela permet de concentrer les efforts de développement sur les méthodes phylogénétiques à proprement parler. Les deux bibliothèques utilisées sont Bio++ [28, 49] pour la Bioinformatique, et Boost[6] pour des aspects plus généraux.

L'utilisation de bibliothèques externes est à double tranchant. Elle permet d'un côté d'économiser du temps de développement, de se protéger des bugs en utilisant du code éprouvé, mais de l'autre, elle oblige les utilisateurs finaux et les développeurs à télécharger ces bibliothèques. Une contrainte supplémentaire est que les différentes versions de ces bibliothèques sont parfois incompatibles entre elles, et le programme

[6]Boost C++ source libraries, `http://www.boost.org`

qui les utilise est de fait lié à une de ces versions. Il reste néanmoins possible d'inclure les bibliothèques dans le fichier binaire final. Néanmoins cette pratique est déconseillée à plusieurs titres : elle augmente sensiblement la taille du programme[7], et casse le principe de modularité du système[8]. De plus, une telle intégration n'est pas toujours possible pour des raisons de licence.

3.1.3.1 Bio++

Bio++ est une bibliothèque destinée à faciliter l'utilisation et l'implémentation de méthodes bioinformatiques en C++ [28, 49]. Elle permet essentiellement de prendre en charge des séquences, des arbres, des modèles d'évolution, et des objets de la génétique des populations. La plus grande partie des développements concernent la phylogénie, au travers de la sous-bibliothèque `bppphyl`. Bio++ apporte ici tant des outils pour gérer les arbres et leur apporter des modifications basiques, que des modèles évolutifs utilisables pour la construction d'arbres et l'inférence de séquences ancestrales. Dans TPMS, je me suis contenté des fonctions de manipulation d'arbres, car les arbres que j'utilise ont déjà été calculés par ailleurs.

Bio++ utilise un modèle d'arbre centré sur la classe d'objet *nœud*. Chaque nœud a numéro d'identifiant (ID), un nom, un pointeur vers le nœud père, et des pointeurs vers les nœuds fils. La classe `arbre` elle-même contient des informations telles que le nom et le nœud racine. Elle propose de nombreuses méthodes pour accéder à des nœuds et obtenir des paramètres de ces nœuds, comme la distance au nœud père, la valeur du soutien statistique, ansi que d'autres paramètres définissables à l'envi. D'autres méthodes encore permettent d'agir sur la topologie de l'arbre : le raciner, le déraciner, changer le groupe externe, mais aussi inverser des nœuds par exemple. Toutes ces fonctions sont écrites de telle sorte que la cohérence de l'arbre soit conservée, et que l'opération se fasse de la manière la plus optimisée possible. La cohérence de l'arbre est une chose très importante : il s'agit à chaque instant d'être certain que l'arbre sur lequel on travaille est bien constitué de nœuds n_i qui ont tous un père p_i, et dont chacun des fils f_i^n a bien n_i comme père. Cette propriété est importante car TPMS contient un

[7] Cette considération peut être utile avec de très grandes bibliothèques de plusieurs centaines de Mio ou bien sur des systèmes embarqués, ou encore pour la mise en mémoire cache des bibliothèques.

[8] Idéalement, une bibliothèque n'est présente qu'une fois sur un système. Cela permet, si nécessaire, de la mettre à jour grâce à une opération unique, souvent effectuée par un gestionnaire de paquets logiciels : les bugs et failles diverses sont ainsi facilement corrigés.

grand nombre de fonctions qui font une exploration récursive des nœuds d'un arbre, et ces processus nécessitent de faire confiance à une topologie intègre de l'arbre.

3.1.3.2 Boost

Boost est une collection de bibliothèques dont le but est d'étendre les fonctions du langage C++. Sa portée est généraliste et n'a pas d'application directe en bioinformatique. Elle s'ajoute à la *Standard Template Library* (STL), bibliothèque de base en C++. Elle se différencie de cette dernière par le fait que Boost n'est pas intégrée par défaut dans les systèmes d'exploitation et les environnements de développement. Cependant, elles sont liées, puisque les nouveautés testées dans Boost sont régulièrement intégrées dans la STL.

Boost apporte de nombreuses améliorations à la bibliothèque standard. TPMS l'utilise essentiellement pour deux modules : l'accès simplifié au système de fichier, et la gestion des processus légers (*threads*), qui sont une des façons de paralléliser un calcul.

3.1.4 Parallélisme

3.1.4.1 Généralités

Sur un ordinateur, les calculs sont effectués par le processeur central, aussi appelé CPU pour *Central Process Unit*. En 1975, Gordon Moore a énoncé le principe suivant : « le nombre de transistors sur un microprocesseur double tous les deux ans », la puissance des processeurs augmentant en fonction du nombre de transistors. Ce principe, bien que totalement empirique, s'est avéré exact jusqu'au début des années 2000. La miniaturisation des transistors a depuis atteint des limites physiques : des gravures plus fines impliquent un échauffement et une consommation électriques incompatibles avec une utilisation normale. La seule solution pour qu'un ordinateur gagne en puissance est de placer plusieurs cœurs, c'est-à-dire plusieurs unités de calcul par processeur, voire plusieurs processeurs, dans la même machine. Il est même fréquent, dans le domaine de la recherche scientifique, de regrouper plusieurs machines en grappes de calculs. Les utilisateurs ont à disposition de nombreux cœurs de calcul qui, combinés, peuvent fournir une puissance considérable.

Un programme qui veut tirer profit du regain de puissance apporté par la présence

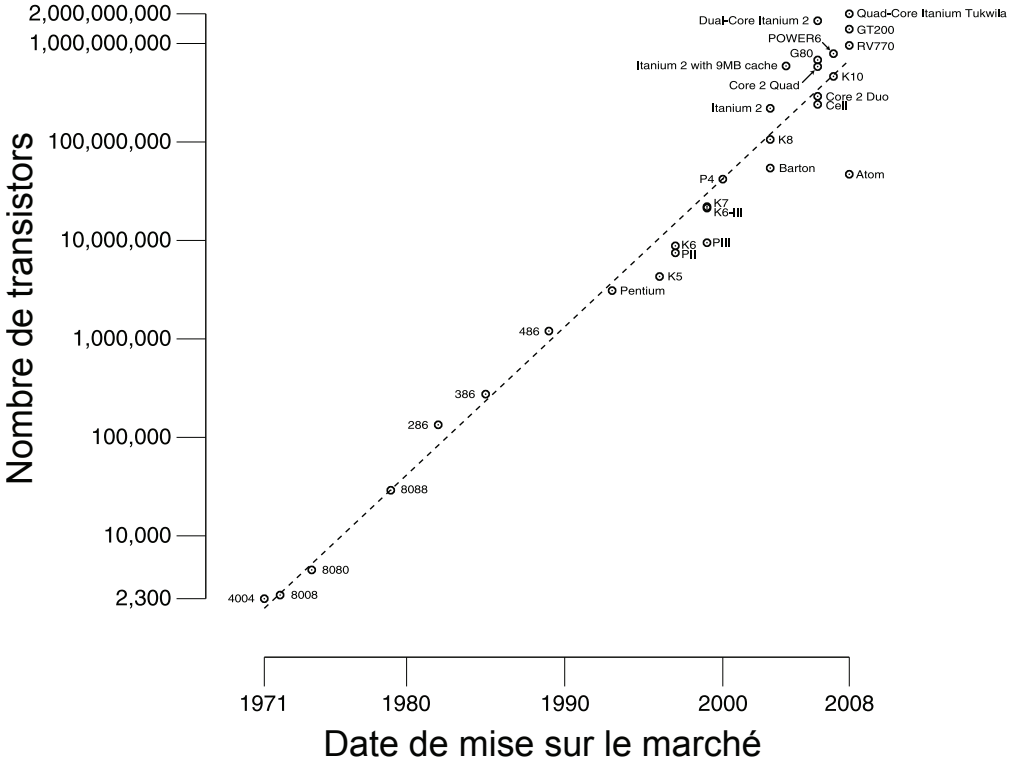


FIGURE 3.2 – **La loi de Moore entre 1971 à 2008.** Nombre de transistors par processeurs en fonction du temps. La droite en pointillés représente la progression du nombre de transistors prévue par la loi de Moore. Les différents points sont les processeurs effectivement commercialisés entre 1971 et 2008. On voit que les prédictions de Moore correspondent bien à la réalité des années 1970 à 2000.

d'unités multiples de calcul doit pouvoir exécuter ses opérations en parallèle. Il existe plusieurs moyens de paralléliser des instructions :

- le programme est exécuté plusieurs fois indépendemment sur des parties différentes du jeu de données,
- le programme est capable de découper le travail de calcul et lance en cascade des sous-programmes, éventuellement sur des machines différentes, qui communiquent entre eux ;
- le programme est capable d'exécuter des processus légers, des *threads*, qui partagent le travail sur la même machine, et utilisent une mémoire commune, et des cœurs ou processeurs différents.

3.1.4.2 Gain de performances et concurrence

C'est l'approche par *threads* qui a été retenue pour TPMS. Les différents calculs exécutés, découlant des algorithmes décrits dans ce chapitre, sont indépendants et propres à chaque famille. Cela rend la conception du parallélisme très simple, il n'y a aucune interdépendance à gérer : à aucun moment le calcul effectué sur une famille de gènes ne doit attendre la fin de l'exécution d'un autre calcul sur une autre famille de gènes. Il suffit donc de faire des groupes de familles, chaque groupe étant exécuté par un *thread* différent. L'utilisateur, lors du lancement du programme, spécifie, *via* le paramètre `-threads` le nombre de *threads* à exécuter.

Le gain de temps décroit quasi-linéairement avec le nombre de *threads*. En effet, bien que le temps de calcul soit réellement divisé par le nombre de processus légers, la mémoire et son accès sont partagés, et l'accès au disque est également partagé entre tous ces fils de calculs. Les différents calculs à exécuter manipulent des données dans la mémoire vive. Bien que ces données soient dans des zones différentes de la mémoire, le *bus* mémoire, c'est-à-dire le canal de communication entre les processeurs et la mémoire, a une taille finie où un certain nombre d'informations peuvent circuler en un temps donné. Cette limite n'est souvent pas pénalisante, car les informations à écrire sont peu nombreuses au regard des calculs à effectuer. Les accès aux unités de stockage comme les disques durs, en lecture ou écriture, sont beaucoup plus lents que les accès à la mémoire vive. Le partage de cette ressource pourrait s'avérer pénalisant. Mais mis à part le chargement initial des arbres et l'enregistrement final des résultats, les différents algorithmes de TPMS n'accèdent pas au stockage de masse.

3.1.5 Programmation récursive

Une partie importante du travail des chargements d'arbres et même de recherche de motifs ou d'autres types d'opérations effectuées sur les arbres par TPMS utilise une programmation récursive.

3.1.5.1 Présentation générale

Une caractéristique évidente d'une fonction récursive est que cette fonction s'appelle elle-même. Cette caractéristique est le fait d'une décomposition d'un problème en

sous-problèmes : on demande à la fonction de traiter un grand problème qu'il est impossible de résoudre en une passe. La solution passe donc par l'appel par la fonction d'elle-même sur des sous-problèmes du problème traité.

Un exemple facilement accessible du raisonnement récursif est la somme d'un ensemble de nombres. Soit S la somme suivante de 5 entiers :

$4+9+1+42+0+3$

Il n'est pas possible de calculer cette somme en une seule étape. La seule chose que nous puissions faire est la somme de deux entiers. La somme de deux entiers étant un entier, il est possible d'écrire la somme S de la façon suivante :

```
somme(4,somme(9,somme(1,somme(42,somme(0,3)))))
```

Ainsi, à chaque étape, la fonction somme traite deux entiers et peut les aditionner. Mais il faut distinguer deux cas :

- le cas `somme(0,3)`, seul cas où on additionne réellement deux entiers, immédiatement accessibles sans calcul supplémentaire. On l'appelle **le cas de base**.
- les quatre autres cas, où la somme ne peut se faire qu'après un appel imbriqué à la fonction somme. Ce sont **les cas récursifs**.

On peut écrire la fonction `somme` comme dans l'Algorithme 1.

Algorithme 1 Version récursive de la fonction somme

```
fonction SOMME(ensemble)
    si cardinal(ensemble) > 2 alors
      retourne ensemble.premierElement + SOMME(ensemble.derniersElements)
    sinon
      retourne ensemble.premierElement + ensemble.secondElement
    fin si
fin fonction
```

3.1.5.2 Applications à la phylogénie

L'exploration de structures arborescentes peut devenir très puissante en utilisant la récursivité. Dans l'exemple précédent, il aurait été possible de simuler une approche

récursive à l'aide d'une boucle, alors que pour l'exploration d'un arbre, il est impossible d'échapper à une fonction réellement récursive.

Dans TPMS, une fonction permet de lister les noms de toutes les feuilles d'un sous-arbre. Appelons-la `feuilles(noeud)`. Elle prend en entrée un nœud, racine du sous-arbre dont on veut obtenir le nom des feuilles, et donne en sortie une liste de chaînes que sont les noms des feuilles. Rappelons qu'à chaque instant, on peut accéder au père d'un nœud et à ses fils.

Comme pour toute fonction récursive, il faut déterminer le cas de base, et le cas récursif. Le cas de base est ici la feuille, car il est possible de connaître son nom directement. Le cas récursif est le nœud interne, car il est impossible de déterminer directement le nom des feuilles qu'il porte. La fonction va ainsi décomposer le sous-arbre en sous-arbres plus petits, jusqu'à tomber sur les feuilles.

Cette fonction peut être codée à la manière de l'Algorithme 2.

Algorithme 2 Cette fonction retourne toutes les feuilles contenues sous un nœud.

```
fonction FEUILLES(nœud)
    si type(nœud) == interne alors
        listeFeuilles ← liste vide
        pour tout fils de nœud faire
            listeFeuilles.ajoute(FEUILLES(fils))
        fin pour
      retourne listeFeuille
    sinon
        retourne singleton contenant nom(nœud)
    fin si
fin fonction
```

3.1.5.3 Débordement de pile

Bien qu'étant une approche puissante, la programmation récursive présente un risque de blocage du programme. Si la profondeur à explorer est trop importante, on peut aboutir à un *débordement de pile*.

La pile d'exécution est l'espace mémoire où sont stockées les données des fonctions. Sa taille est limitée par le système dès le lancement du programme. Un programme débute par l'exécution de la fonction principale, appelée *main* en C/C++.

Cette fonction stocke ses variables sur la pile, puis lance une fonction f. Dans la pile, se trouvent donc les données de la fonction principale, qui doivent être conservées pour la suite, puis les données de la fonction f, au dessus. La fonction f s'exécute, se termine, retourne une valeur : ses données ne sont alors plus utiles. La partie la plus haute de la pile est supprimée. À nouveau, seules les données de la fonction principale sont sur la pile. On dit qu'une pile d'exécution fonctionne en mode LIFO (*Last In First Out*), car ce sont les éléments qui y ont été ajoutés les derniers qui sont supprimés les premiers.

Quand des fonctions sont imbriquées, la pile augmente d'autant de niveaux. La taille de chaque niveau est déterminée par le nombre et la taille des variables de la fonction. Si le nombre de récursions et/ou la taille des données sont trop importants, la pile débordera. Le programme sera interrompu et les données perdues.

Il convient alors de :

- limiter l'empreinte mémoire de chaque itération, par exemple en faisant appel à l'allocation dynamique (les objets sont ainsi créés en dehors de la pile),
- limiter la profondeur maximale des itérations.

Pour TPMS, l'allocation dynamique a suffi pour permettre de prendre en charge des arbres extrêmement profonds : certaines collections contiennent des arbre d'une profondeur maximale de plusieurs centaines de nœuds, et leur chargement ne pose pas de problème.

3.2 Chargement des données et gestion de la taxonomie

3.2.1 Stockage des familles de gènes

TPMS utilise le format de collection du programme RAP [27] : les familles de gènes sont stockées dans un fichier texte en clair, c'est-à-dire éditable avec un éditeur de texte, et avec un balisage très simple et concis. Il contient un arbre d'espèces de référence, et est suivi des familles de gènes. Chaque famille a un nom, une liste de noms de séquences, puis l'arbre phylogénétique des séquences optionnellement annoté. Les annotations éventuelles concernent la nature du nœud : spéciation ou duplication. La

présence de telles annotations requiert une réconciliation préalable entre les arbres de gènes et l'arbre d'espèce.

L'arbre d'espèces permet de définir les relations phylogénétiques supposées ou connues entre les différentes espèces dont les séquences sont présentes dans les différents arbres de gènes de la collection. Il permet, lors des requêtes et des calculs qui seront fait sur les arbres de gènes, d'utiliser la notion de taxon. En effet, alors que les feuilles portent les noms des espèces, les nœuds internes sont nommés en fonction du groupe taxonomique associé aux espèces qu'ils contiennent. La racine de l'arbre est également concernée : elle permet de désigner toutes les espèces que contient l'arbre, par exemple *Cellular organisms* si l'arbre contient des espèces des trois domaines, ou plus précicément *Agrobacterium*, si la collection ne contient que des bactéries de ce genre. Les noms d'espèces et de taxons sont indiqués entre guillemets, et ils tolèrent les synonymes, séparés par des barres obliques. Par exemple, pour coder *Escherichia coli* souche K12 sous-type W311, on peut écrire :

```
"ESCHERICHIA COLI W3110"/"ESCHERICHIA COLI STR. K12 SUBSTR. W3110"/
"ESCHERICHIA COLI STR.W3110"/"ESCHERICHIA COLI STRAIN W3110"
```

L'arbre des espèces peut contenir des nœuds non résolus (multifurcations). Ce qui n'est pas le cas des arbres de gènes. En effet l'algorithme de recherche de motifs dans les arbres binaires non ordonnés ne fonctionne par définition que dans des abres binaires, c'est à dire résolus [62].

3.2.2 Chargement

Pour être exploitées, les collections de familles de gènes sont intégralement chargées en mémoire vive : une fonction transforme les chaînes au format Newick vers une structure d'objets `bpp::Node` contenus dans des objets `bpp::TreeTemplate<N>`[9]. Cela suppose une machine ayant une certaine quantité de mémoire. Pour les bases HOGENOM [97], un peu plus de 4 Go de mémoire sont nécessaires. Cette taille est supérieure à celle du fichier d'où sont extraits les arbres (230 Mo) et ceci pour deux raisons. Tout d'abord, tandis que pour chaque nœud le fichier stocke un nom, qui est vide pour les nœuds internes d'un arbre de gènes, le programmme stocke un objet nœud, avec

[9] Voir `http://biopp.univ-montp2.fr/Documents/ClassDocumentation/bpp-phyl/html/`

un espace mémoire réservé pour chacun de ses attributs possibles. Ensuite, de nombreux calculs peuvent être faits en prévision de requêtes ultérieures, et les résultats de ces calculs prennent de l'espace supplémentaire. Par exemple, on peut vouloir calculer, pour chaque nœud interne d'un arbre de gènes, le groupe taxonomique le plus récent (le plus proche des feuilles dans l'arbre des espèces) qui contient toutes les séquences présentes sous ce nœud.

3.2.3 Représentation formelle de la taxonomie

À chaque fois qu'un programme de la suite est lancé sur une collection, la première étape est le chargement de l'arbre d'espèces. C'est à ce moment que les informations concernant les groupes taxonomiques, les espèces, et les liens qui les lient sont initialisées. Cette interprétation de l'arbre des espèces ne consiste pas seulement à copier en mémoire l'arbre Newick sous forme d'objet Bio++. Ce chargement est accompagné de la création des objets `Taxon`, décrite ci-après.

3.2.3.1 La classe `Taxon`

Pour chaque groupe taxonomique (qu'il ait un nom ou pas), comme pour chaque espèce, un objet de la classe `Taxon` est instancié. Il contient le nom le cas échéant, le groupe taxonomique père (sauf pour le nœud racine de l'arbre des espèces qui n'a pas de père par définition), la liste des taxons ancêtres, et la liste des taxons descendants. Ces listes d'ancêtres et de descendants sont des listes ordonnées, de l'objet `set` de la bibliothèque standard C++. On peut se poser la question de l'utilité d'une telle indexation. En effet, pourquoi référencer pour chaque taxon les ancêtres, les descendants, et le père, alors que ces informations sont accessibles directement dans l'arbre, en utilisant une fonction récursive ? Il y a une redondance des informations.

Mais bien que les informations dans l'arbre et dans les objets de la classe `Taxon` soient identiques, elles ne sont pas du tout stockées de la même façon. Dans la classe `Taxon`, c'est une liste ordonnée (un *set* en C++) qui est utilisée. Cette structure de données permet de rechercher très rapidement un élément de la liste, par dichotomie. Il y a donc une petite perte de mémoire, et de rapidité à ce moment-là, mais cela permet de faire des recherches très rapides par la suite.

3.2.4 Assignation taxonomique des nœuds d'arbres de gènes

Tout au long de son fonctionnement, TPMS s'appuie sur l'assignation taxonomique des nœuds internes. Cette assignation est réalisée de nombreuse fois, et doit être refaite chaque fois qu'une modification dans l'arbre implique potentiellement un changement de taxonomie : un soin particulier a donc été apporté à son optimisation. La fonction assurant cette opération est décrite ci-dessous.

Cette fonction a pour but d'annoter le nœud d'un arbre de gènes avec le plus petit groupe taxonomique qui englobe toutes les espèces présentes aux feuilles du sous-arbre, le dernier ancêtre commun (ou *Last Common Ancestor* — LCA). Elle est souvent utilisée dans TPMS, que ce soit pour annoter un arbre entier, ou localement pour voir les conséquences d'une modification de l'arbre sur la taxonomie.

Comme indiqué en 3.2.3.1, la classe `Taxon` permet de connaître pour chaque taxon, la liste de ses descendants et ascendants, ainsi que son groupe taxonomique d'appartenance.

La fonction prend en entrée une liste de `Taxons`. Elle choisit arbitrairement un de ces taxons (en l'occurrence le premier), puis une boucle remonte progressivement dans l'arbre des espèces et s'arrête quand le taxon courant les contient tous.

Algorithme 3 Fonction FINDLCA qui identifie le dernier ancêtre commun d'une liste d'espèces

```
fonction FINDLCA(listeTaxons)
    ancetreCourant ← listeTaxons.premier()
    tant que NON ancetreCourant.contientTous(listeTaxon) AND NON ancetre-
Courant.estRacine() faire
        ancetreCourant ← ancetreCourant.père()
    fin tant que
    si ancetreCourant.contientTous(listeTaxons) alors retourne ancetreCourant
    sinon
        retourne erreur
    fin si
fin fonction
```

La fonction *contientTous* retourne une valeur logique (Vrai/Faux).

Algorithme 4 Fonction CONTIENTTOUS retourne vrai si tous les éléments de `listeTaxons` sont contenus dans `taxonAncetre`

```
fonction CONTIENTTOUS(taxonAncetre,listeTaxons)
    resultat ← Vrai
    pour tout taxonCourant in listeTaxons faire
        resultat ← (taxonCourant ∈ taxonAncetre.descendants)
        si NON resultat alors sortie
        fin si
    fin pour
fin fonction
```

3.3 Recherche de motifs d'arbres : le *tree pattern matching*

3.3.1 Recherche de motifs dans les arbres binaires non ordonnés

La recherche de motifs d'arbres, dite *tree pattern matching*, est basée sur l'algorithme de Pekka Kilpeläinen [62]. Cet algorithme permet de chercher récursivement un arbre motif dans un arbre cible. Il a été implémenté pour la première fois dans une optique de phylogénie dans les programmes TQuery/FamFetch [27], le premier étant la partie serveur du second qui permet l'interrogation en ligne de bases. Le défaut majeur de ce programme est qu'il ne permet pas à un utilisateur de créer sa propre collection : le seul programme utilisé est la partie client qui ne permet que d'accéder à des banques choisies sur le serveur du PBIL[10] de l'Université Claude Bernard Lyon 1, même s'il est théoriquement possible d'installer son propre serveur.

3.3.2 Langage d'interrogation

Pour interroger les collections, il a fallu créer un formalisme de requêtage. Il est basé sur Newick, qui utilise comme codage les parenthèses et les virgules. Il a été enrichi de signes supplémentaires afin de permettre d'ajouter des contraintes plus complexes que le branchement d'espèces. Comme pour tout arbre Newick, une requête TPMS doit être terminée avec le caractère *point-virgule*.

On utilise la possibilité offerte par le format Newick d'avoir potentiellement des

[10] Pôle Bio-Informatique Lyonnais, `http://pbil.univ-lyon1.fr`.

noms à tous les nœuds (nœuds internes ou feuilles). Chaque nœud peut donc se voir associer une chaîne dans laquelle l'utilisateur pourra exprimer sa requête. Une chaîne se décompose en trois parties séparées par des barres obliques (*slash*). La première partie contient les options, la deuxième la contrainte taxonomique de sous-arbre, et la troisième la contrainte taxonomique de feuille. Ces trois parties sont l'objet des explications suivantes.

3.3.2.1 Recherche simple d'espèces

La requête la plus simple permet de sélectionner, dans une collection, les arbres qui contiennent des feuilles d'espèces données agencées selon une certaine topologie. Par exemple, la requête suivante :

```
(//Mus musculus,(//Homo sapiens,//Pan troglodytes));
```

va permettre de retenir des arbres dans lesquels des séquences d'homme et de chimpanzé sont cousines, et une séquence de souris se situe à l'extérieur du sous-arbre qui engendre homme et chimpanzé. Elle est schématisée dans la figure 3.3, et un exemple d'arbre de gène qui valide ce motif est donné en figure 3.4.

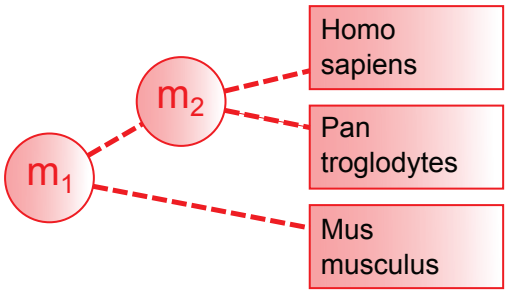


FIGURE 3.3 – **Un motif d'arbre phylogénétique avec trois espèces.** Le motif illustré correspond à la chaîne `(//Mus musculus,(//Homo sapiens,//Pan troglodytes));`. Les pointillés représentent le fait que les liens entre les différents nœuds ne sont pas forcément directs (voir le paragraphe 3.3.2.3).

Cette requête n'est pas très informative à plusieurs titres :

- le sous-arbre contenant des séquences de *Homo sapiens* et *Pan troglodytes* est autorisé à contenir une séquence de souris ;

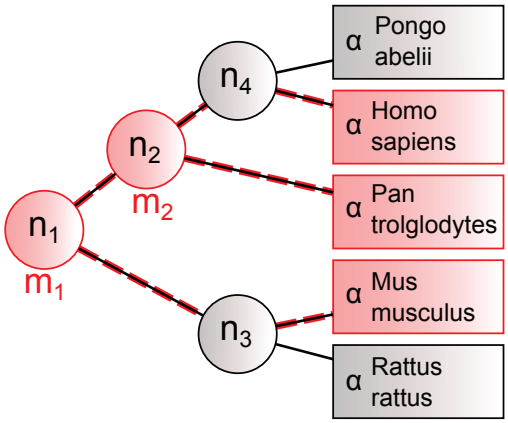


FIGURE 3.4 – **Arbre de gènes de la famille α qui valide le motif** `(//Mus musculus,(//Homo sapiens,//Pan troglodytes));`

- il peut exister, dans le sous-arbre contenant homme, chimpanzé et souris, des séquences d'homme et de chimpanzé qui se situent à l'extérieur du sous-arbre contenant les séquences d'homme et de chimpanzé détectées par le programme ;
- les séquences d'homme et de chimpanzé peuvent être topologiquement très éloignées dans un arbre, et cela n'empêchera pas l'arbre d'être sélectionné, pourvu qu'une séquence de souris se trouve à l'extérieur du sous-arbre les contenant.

TPMS prévoit de restreindre les arbres sélectionnés, par exemple en utilisant une contrainte taxonomique de sous-arbre.

3.3.2.2 Contrainte taxonomique de sous-arbre

Cette contrainte, qui correspond à la deuxième partie d'une chaîne de requête, permet de limiter la présence de séquences appartenant à certaines espèces dans un sous-arbre. En partant du motif précédent, on peut vouloir ajouter une contrainte sur le nœud qui relie l'homme et le chimpanzé, de la façon suivante :

```
(//Mus musculus,(//Homo sapiens,//Pan troglodytes)/Primates);
```

Cela a pour effet de limiter aux primates les espèces autorisées dans le sous-arbre contenant l'homme et le chimpanzé. Les arbres sélectionnés par ce motif ne peuvent pas contenir une espèce n'appartenant pas au groupe des primates à partir du der-

nier ancêtre commun à l'homme et au chimpanzé, comme indiqué sur la figure 3.5. Contrairement à la recherche simple d'espèces, la contrainte de sous-arbre est stricte : si une seule séquence de l'arbre cible contient un non-primate dans le sous-arbre concerné, l'arbre est rejeté. La figure 3.6 donne un exemple d'un arbre de gènes qui valide le motif donné dans la figure précédente.

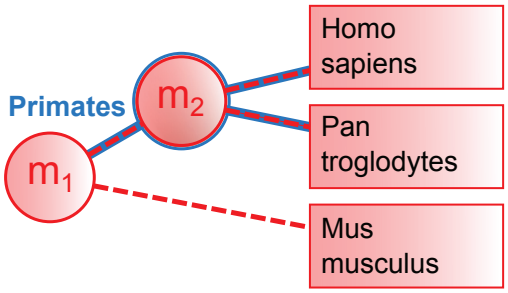


FIGURE 3.5 – **Un motif d'arbre phylogénétique avec contrainte de sous-arbre.** Le motif illustré correspond à la chaîne `(//Mus musculus,(//Homo sapiens,//Pan troglodytes)/Primates);`. Les traits sur fond bleu indiquent la partie du motif où toute espèce n'appartenant pas aux primates conduit au rejet de l'arbre.

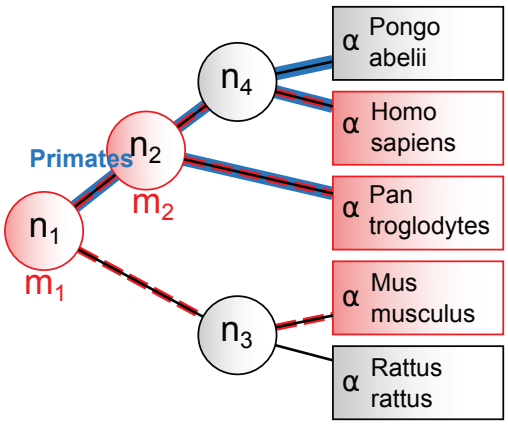


FIGURE 3.6 – **Arbre d'un gène α validant une contrainte de sous-arbre.** Le motif utilisé est celui de la figure 3.5. Sous la contrainte de sous-arbre, en bleu, il n'y a que des primates (orang-outan, homme et chimpazé), et la contrainte classique en rouge est respectée (homme frère de chimpanzé, tous deux en groupe frère d'une souris).

Comme indiqué sur la figure 3.7, la contrainte de sous-arbre concerne tout le sous-arbre sélectionné depuis le père. On peut également vouloir faire une demande plus

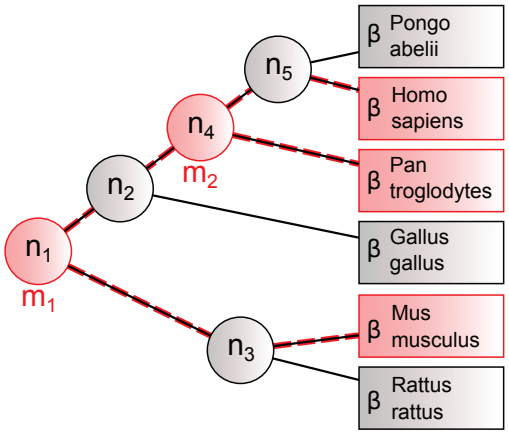


FIGURE 3.7 – **Arbre de gènes β ne validant pas un motif de sous-arbre.** Cet arbre ne valide pas le motif indiqué sur la figure 3.5. Les feuilles exigées sont bien présentes (motif représenté en rouge), mais la contrainte de sous-arbre n'est pas respectée. En effet, il aurait fallu que le sous-arbre généré par le fils du nœud de l'arbre associé au nœud de motif m_1 (n_1) conduisant au nœud de l'arbre associé au nœud de motif m_2 (n_4), c'est-à-dire n_2, ne contienne que des séquences du genre Primate. Ce qui n'est pas le cas, puisque ce sous-arbre contient une séquence de poule.

précise, en requérant que les feuilles de l'homme et du chimpanzé soient sœurs, c'est ce que permet de faire le lien direct.

3.3.2.3 Lien direct

Dire que deux feuilles sont sœurs est équivalent à dire que les deux ont un nœud père unique. Cela revient, dans la description du motif, à exiger que chacune des feuilles soit liées directement à leur père. Alors que le motif `(//Homo sapiens,//Pan troglodytes)` décrit un sous-arbre contenant d'une part une séquence humaine, et d'autre part une séquence simienne, le motif `(!//Homo sapiens,!//Pan troglodytes)` décrit quant à lui un nœud dont les deux fils sont des séquences d'une part d'homme et d'autre part de chimpanzé.

3.3.2.4 Soutien statistique

En phylogénie, aucune interprétation ne peut être tirée d'une topologie sans connaître le soutien statistique aux branches qui la constituent. Il est donc fort utile de pouvoir ne retenir, lors d'une requête, que les arbres qui présentent des valeurs de soutien

suffisantes aux branches d'intérêt. Pour ce faire, on utilise là encore la première partie de la chaîne de requête qui correspond aux options. La valeur minimale souhaitée doit être précédée du signe dollar. Par exemple :

```
(Ratus ratus,(//Homo sapiens,//Pan troglodytes)$90);
```

permet d'exiger que la branche qui sépare le sous-arbre contenant l'homme et le chimpanzé de celui du rat soit soutenue à 90%.

3.3.2.5 Type de nœud

Une dernière option permet de spécifier la nature d'un nœud. Ce critère est utilisable sur les arbres réconciliés, dans lesquels il est fait une différence entre les évènements de spéciation ou de duplication, respectivement identifiés par les lettres S et D. Par exemple, dans ce motif :

```
(Ratus ratus,(//Homo sapiens,//Pan troglodytes)S$90);
```

on souhaite que les séquences correspondant à l'homme et au chimpanzé aient pour ancêtre commun un nœud de spéciation.

3.3.2.6 Noms d'espèces et de groupes taxonomiques utilisables

À chaque fois qu'il doit nommer des espèces, que ce soit pour une recherche de feuille ou bien une contrainte de sous-arbre, l'utilisateur peut utiliser toute espèce ou groupe taxonomique présent dans l'arbre des espèces. Il est également possible de combiner ces groupes et ces espèces par des opérations ensemblistes. Par exemple, si dans un sous-arbre on veut limiter les espèces autorisées aux γ-protéobactéries, mais interdire les *Escherichia coli*, on pourra indiquer `Gammaproteobaceria-Escherichia coli`. Il est possible d'enchaîner ces opérations, et d'aboutir à des sélections d'espèces très fines, en ajoutant des groupes avec le signe + ou en en soustrayant avec le signe −.

Quelques règles s'appliquent à ces combinaisons :

- la lecture se fait de gauche à droite ;
- une expression qui commence par le signe moins (−), est interprétée comme «Toutes les espèces sauf». Par exemple, si l'arbre des espèces a pour racine le groupe taxonomique «Cellular organism», l'expression `-Bacteria` est interprété comme `Cellular organism-Bacteria` ;

- une expression qui ne commence pas par un signe + ou − est interprétée comme commençant par un +. Ainsi l'expression `Cellular organism-Bacteria` est lue comme `+Cellular organism-Bacteria`.

3.3.3 Algorithme

L'algorithme reprend celui de la recherche dans les arbres binaires non ordonnés [62, 27]. Il est consigné dans l'Algorithme 5 : la fonction décrite est récursive, et donc à chaque étape de la récursion, on recherche un sous-arbre du motif dans un sous-arbre de l'arbre cible (sous-arbres respectivement désignés par le nœud qui les engendre), sauf dans le cas particulier du lancement initial de cette fonction où on recherche l'intégralité du motif dans l'intégralité de l'arbre de gènes.

Algorithme 5 Recherche un nœud de motif `pattern` dans un nœud `cible`.

Sortie : booleen : est-ce que le pattern est inclus dans la cible

```
1:  fonction PATTERNDANSCIBLE(pattern,cible)
2:      si pattern.estFeuille() et cible.estFeuille() alors
3:          retourne cible.Taxon ∈ pattern.EspecesAutorisées
4:      sinon si non pattern.estFeuille() et cible.estFeuille() alors
5:          retourne Faux    → on ne peut pas chercher un sous-arbre dans une feuille
6:      sinon si pattern.estFeuille() et non cible.estFeuille() alors
7:          retourne patternDansCible(pattern, cible.premierFils) ou patternDans-
    Cible(pattern, cible.secondFils)    → la feuille du pattern est peut-être dans l'un des deux
    sous-arbres fils de la cible
8:      sinon    → pattern et cible sont deux sous-arbres
9:          retourne (patternDansCible(pattern.premierFils, cible.premierFils)
    et patternDansCible(pattern.secondFils, cible.secondFils)) ou ( pat-
    ternDansCible(pattern.secondFils, cible.premierFils) et patternDans-
    Cible(pattern.premierFils, cible.secondFils))
10:         ou patternDansCible(pattern, cible.premierFils) ou patternDans-
    Cible(pattern, cible.secondFils)
11:     fin si
12: fin fonction
```

Cet algorithme a été amendé de plusieurs modifications. La plus importante est une généralisation du cas de base de la condition présentée en ligne 4 dans laquelle on interdit de rechercher un arbre dans une feuille. Cette généralisation consiste à ne pas rechercher un motif dans un arbre si le motif est plus grand que l'arbre : l'algorithme

finira forcément par aboutir à la situation testée par la ligne 4, et la fonction retournera *Faux*. Au lancement de la recherche de motifs, pour chaque nœud du motif comme de l'arbre cible, on stocke le nombre de nœuds maximum entre le nœud d'intérêt et une feuille. Cela permet de déterminer la profondeur maximale de chaque sous-arbre. On ne peut pas rechercher un motif ayant une profondeur maximale de n dans un arbre de profondeur maximale de m si $n > m$. L'indexation de la profondeur maximale sous un nœud est très rapide, car elle utilise la programmation dynamique : ce calcul est récursif, mais il suffit, lors du calcul, de mémoriser les valeurs qui sont calculées lors du retour d'une itération à son itération parente.

Autre modification substantielle : les expressions booléennes des lignes 7 et 8 ont été remplacées par des tests séparés. En effet, dans sa forme d'origine, l'algorithme doit décider si oui ou non il y a un sous-arbre correspondant au motif recherché dans un arbre cible, donc une expression booléenne avec l'opérateur *ou* suffit : dès qu'une des conditions est *vraie*, le test s'arrête et la valeur *Vrai* est renvoyée. Mais ce qui nous intéresse ici, c'est également d'identifier quels sous-arbres valident ce motif. Il faut donc impérativement tester toutes les conditions sans s'arrêter dès que la première est validée. C'est ce qui est illustré dans l'Algorithme 6. Cela permet d'identifier des cas ou plusieurs motifs sont présents mettant en jeu des nœuds en commun, comme illustré sur la figure 3.8.

Algorithme 6 Remplacement de la ligne 7 de l'Algorithme 5

```
arbreValidé ← patternDansCible(pattern, cible.premierFils)
arbreValidé ← arbreValidé ou patternDansCible(pattern, cible.secondFils)
retourne arbreValidé
```

3.3.4 Trouver le chemin : le Petit Poucet

Cet algorithme permet de trouver les feuilles qui valident les motifs recherchés. Mais identifier ces feuilles ne suffit pas, on veut aussi pouvoir rapidement retrouver les nœuds qui ont conduit à ces feuilles. Il arrive souvent qu'un nœud conduise à plusieurs sous-arbres validant une partie du motif, tel que le nœud n_2 dans la figure 3.8. Ce nœud a plusieurs statuts : il valide la condition qui se trouve ligne 9 et la condition de la ligne 10 de l'algorithme 5. Dans ce cas de figure, retourner les sous-arbres extraits

de l'arbre d'un gène (4 sous-arbres différents dans le cas de l'exemple) n'est pas trivial, et mieux vaut utiliser un balisage des nœuds internes qui valident un motif.

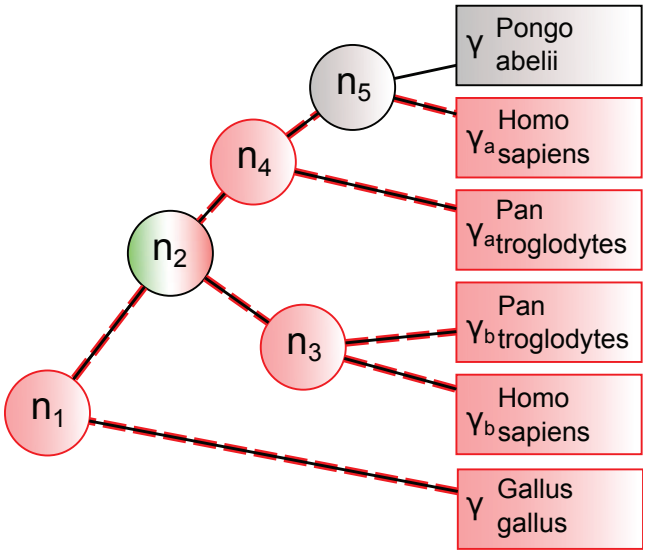


FIGURE 3.8 – **Motif trouvé plusieurs fois dans un arbre** Dans cet arbre d'un gène γ, on recherche le motif `((Pan troglodytes, Homo sapiens),Gallus gallus);`. Le sous-arbre de ce motif `(Pan troglodytes, Homo sapiens)` est présent plusieurs fois dans l'arbre, respectivement sous les nœuds n_3 et n_4. Le nœud n_2 a un statut particulier, il est à la fois un nœud qui valide `(Pan troglodytes, Homo sapiens)`, avec par exemple les feuilles γ_a Homo sapiens et γ_b Pan troglodytes, et à la fois un nœud qui mène vers des descendants n_3 et n_4 qui valident chacun respectivement ce motif.

Pour garder en mémoire les chemins empruntés, on utilise des objets de la classe `CandidateNode`. Pour une recherche d'un motif dans un arbre de gène, on va créer une arborescence de nœuds `CandidateNode` : à chaque fois qu'un nœud de l'arbre valide un nœud du motif, on crée un objet de cette classe qui lie précisément ces deux nœuds. Or, on ne peut pas savoir à l'avance si un sous-arbre va valider un motif. La solution utilisée est donc de créer systématiquement un nœud candidat, de le transmettre lors des appels récursifs, afin que ces appels et les sous-arbres qui y sont associés puissent s'accrocher correctement dans la hiérarchie. Ensuite, et pour chaque itération, si le retour de la fonction est *Vrai*, on valide la hiérarchie de nœuds candidats sous-jacents. Dans le cas contraire, on supprime tous les objets créés dans le cadre d'appels récursifs de l'itération courante.

3.4 Racinement automatique des arbres

3.4.1 Introduction

Le racinement, comme vu en 2.1.8, est une des problématiques de la phylogénie. Une fois l'arbre construit, il faut trouver quel est le groupe le plus externe, et placer la racine sur la branche qui sépare ce groupe. Mais il n'est pas envisageable de procéder manuellement à cette opération sur des centaines voire des milliers d'arbres de collections générés automatiquement. Nous proposons ici deux méthodes principales qui vont tenter de reproduire la logique humaine. La première se base sur un arbre taxonomique de référence, tandis que la seconde tente de diminuer le nombre d'évènements de duplication d'un arbre. Une troisième combine les deux précédentes.

3.4.2 Racinement en suivant un arbre de référence

Cette méthode nécessite un arbre des espèces, même non résolu. Bien que des branchements puissent être incertains, il existe de nombreux arbres d'espèces soutenus disponibles. Le NCBI propose [111] par exemple une taxonomie sous forme d'un arbre phylogénétique contenant 262 124 espèces, et au total 364 599 nœuds taxonomiques [89].

3.4.2.1 Principe

Sur de petits arbres, on est capable de reconnaître en quelques instants la bonne racine dans un arbre de gènes, si on connaît la phylogénie de référence. Mais dans le cas présent, il s'agit d'automatiser cette tâche : le bon sens n'a ici aucune prise. Il faut parvenir à définir en quoi l'enracinement d'un arbre est conforme à un racinement de référence.

Une des propriétés d'un arbre mal raciné est qu'il va casser des groupes taxonomiques. Comme sur la figure 3.9, cela a pour conséquence que les affectations taxonomiques attribuent des groupes taxonomiques profonds — proches de la racine — aux nœuds de l'arbre de séquences.

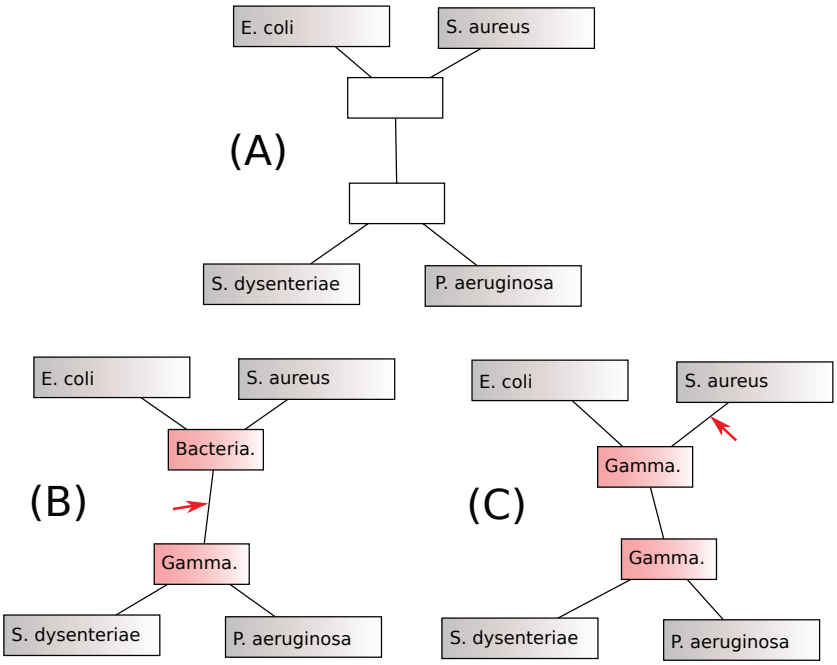


FIGURE 3.9 – **Conséquence d'un racinement sur l'affectation taxonomique des nœuds internes.** **(A)** Un arbre non raciné de quatre espèces, et de deux nœuds internes. **(B)** On place la racine au niveau de la flèche rouge, et on observe les affectations taxonomiques de chaque nœud interne. Alors que *E. coli*, *S. dysenteriae* et *P. aeruginosa* sont des *Gammaproteobaceria*, noté *Gamma.* sur la figure, ce racinement place *E. coli* de l'autre côté de la racine. Le groupe contenant *E. coli* et *S. aureus* a une affectation plus générale : *Bacteria*. **(C)** Ce nouveau racinement corrige le problème, et les deux nœuds internes sont annotés avec des groupes plus précis. C'est ce dernier racinement qui rend l'arbre plus conforme à l'arbre d'espèces.

3.4.2.2 Mesure de profondeur dans l'arbre des espèces

Cette méthode se base sur une notion de profondeur d'un nœud taxonomique dans l'arbre des espèces. On peut attribuer un entier positif à tout nœud de l'arbre des espèces. Cet entier correspond au nombre de nœuds entre la racine et le nœud du taxon considéré. Dans le reste de l'explication de cette méthode nous appellerons cet entier *distance*. Plus la distance est élevée, plus le nœud taxonomique considéré est éloigné de la racine.

3.4.2.3 Annotation de l'arbre de gènes à raciner

Pour un arbre donné, et pour une racine donnée, on annote les nœuds avec les groupes taxonomiques correspondants. Pour chaque nœud v_i, on connaît le taxon affecté Tax_i, et donc la distance de ce taxon à la racine dans l'arbre des espèces d_i. Une fois que tous les n nœuds de l'arbre de gènes sont ainsi annotés, on somme les distances, et on obtient, pour un arbre raciné R la somme $D^{(R)}$ suivante :

$$D^{(R)} = \sum_{i=1}^{n-1} d_i^{(R)} \tag{3.1}$$

3.4.2.4 Choix de la racine

TPMS va essayer les $2n-3$ racines possibles, et sélectionner celle(s) qui maximise(nt) $D^{(R)}$. Pour un arbre complètement congruent avec l'arbre des espèces, il n'y a qu'une seule racine qui maximise cette valeur. Lorsque l'arbre présente des incongruences, il y a plusieurs racines qui peuvent présenter le score le plus élevé. En cas d'ex-æquo, on choisit arbitrairement, parmi celles qui présentent le meilleur score, la racine située sur la branche la plus longue, afin d'équilibrer l'arbre.

La figure 3.10 reprend l'arbre de gènes de la figure 3.9 en utilisant le score de taxonomie.

3.4.3 Racinement par duplications

Pour cette méthode, l'arbre de référence des espèces n'est pas utilisé. Il s'agit ici de minimiser le nombre de duplications de l'arbre en les rejetant vers la racine.

3.4.3.1 Définition de l'unicité

Nous introduisons ici le concept d'*unicité*. L'unicité d'un nœud d'arbre de gène correspond au nombre de duplicats d'espèces qu'il porte. Un nœud qui aboutit à des séquences qui sont toutes d'espèces différentes porte un score de 1. Ce critère permet donc dans un premier temps de déterminer quels sont les sous-arbres d'un arbre qui ne contiennent qu'une seule séquence par espèce. Il permet aussi d'évaluer différents racinements d'un arbre en favorisant le ou les racinements qui vont maximiser le nombre de sous-arbres présentant cette caractéristique.

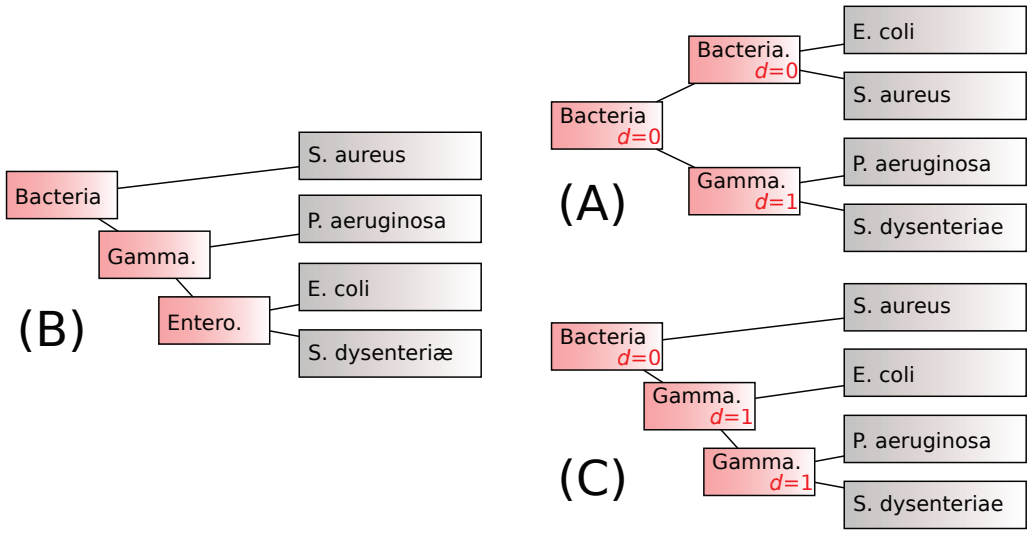


FIGURE 3.10 – **Utilisation du score de distance pour un arbre de gènes avec deux racinements.** L'arbre d'espèces de référence est représenté en **(A)** : les nœuds internes indiquent les noms des groupes taxonomiques (Bacteria, γ-proteobacteria, et Enterobacteria). Le racinement donné en **(B)** donne une somme des distances $D^{(R)} = 1+0+0 = 1$, tandis que l'enracinement en **(C)** donne lui une somme des distances $D^{(R)} = 1+1+0 = 2$. On préférera donc l'enracinement donné en **(C)**.

3.4.3.2 Annotation de l'arbre

Pour une racine donnée, on attribue à chaque nœud v_i ($i = 1,2,\ldots,n-1$), le score u_i qui est égal au produit du nombre de séquences par espèce rencontrée sur sous-arbre généré par ce nœud. Comme ce produit peut atteindre de très grands nombres pour un arbre contenant plusieurs duplications, on utilise le logarithme népérien de ce score.

Ainsi, pour l'enracinement d'un arbre, son score d'unicité $U^{(R)}$ est égal à :

$$U^{(R)} = \sum_{i=1}^{n-1} \ln\left(u_i^{(T)}\right) \tag{3.2}$$

3.4.3.3 Choix d'une racine

Comme précédemment, les $2n-3$ racines possibles de l'arbre à raciner sont testées, et TPMS garde l'arbre qui minimise la somme $U^{(R)}$. Dans le cas où plusieurs racinements auraient le même score d'unicité, la racine située dans la branche la plus longue est conservée. Dans le cas où il y a plusieurs branches de même longueur, l'algorithme

choisit arbitrairement la première des racines possibles satisfaisant ces critères.

La figure 3.11 donne un exemple de l'annotation d'un arbre par score d'unicité.

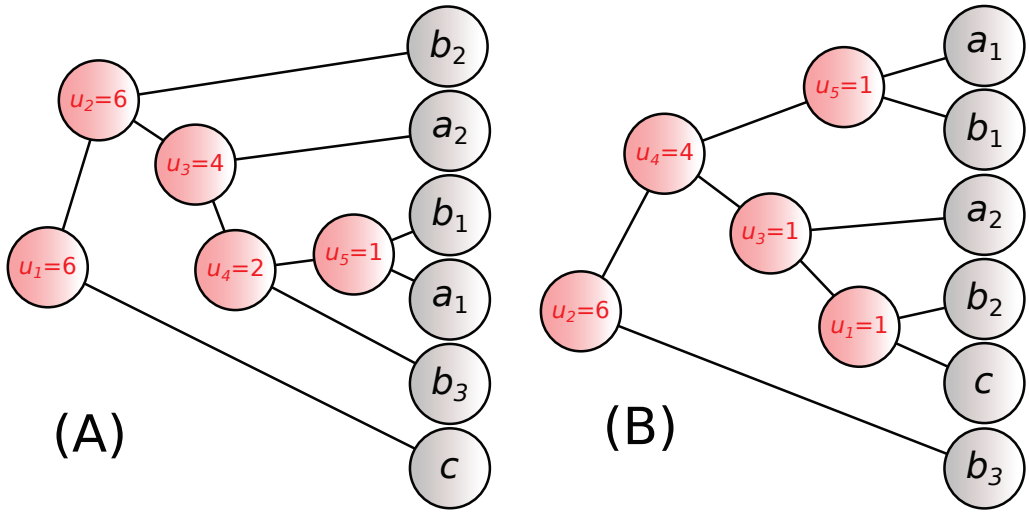


FIGURE 3.11 – **Utilisation du score d'unicité pour un arbre de gènes avec deux racinements.** Dans cet arbre, il y a six séquences a_1, a_2, b_1, b_2, b_3, and c. Les séquences a_1 et a_2 appartiennent à l'espèce A ; b_1, b_2 et b_3 à l'espèce B ; et la séquence c appartient à C. Les scores d'unicité u_n sont inscrits dans les cercles représentant les nœuds internes. Dans cet exemple, l'enracinement donné en **(A)** donne le score d'unicité pour l'arbre de $U^{(R)} = \ln(6)+\ln(6)+\ln(4)+\ln(2)+\ln(1) \approx 5,663$, tandis que l'enracinement donné en **(B)** donne le score $U^{(R)} = \ln(6)+\ln(4)+\ln(1)+\ln(1)+\ln(1) \approx 3,178$. Le racinement en **(B)** est donc préféré au racinement en **(A)**.

3.4.4 Méthode combinée

La méthode combinée utilise ces deux méthodes de racinement. Elle privilégie l'enracinement par le critère d'unicité : c'est cette première approche qui est utilisée pour isoler les racines conformes avec les hypothèses les plus parcimonieuses en terme de duplications. Dans le pire des cas pour cette méthode, il n'y a aucune duplication : la racine a un score d'unicité de $\ln(1) = 0$: toutes les racines sont alors également acceptables. Dans le meilleur des cas, il n'y a qu'une racine qui minimise le score $U^{(R)}$. Très souvent, on a plusieurs branches internes qui sont équivalentes au point de vue de ce score. Parmi les n racines de départ, on a sélectionné n_u racines avec $n_u \leq n$.

Ensuite, on applique à la sélection de racinements possibles n_u la sélection par cri-

tère de taxonomie. C'est-à-dire qu'on sélectionne, parmi ces racinements celui (ceux) qui est (sont) le (les) plus cohérent(s) avec l'arbre d'espèces. Là, on obtient n_t racines qui maximisent $U^{(R)}$, avec $n_t \leq n_u$.

Malgré l'enchaînement des deux critères, il se peut qu'il reste des ex-æquos. Dans ce cas, c'est la racine située sur la branche la plus longue qui est choisie. S'il y a plusieurs branches les plus longues, on en choisit arbitrairement une (la première rencontrée).

3.5 Détection d'incongruences

Comme vu en 2.2.2, les incongruences entre un arbre d'espèces et un arbre de gènes peuvent être, chez les procaryotes, associées à des transferts horizontaux. Plus généralement, une incongruence locale statistiquement soutenue est toujours associée à un évènement évolutif ou à un artéfact de reconstruction. Dans tous les cas, il peut être utile de faire un inventaire de ces incongruences, et c'est ce que propose un des algorithmes de TPMS.

3.5.1 Principe

Il est possible de rechercher, avec l'algorithme de recherche, un motif typique d'un transfert particulier. Par exemple, pour identifier tous les sous-arbres dans lesquels une *Escherichia coli* est nichée dans des bactéries qui ne sont pas apparentées à des *Gammaproteobaceria*, on peut interroger une collection avec le motif `((//Escherichia coli, /-Gammaproteobaceria/-Gammaproteobaceria), /-Gammaproteobaceria/-Gammaproteobaceria);` qui est représenté sur la figure 3.12.

Mais pour faire une détection systématique et automatique dans une grande collection avec de nombreuses espèces, il faudrait écrire d'innombrables requêtes, ce qui n'est pas envisageable. L'algorithme présenté ici consiste à mesurer la perturbation taxonomique éventuelle engendrée par la présence de chaque nœud. Par perturbation taxonomique induite par un nœud, on entend la conséquence sur l'affectation taxonomique de son grand-père si on supprime ce nœud. Pour cela, il faut utiliser l'assignation du LCA (*Last Common Ancestor* - ancêtre commun le plus récent entre deux séquences), dont l'algorithme est décrit plus haut[11], et qui utilise l'arbre d'espèces de

[11] Voir 3.2.4.

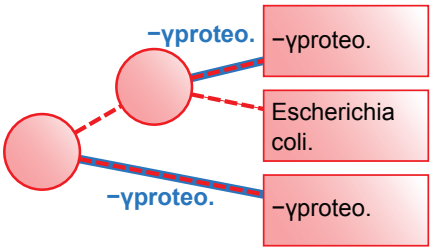


FIGURE 3.12 – **Motif de détection d'un Escherichia coli niché dans des séquences non-γproteobacteria.** Correspondance graphique de `(((//Escherichia coli, /−Gammaproteobaceria/−Gammaproteobaceria), /−Gammaproteobaceria/−Gammaproteobaceria);`

référence.

3.5.2 Algorithme

Dans un arbre de gène, on explore la perturbation induite par tous les nœuds. Pour déterminer si un nœud est perturbateur, on note l'affectation taxonomique de son grand-père, on considère temporairement que le nœud d'intérêt (et le sous-arbre qu'il engendre) n'existe plus dans la topologie. Là, on observe la nouvelle assignation taxonomique du grand-père. Si l'arbre de gènes est parfaitement congruent avec l'arbre des espèces, l'ancienne et la nouvelle assignation taxonomique sont toujours identiques.

Dès qu'il y a une incongruence apportée par un nœud (et son sous-arbre), il y a une différence d'assignation taxonomique. Tous les nœuds apportant une telle perturbation sont consignés dans un fichier de sortie, si le soutien statistique est suffisant. S'il ne l'est pas, on observe alors l'effet de la suppression temporaire du nœud d'intérêt sur les ascendants du grand-père, tant qu'il y a toujours une perturbation taxonomique induite, et jusqu'à ce que le soutien statistique soit suffisant, et que le nœud ancêtre n'est pas la racine.

3.5.3 Exemple

Un exemple du changement d'affectation taxonomique du nœud grand-père est donné sur la figure 3.13. **(A)** Un arbre d'espèces de référence. Dans chaque nœud, figure le nom du groupe taxonomique ou de l'espèce, ainsi que, en rouge, la distance d à la racine, c'est-à-dire le nombre de nœuds entre la racine et le nœud concerné. En **(B1)**,

un sous-arbre contenant une incongruence, avec *S. aureus* dans le rôle de l'intrus mal placé (nœud signalé en rouge). Dans ce sous-arbre, le groupe taxonomique associé au grand père (coloré en bleu) de *S. aureus* porte l'affectation taxonomique *Bacteria*, car le groupe *Bacteria* est le dernier ancêtre commun de tous les nœuds du sous-arbre qu'il génère, selon l'arbre des espèces. Le groupe taxonomique *Bacteria* est associé à la distance $d = 0$. En **(B2)**, on élague le nœud d'intérêt (*S. aureus*), et ce faisant, on supprime son nœud père. La racine de ce sous-arbre ne porte alors plus que deux feuilles : *E. coli* et *S. dysenteriae*. La nouvelle affectation de ce nœud devient alors *Enterobacteria*, ce qui correspond à une distance à la racine de $d = 2$, dans l'arbre des espèces. La perturbation induite par le nœud *S. aureus* est donc égal à $2 - 0 = 2$.

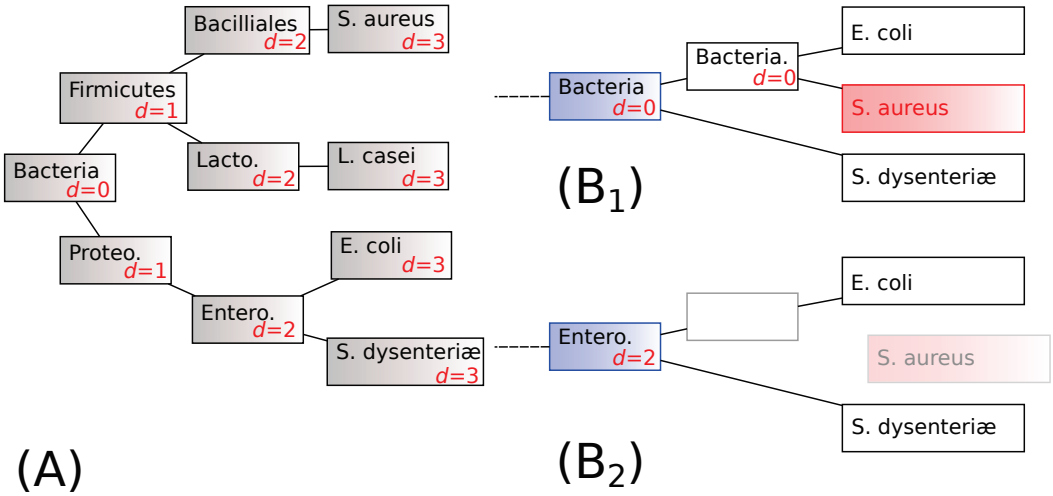


FIGURE 3.13 – **Effet d'un nœud sur l'affectation taxonomique de son grand-père.** Dans cet exemple, *Staphylococcus aureus*, une Firmicute, est groupée de manière inattendue avec *Escherichia coli*, une entérobactérie. Si on masque le nœud *S. aureus*, on aboutit à un changement d'affectation taxonomique de son nœud grand-père de Bacteria à Enterobacteria. Selon le nombre de nœuds présents entre Bacteria et Enterobacteria, l'indice de perturbation associé avec cette incongruence est de 2.

3.6 Programmes résultants et disponibilité

3.6.1 Programmes de la suite

TPMS est une suite, c'est-à-dire un ensemble de programmes. Ils sont au nombre de trois :

`tpms_mkdb` permet à l'utilisateur de créer sa propre collection de séquences. Les fichiers nécessaires sont : (1) un arbre d'espèces de référence, (2)tous les arbres de gènes de la collection, et (3) un fichier qui fait l'association entre les noms de séquences présents dans les arbres de gènes et les noms d'espèces de l'arbre d'espèces. Ce programme peut être lancé en mode assistant : il va générer un canevas de fichier correspondant à (3) : tous les noms de séquences y sont présents, et seuls les noms d'espèces restent à compléter. De plus, un module expérimental permet d'associer automatiquement une espèce à un nom de séquence si la séquence est extraite d'une banque gérée par le système ACNUC [43] (comme GenBank, SwissProt, et d'autres).

`tpms_query` est le programme qui permet d'interroger une collection avec un motif d'arbre. L'utilisateur est invité à rentrer le nom de la requête qui correspond au nom du fichier dans lequel seront mémorisés les résultats. Il entre ensuite le motif avec le formalisme décrit plus haut. Il existe plusieurs options pour demander au programme de ne retourner que les noms des familles qui correspondent, de retourner tous les motifs trouvés, ou encore de retourner tous les sous-arbres entiers qui contiennent un motif validé (sans en retirer les feuilles et nœuds qui ne correspondent pas au motif).

`tpms_computations` permet quant à lui toutes les autres opérations décrites plus haut : annotation avec le score d'unicité, racinement avec les différents critères, détection d'incongruences. L'interface permet un menu des opérations disponibles tel que montré sur la figure 3.14.

3.6.2 Disponibilité

TPMS est disponible gratuitement pour la communauté. De plus, son code source (presque 6 500 lignes) est libre et ouvert, ce qui implique que tout le monde peut ob-

```
  +---------------------------
  |        MENU
  |
  |SCORING
  |=======
  |   CU) compute unicity scores
  |   CB) compute bipartition scores
  |
  |ROOTING
  |==========
  |   RU) root with unicity criteria
  |   RT) root with taxonomic criteria
  |   RC) root with the unicity+taxonomy criteria
  |
  |MISC
  |====
  |    T) collection status
  |   XD) incongruencies detection
  |    M) help menu
  |    S) Save collection
  |   SU) Save collection with Unicity scores
  |    Q) QUIT

command? (m for help) >
```

FIGURE 3.14 – Menu de choix des options du programme `tpms_computations`.

server le code, le modifier, et le redistribuer en respectant les termes de la licence CeCill[12], transposition de la licence GNU/GPL en droit français. Le code source est disponible sur la plateforme de développement collaboratif GitHub à l'adresse `https://github.com/tbigot/tpms`, qui permet de proposer des modifications, et de signaler des dysfonctionnements. Ce site héberge également la documentation sur un wiki : `https://github.com/tbigot/tpms/wiki`. Les binaires sont disponibles sur le site officiel `http://pbil.univ-lyon1.fr/software/tpms/`.

Le programme se compile à l'aide de CMake [83] qui a pour rôle de vérifier les pré-requis à la compilation (bibliothèques externes, outils de compilation…).

[12]Signifie Ce(A)C(nrs)I(NRIA)L(ogiciel)L(ibre), car issue d'une collaboration entre CNRS, INRIA, et CEA. `http://www.cecill.info/licences/Licence_CeCILL_V2-fr.html`

4 Utilisation de tpms pour répondre à des questions biologiques

Dans ce chapitre, je vais détailler des utilisations possibles de TPMS. Ces utilisations ont pour la plupart été faites dans le cadre de collaborations avec d'autres chercheurs dont la thématique de travail était parfois éloignée du sujet principal de cette thèse. Je présente donc à chaque fois en quelques mots le contexte problématique, puis donne les résultats majeurs de l'étude.

4.1 Trouver des orthologues

Les gènes orthologues suscitent beaucoup d'intéret en bioinformatique (voir 2.1.3.1). TPMS propose plusieurs stratégies pour trouver les jeux de gènes orthologues dans les collections de familles de gènes. La première d'entre elles utilise l'interrogation explicite de la collection avec un motif précis d'espèces, tandis que la seconde utilise le score d'unicité pour extraire les plus grands arbres contenant au plus une séquence par espèce.

4.1.1 Avec topologie précise — Propriétés évolutives des espèces dioïques chez *Silene*

On sait que les gènes orthologues sont issus de spéciations. Leur histoire reflétée par la topologie de leur arbre suit donc directement l'histoire des espèces et donc l'arbre des espèces. On cherche alors, parmi les arbres de gènes, ceux qui contiennent exactement l'arbre des espèces dont nous souhaitons connaître les orthologues.

C'est la technique que nous avons utilisée dans une étude portant sur des plantes du genre *Silene* [66]. Il s'agit de déterminer des patrons d'évolution chez des *Silene*, en fonction du caractère dioïque de l'espèce de *Silene* en question (car certaines espèces de *Silene* sont dioïques, tandis que d'autres ne le sont pas). La dioécie est le fait, pour une plante, d'avoir une séparation des sexes : certains plants sont mâles, tandis que d'autres plants sont femelles. Cela ne concerne que 5 à 10% des plantes à fleurs [105, 125], les autres plantes étant hermaphrodites, c'est-à-dire que les fleurs présentent à la fois des organes mâles et femelles.

Cette propriété a plusieurs conséquences attendues : une taille efficace de population plus faible, et ainsi une sélection moins efficace. D'autre part, la sélection sexuelle dans les espèces dioïques est censée affecter préférentiellement quelques gènes, lesquels évoluent sous sélection positive. Pour vérifier ces hypothèses, de nombreuses séquences ont été utilisées. Plus particulièrement, pour tester l'hypothèse de la sélection sexuelle différente, il a fallu obtenir des jeux de gènes orthologues.

Pour ce faire, on a utilisé la banque de données SiESTa [14] qui contient environ un million de lectures de séquences d'expression de *Silene*. Parmi les espèces présentes, certaines sont dioïques : *S. latifolia*, *S. dioica*, et *S. marizii*, d'autres sont gynodioïques : *S. vulgaris* et *Dianthus superbus*, qui ne fait pas partie du genre *Silene* et qui est utilisé comme groupe externe. Le but à ce stade est d'extraire les jeux d'orthologues de cette banque. On a suivi l'enchaînement classique des opérations pour obtenir des familles de gènes et leurs arbres[1] : alignement local avec BLASTP [5], regroupement en familles d'après le résultat de BLAST avec SiLiX [86], alignement multiple au sein des séquences avec MUSCLE [29], puis établissement de la phylogénie avec FasTree [101]. On a donc à ce stade un arbre par famille. Puis les arbres ont été réconciliés en utilisant RAP [27].

[1] Voir 2.3.3.

Le but de l'opération est alors de sélectionner toutes les familles contenant des séquences respectant la topologie des espèces d'intérêt. Mais malgré des efforts faits en ce sens, des incertitudes persistent dans la phylogénie des *Silene* qui reste à ce jour non résolue [104, 82]. La figure 4.1 montre la version la plus communément acceptée de la phylogénie des *Silene* telle qu'elle est actuellement connue. L'algorithme de recherche dans les arbres binaires non ordonnés suppose un motif d'arbre binaire, c'est-à-dire résolu. Pour pouvoir utiliser l'arbre non résolu, il a fallu modifier TPMS pour qu'il accepte les multifurcations (c'est-à-dire les nœuds d'arbre ayant plus de deux fils), et qu'il les transforme en autant de versions d'arbres binaires possibles. L'algorithme 7 a été utilisé.

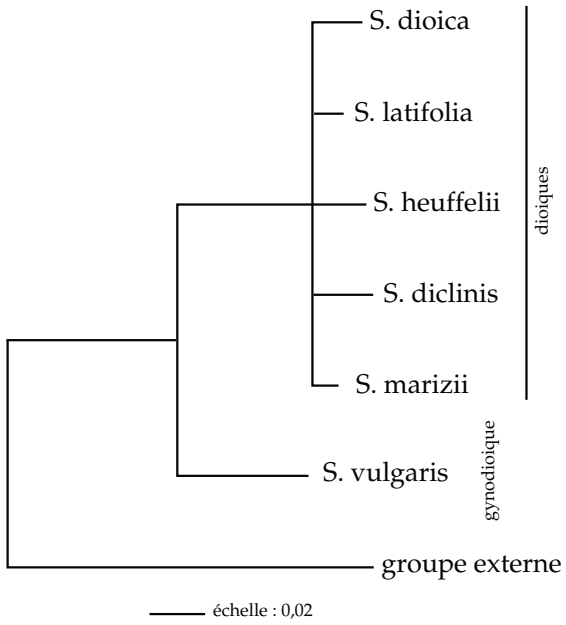


FIGURE 4.1 – **Phylogénie des *Silene***, adaptée de Käfer *et al.* [66].

Nous avons vu plus haut que le nombre de topologies possibles augmente très rapidement avec le nombre de nœuds, il convient donc de rester très prudent dans le nombre de nœuds non résolus dans les arbres requêtes. On a ensuite utilisé le formalisme de TPMS pour représenter cet arbre d'espèces :

```
(DIANTHUS SYLVESTRIS/DIANTHUS SYLVESTRIS, (SILENE VULGARIS/SILENE VULGARIS, (SILENE
MARIZII, (SILENE DIOICA, SILENE LATIFOLIA))/DIOICOUS)/SILENE));
```

La recherche a pris une fraction de seconde, et cela a permis de sélectionner 25

Algorithme 7 Transforme un nœud non binaire en différentes sous-arbres binaires.

```
fonction GENEREBINAIRE(nœudsRequête)
    si cardinal(nœudsRequête) > 2 alors
        pour tout éléments de nœudsRequête faire
            retourne couple de nœuds (élément courant, genereBinaire(le reste de
nœudsRequête))
        fin pour
    sinonretourne nœuds
    fin si
fin fonction
```

familles et de les utiliser comme orthologues putatifs.

4.1.2 Sans contrainte sur la topologie — Détermination de l'âge des chromosomes sexuels de l'algue *Ectocarpus*

Il arrive parfois qu'on ne connaisse pas exactement la topologie de l'arbre des espèces dont on souhaite trouver des gènes orthologues. Il est également possible de vouloir des orthologues de manière floue dans un grand nombre d'espèces, et TPMS ne permet pas de faire une recherche avec des espèces facultatives (il n'est pas possible de demander « Je veux que si telle espèce se trouve dans l'arbre elle soit placée ici, mais sa présence n'est pas obligatoire »). Une autre stratégie peut alors être utilisée : rechercher dans les arbres de gènes les plus grands sous-arbres possibles contenant au plus un exemplaire d'un gène de chaque espèce. Le fait de n'avoir qu'une seule copie par espèce ne donne pas la certitude d'être en présence d'orthologues, mais c'est une bonne présomption.

C'est ce qui a été utilisé par Ahmed et collaborateurs [2] dans une étude tentant de déterminer l'âge des deux chromosomes sexuels de l'algue brune *Ectocarpus*. Les chromosomes sexuels [19] sont issus de la divergence d'une paire d'autosomes [94]. Déterminer leur âge revient à déterminer ce temps de divergence. Pour ce faire, il est nécessaire de déterminer un profil de divergence des espèces apparentée à l'espèce d'intérêt. Il s'agit de mettre en lien deux valeurs : d'une part le temps de divergence entre *Ectocarpus* et une autre espèce, et d'autre part, le taux de substitutions synonymes d_S entre *Ectocarpus* et cette autre espèce. On utilise comme point de comparaison la diver-

gence entre espèces proches, car la divergence entre deux chromosomes sexuels peut être assimilée à une spéciation : un fois devenus chromosomes sexuels, ils ne recombinent plus entre eux[2]. Le temps de divergence entre *Ectocarpus* et d'autres espèces a déjà été établi par d'autres études, et il reste à calculer des d_S moyens représentatifs de chaque couple {*Ectocarpus*-une autre espèce}. Ainsi, ce profil pourra être utilisé, pour déterminer le temps de divergence à partir du d_S entre les deux chromosomes sexuels U et V de *Ectocarpus*.

Pour calculer ces d_S moyens, il faut travailler sur des gènes orthologues. Il est donc nécessaire d'extraire des jeux d'orthologues à partir de séquences connues d'algues brunes. Un des problèmes est que le génome des algues d'intérêt n'est pas séquencé (on ne dispose que de quelques gènes issus du transcriptome), donc on ne sait pas si les séquences d'algues dont on dispose sont bien des séquences appartenant à des autosomes (séquences ARNt, ARNr, chloroplastiques et mitochondriales exclues). Une solution a été de ne sélectionner que les orthologues à des gènes identifiés appartenant aux deux espèces d'un groupe externe (diatomées, *Bacillariophyta*), qui elles sont entièrement séquencées et annotées. Ces séquences ont été sélectionnées avec l'outil ACNUC [43] dans la banque de données HOGENOM [97].

L'étape suivante a consisté à former des clusters avec des séquences d'algues non publiques séquencées par le projet [2]. Les familles ont été formées avec BLAST [5], SiLiX [86], les alignements multiples réalisés avec MUSCLE [29], et les arbres avec PhyML [48]. Ceci a permis de générer 856 clusters. Les arbres ont ensuite été transformés en collection TPMS, racinés selon la méthode taxonomique, puis annoté avec le score d'unicité. Cela a permis de ne garder que 287 sous-arbres ne contenant qu'une séquence par espèce (score d'unicité de 1). Après une étape d'expertise manuelle consistant à vérifier la conformité des arbres sélectionnés avec l'arbre des espèces[3], 54 arbres ont été conservés.

Ensuite, on a calculé le d_s de chaque couple {*Ectocarpus*-une autre espèce} dans chaque cluster, ce qui permet d'établir le profil de divergence. L'étude [2] conclut

[2]Certaines parties du chromosome sexuel se comportent comme des autosomes, les régions pseudo-autosomales (RPA), tandis que le reste ne présente plus de recombinaison. Ce sont ces dernières régions qui intéressent ici l'étude et sont désignées par le terme *chromosome sexuel*.

[3]TPMS ne permet pas encore de vérifier si un arbre est conforme à un arbre des espèces en tolérant l'absence de certaines des espèces.

un âge beaucoup plus ancien[4] que ce que laisserait penser la taille des zones U- et V-specificiques (on s'attendait à un âge de 5 à 50 millions d'années).

4.2 Tester des hypothèses évolutives

Il existe de nombreux sujets de controverse en phylogénie. Bien que les données récemment acquises viennent toujours et encore augmenter la quantité de savoir sur l'histoire évolutive du vivant, il reste des points de phylogénie qui soulèvent la controverse, pour lesquelles les données ne sont pas très éloquantes ou ne donnent pas des résultats unanimes selon les expérimentateurs et les méthodes utilisées. TPMS peut permettre de compter le nombre d'arbres qui sont le plus congruents avec une hypothèse donnée. Je donnerai ici quelques exemples, ainsi que les résultats obtenus.

4.2.1 Cœlomates vs Ecdysozoa

Nous avons vu que la classification des êtres vivants peut être établie à partir de données morphologiques. Les étapes du développement embryonaire et les différents types de tissus engendrés pendant ces étapes sont également des marqueurs permettant de classer les espèces. Par exemple, chez les métazoaires, c'est-à-dire les eucaryotes pluricellulaires, mobiles et hétérotrophes, l'embryon voit ses cellules organisées en feuillets. Selon les espèces, ces feuillets peuvent être au nombre de deux, ectoderme et endoderme, ou trois, soit les deux précédents et le troisième, le mésoderme (qui engendre le péritoine et les organes qu'il contient). Ces derniers ont été regroupés dans le groupe taxonomique des *Coelomata*. Mais certains organismes non triblastiques, appelés *pseudocoelomates*, auraient en fait perdu le mésoderme que leur ancêtre avait précédemment acquis au cours de l'évolution.

La phylogénie moléculaire permet d'obtenir des éléments de réponse quant au fait de savoir si les coelomates constituent un groupe monophylétique[5], ou au contraire si tous les animaux à exosquelette qui muent (comme les arthropodes, qui sont des coelomates, et les nématodes classés dans les pseudocoelomates) forment un groupe

[4] Les auteurs de l'étude [2] ne souhaitent pas que l'âge précis soit divulgué avant la parution de l'article, mais l'âge inféré par cette méthode est significativement supérieur à l'âge attendu.

[5] Un groupe de séquences est dit monophylétique s'il existe, dans l'arbre des espèces, un sous-arbre qui contient toutes les espèces concernées et seulement celles-ci.

distinct, les *ecdysozoaires*. La question est alors de savoir laquelle des deux hypothèses présentée en figure 4.2 est vraie.

De nombreuses études ont été menées en ce sens. Alors que certaines, basées sur des phylogénies de gènes uniques, soutiennent l'hypothèse Ecdysozoa [81, 108, 80], d'autres, ayant une approche phylogénomique, ont d'abord favorisé l'hypothèse Coelomata [12, 131, 137], avant de faire pencher la balance du côté Ecdysozoa [25, 69, 53].

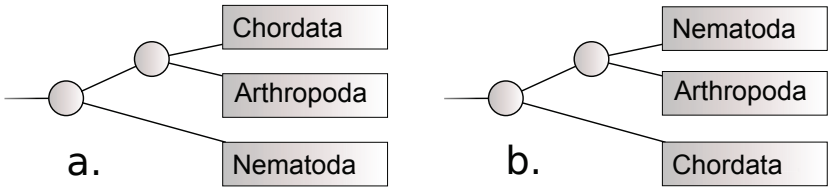


FIGURE 4.2 – **Hypothèses Coelomates *vs* Ecdysozoa.** a. Phylogénie attendue si le groupe *Coelomata* est monophylétique (*Chordata* + *Arthropoda*). b. Phylogénie attendue si les coelomates ne sont pas un groupe monophylétiques, mais que les *Ecdysozoa* (*Nematoda* + *Arthropoda*) le sont.

TPMS permet quant à lui de répondre à la question suivante : dans une collection de gènes donnée, combien d'arbres plaident pour une hypothèse, et combien plaident pour l'autre ? Pour ce faire [11], nous avons utilisé comme collection la base de données de familles homologues HOMOLENS 5 [97], qui regroupe les génomes de la banque Ensembl [54]. Nous avons utilisé les requêtes suivantes : `(//Fungi, (/Nematoda/Nematoda, (/Chordata/Chordata, /Arthropoda/Arthropoda)$80)$80)` et `(//Fungi, (/Chordata/Chordata, (/Nematoda/Nematoda, /Arthropoda/Arthropoda)$80)$80)` qui représentent respectivement les hypothèses Coelomata et Ecdysozoa, avec un bootstrap exigé supérieur ou égal à 80%. Plusieurs valeur de bootstrap minimum ont été essayées, et les résultats ont été consignés dans le tableau 4.1. Tous les résultats (exceptés les arbres obtenus sans minimum de bootstrap) montrent un plus grand nombre correspondant à l'hypothèse coelomates.

Cela ne permet pas pour autant de conclure que plus de gènes sont en faveur de l'hypothèse coelomates : il faudrait pour cela utiliser les alignements qui ont été à la base de ces arbres et tester les éventuelles différences de vitesses d'évolution qui auraient pu conduire à un phénomène d'attraction des longues branches[6], d'autant

[6]L'attraction des longues branches est le phénomène qui regroupe artefactuellement les organismes présentant un fort taux d'évolution, et donc de longues branches, à la base de l'arbre.

plus que le seul nématode disponible dans HOMOLENS 5 est *Caenorhabditis elegans*, connu pour avoir un fort taux d'évolution, et donc présenter un tel artefact [24]. Cette démarche ne constitue donc pas une démonstration, mais donne un exemple de la simplicité avec laquelle il est possible d'interroger une large collection de séquences pour en extraire une tendance.

Seuil bootstrap	aucun	$\geq 80\%$	$\geq 85\%$	$\geq 90\%$	$\geq 95\%$
Coelomata	768	241	174	110	54
Ecdysozoa	1309	149	83	45	20

TABLEAU 4.1 – Nombre d'arbres de HOGENOM correspondant respectivement à la phylogénie de l'hypothèse Coelomata et Ecdysozoa, selon le soutien statistique aux branches en bootstrap.

4.2.2 *Chiroptera* dans *Pegasoferæ*, phylogénie des mammifères

Nous nous intéressons ici à la phylogénie des mammifères. Dans l'article [11], nous sommes partis de l'arbre des espèces établi par Dos Reis et collaborateurs [26] qui est reproduit sur la figure 4.3. La démarche a été de décomposer cet arbre des espèces et d'interroger HOMOLENS [97] pour voir le soutien qu'il apportait à chaque sous-arbre. L'arbre de Dos Reis a été légèrement modifié pour correspondre aux espèces présentes dans HOMOLENS. Ainsi, l'orang-outan *Pongo abelii* a été remplacé par *Pongo pygmaeus*, l'alpaga *Vicugna pacos* par *Lama pacos*, et le chien *Canis familiaris* par *Canis lupus familiaris* ainsi que le panda géant *Ailuropoda melanoleuca*, un autre caniforme.

Tester un sous-arbre de l'arbre des espèces revient ici à compter tous les arbres de gènes qui contiennent un jeu d'orthologues congruent avec ce sous-arbre. Par exemple, pour les rongeurs, le motif

```
((((!//Rattus Norvegicus,!//Mus Musculus)!,!//Dipodomys ordii)!,!//Cavia Porcellus)!,
!//Spermophilus Tridecemlineatus)
```

sélectionne toutes les familles de gènes contenant exactement la phylogénie des rongeurs. On a ici utilisé le point d'exclamation qui permet d'exiger un lien direct entre le nœud parent et son fils ainsi annoté. Une liste de requêtes de ce genre a été générée et utilisée pour interroger la collection.

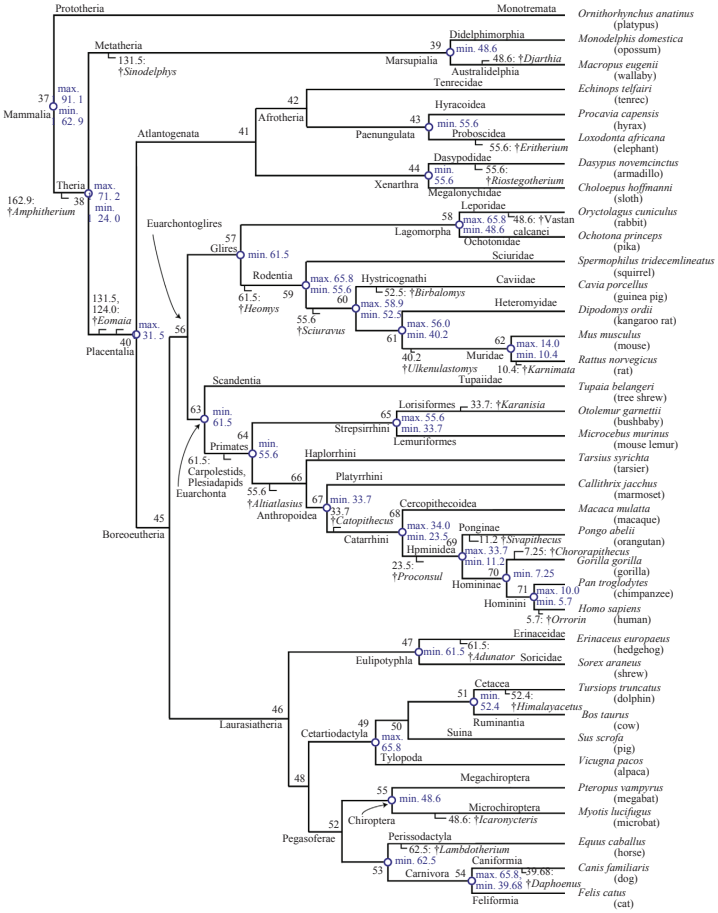


FIGURE 4.3 – **Phylogénie des mammifères** selon Dos Reis *et al.* [26].

Les résultats ont été consignés sur la figure 4.4 : chaque sous-arbre de l'arbre des espèces de Dos Reis est associé au ratio du nombre d'arbres de HOMOLENS contenant ce sous-arbre des espèces sur le nombre d'arbres éligibles, c'est-à-dire contenant toutes les espèces impliquées, quelle que soit leur topologie. Comme on peut s'y attendre, ces deux nombres décroissent quand on va des feuilles vers la racine : plus on s'approche de cette dernière, plus le nombre d'espèces impliquées est important, et moins le nombre de familles de gènes contenant toutes ces espèces est grand. Cela est particulièrement caractéristique pour les groupes *Boreoeutheria*, *Eutheria* et *Theria*, dont aucune famille ne soutient la topologie. Cela est dû au fait que nous cherchons ici une topologie très stricte. Si TPMS pouvait accepter les motifs flous, l'absence de certaines espèces ne serait pas constitutif d'une incongruence, et autoriserait plus de motifs. Mais comme il

ne le peut pas, il exige des motifs stricts impliquant de nombreuses espèces, ce qui est très difficile à trouver.

Comme pour la controverse Coelomata/Ecdysozoa, nous pouvons, de plusieurs hypothèses évolutives, vouloir savoir laquelle est la plus soutenue par les arbres de la collection. Ici nous nous sommes intéressés à la place des chiroptères (mammifères volants) dans le groupe des pégasofères qui contient également les canidés, félidés, et ursidés. Waddel *et al.* [128] ont fait l'hypothèse que, contrairement à ce qui est rapporté par Dos Reis, le groupe *Chiroptera* est en fait frère de *Equus caballus*. Pour savoir combien d'arbres, dans notre collection, sont conformes à cette phylogénie, nous l'interrogeons avec la requête `((!//Equus caballus, (!//Pteropus vampyrus,!//Myotis lucifugus)!)!, ((!//Canis lupus familiaris,!//Ailuropoda melanoleuca)!,!//Felis catus)!);`. Seuls 6/6466 arbres contiennent ce motif, alors que la phylogénie avancée par Dos Reis concerne 168/6466 arbres. La phylogénie de Waddell est donc significativement moins soutenue.

4.2.3 Place du *Rhizobium* sp. NT-26

Andres et collaborateurs ont étudié une souche du genre *Rhizobium* [6], une bactérie capable d'oxyder l'arsenic et donc de vivre dans un environnement assez défavorable : une mine d'or contenant de l'arsenic. Cette bactérie a été séquencée et son génome étudié afin de mieux comprendre les mécanismes évolutifs impliqués dans son mode de vie. Elle présent notamment la particularité de contenir un méga-plasmide codant pour les gènes nécessaires à l'oxydation de l'arsenic.

Une des questions de l'article est de savoir où se place cette souche par rapport aux *Agrobacterium* : soit à l'intérieur du clade, en tant que groupe frère de *A. tumefaciens*, soit en groupe externe immédiat, c'est-à-dire en groupe frère du clade. Les deux motifs TPMS correspondants sont respectivement :

```
(//ROOT-N11-N15,(/RNT26/RNT26,(//N11,//N15)$90/N11+N15));
```

et

```
(//ROOT-N11-N15,(/N11/N11,(/RNT26/RNT26,/N15/N15)$90/RNT26+N15));
```

Avec dans l'arbre d'espèces RNT26 la bactérie d'intérêt, N15 *A. tumefaciens*, et N11 les *Agrobacterium* (N11 contient N15). Le nœud `//ROOT-N11-N15` est un nœud

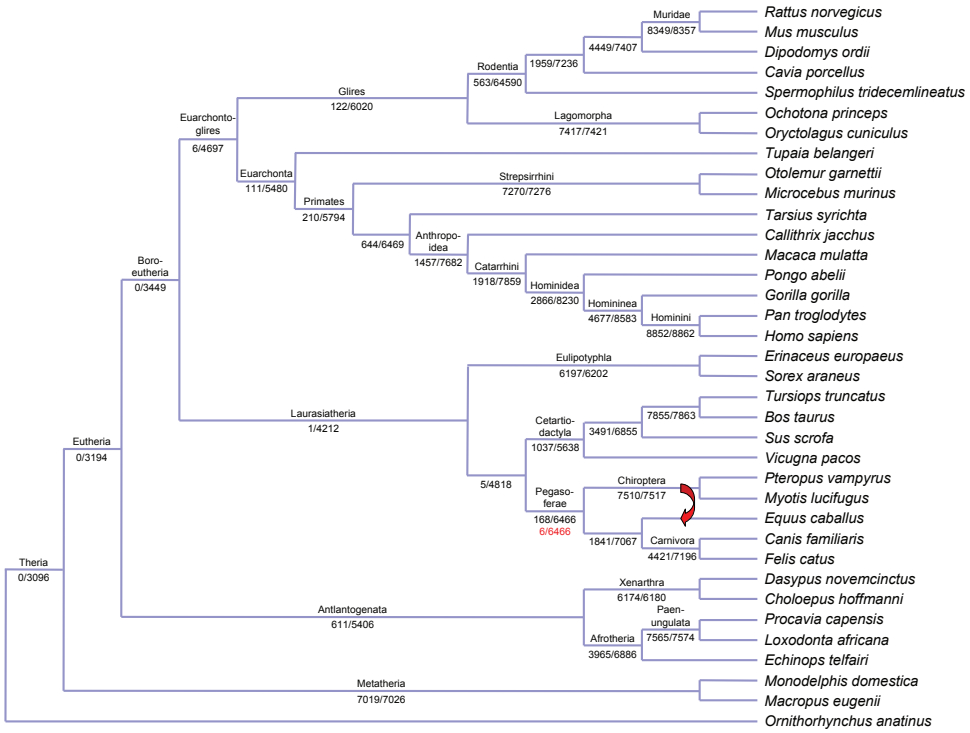


FIGURE 4.4 – **Soutien de la phylogénie des mammifères** selon Dos Reis *et al.* [26] par les arbres de la collection HOMOLENS [97]. Sur chaque branche, on voit le soutien à l'arbre qu'elle porte. En numérateur, le nombre de familles de la collection correspondant à la topologie des espèces, et en dénominateur, le nombre de familles qui contiennent au moins une séquence de chaque espèce impliquée dans la topologie. La flèche rouge représente l'hypothèse évolutive selon laquelle les *Chiroptera* sont un groupe frère de *Equus caballus*.

externe qui contient des séquences de l'arbre qui ne soient ni des *Agrobacterium*, ni des *A. tumefaciens*. Dans ces motifs, on exige 90% de bootstrap aux nœuds internes.

Les 2 878 familles de gènes contenant 3 537 gènes de la souche *Rhizobium* sp. NT-26 ont été exploités à l'aide de TPMS. Il en ressort que 338 familles (268 en utilisant la restriction de bootstrap à 90%) correspondent au premier motif, c'est-à-dire RNT26 à l'intérieur du clade contre 255 (146 idem) pour la seconde hypothèse. Il y a donc significativement plus d'arbres qui placent RNT26 au sein des *Agrobacterium* selon un test de χ^2 (p-value $< 10^{-3}$).

À l'issue de cette première expérience, il y a donc deux groupes de gènes correspondant chacun à l'une ou l'autre des hypothèses. Mais les auteurs ne trouvent aucun

lien entre l'appartenance à un de ces groupes et la localisation du gène impliqué sur le génome. Cela les laisse à penser que ces différences de phylogénie sont dues à un manque de données ou bien à des transferts proches.

4.3 Détecter des incongruences dans des grandes collections

Dans cette section, nous essayons de quantifier le nombre d'incongruences présentes dans une grande collection de familles de gènes. La collection choisie est HOGENOM [97]. La détection systématique d'incongruences a été utilisée sur les 128 674 arbres de gènes disponibles dans HOGENOM, après qu'ils ont été racinés par l'algorithme combiné de racinement [11].

4.3.1 Comptage des incongruences soutenues

L'algorithme détecte 110 359 incongruences soutenues par des valeurs de bootstrap $\geq$ 90%. Cela correspond à une moyenne de 0,86 erreur par arbre. Différents évènements peuvent expliquer ces incongruences :

- Des erreurs de reconstruction phylogénétique : par exemple le phénomène d'attraction des longues branches qui groupe les longues branches ensemble et à la base de l'arbre.
- Des erreurs dans l'arbre de référence : ici c'est l'arbre du NCBI qui est utilisé dans HOGENOM. Cet arbre a deux défauts du point de vue de la phylogénie : il est non résolu, c'est-à-dire qu'il présente des multifurcations à de nombreux nœuds internes, et il n'est pas forcément congruent avec tous les arbres d'espèces reconstitués. Ce genre de contradiction peut alors amener à des incongruences apparentes entre une partie de l'histoire d'un gène et l'arbre de référence, alors qu'en fait c'est l'arbre de référence qui est incongruent avec l'histoire évolutive du vivant.
- Un mauvais enracinement de TPMS : dans certains cas, TPMS peut être induit en erreur et mal raciner les arbres. Alors que normalement, deux parties de-

vraient être séparées par la racine, à la suite d'un mauvais enracinement, l'une se retrouve nichée dans l'autre, et semble être une incongruence.

- Des paralogies cachées : les arbres de gènes de HOGENOM ne sont pas réconciliés, et donc on n'a pas inféré de duplications.
- Un tri de lignée incomplet.
- Un transfert horizontal.

La figure 4.5 présente les comptages des incongruences intra et inter-domaines. On entend par *incongruence intra-domaine* un mauvais placement de la séquence d'une espèce à l'intérieur de son domaine mais hors du groupe taxonomique auxquel elle appartient, et par *incongruence inter-domaine* la séquence d'une espèce placée dans un autre domaine que celui auquel elle est censée appartenir.

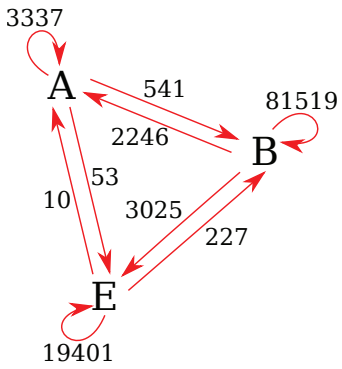


FIGURE 4.5 – **Nombre d'incongruences mesurées entre et à l'intérieur des trois domaines du vivant.** Les trois domaines (Archées, Bactéries et Eucaryotes) sont représentés respectivement par les lettres A, B et E. Concernant les incongruences inter-domaines, les flèches donnent la direction du changement observé.

4.3.2 Endosymbiose : mitochondries et chloroplastes

À ce stade, on peut se demander si la méthode de détection d'incongruences est un bon moyen de détecter des transferts horizontaux. Pour cela, on utilise des transferts connus et certains : l'intégration des chloroplastes et des mitochondries dans les noyaux de cellules eucaryotes. Pour ce faire, l'algorithme de détection a été lancé

dans les familles de gènes étiquetées comme relatives aux mitochondries dans HOGENOM5, au nombre de 374. Parmi ces gènes, 252 (soit 67.4%) ont été identifiés comme impliqués dans une incongruence les plaçant au milieu de bactéries. De même, sur les 377 gènes de chloroplastes, 249 (soit 66%) sont détectés comme étant impliqués dans une incongruence les plaçant au milieu de bactéries.

5 Discussion et perspectives

5.1 Perspectives d'amélioration

5.1.1 Détection de transferts

TPMS contient un algorithme de détection d'incongruences. Souvent, dans les arbres contenant des gènes d'espèces unicellulaires, ces incongruences peuvent être la conséquence de transferts horizontaux. Actuellement, l'argorithme est déjà capable de trouver des motifs d'incongruence dans lequel un sous-arbre est responsable d'une perturbation taxonomique selon l'arbre des espèces. Pour détecter les transferts, un arbre correctement raciné est nécessaire, car les perturbations taxonomiques sont déduites des affectations taxonomiques. Le problème est que la procédure actuelle d'enracinement ne garantit pas la racine optimale dans un arbre contenant beaucoup de transferts. En effet, dans ce cas, l'algorithme peut aboutir à un enracinement erroné, ce qui cause des affectations taxonomique partiellement fausses et donc des faux-positifs.

Par contre, l'utilisation de la recherche de motifs explicites de transferts (un taxon situé dans un autre groupe taxonomique alors que cela ne devrait pas être le cas dans l'arbre des espèces), en utilisant `tpms_query`, est très efficace. Un outil comme Prunier [1] qui est dédié à cette tâche est évidemment plus fiable que TPMS, mais il a comme limitation de n'accepter que les arbres de gènes unicopie, c'est-à-dire les arbres ne contenant qu'un seul gène pour chaque espèce, ce qui limite son utilisation à une fraction des familles disponibles.

Lorsque le problème d'enracinement précédemment évoqué sera résolu, l'utilisation de l'algorithme de détection d'incongruences de TPMS permettra alors de faire une recherche systématique et complète des transferts horizontaux contenus dans une grande collections d'arbres de gènes comme HOGENOM [97]. Cela pourrait donner lieu à l'établissement de statistiques sur les différents flux de gènes, permettant alors de différencier l'importance du nombre de transferts selon que les gènes concernés soient informationnels (c'est-à-dire les gènes impliqués dans les processus du maintien de l'intégrité de l'information génétique) ou opérationnels : en effet, il a été montré que les gènes informationnels sont moins transférés que les gènes opérationnels [57]. Comme chez les procaryotes la majorité des gènes multicopies sont opérationnels, ils pourraient enfin être pris en compte à large échelle dans l'évaluation de leur part dans les tranferts horizontaux.

5.1.2 Exemple d'une nouvelle fonctionnalité

L'exploration de nombreuses familles de gènes peut conduire a beaucoup d'applications. Par exemple, il est facile d'imaginer déterminer un arbre d'espèces consensus à partir d'un grand nombre d'arbres de gènes. Si, dans la collection, on détermine pour chaque arbre, la liste des bipartitions existantes, cela peut permettre d'établir une matrice mettant en regard toutes les espèces présentes dans ces arbres. Pour chaque couple d'espèces, on détermine le nombre de bipartitions (ramené au nombre d'arbres dans lesquelles ces bipartitions sont possibles) les séparant. On peut ensuite utiliser une méthode de distance pour reconstituer l'arbre des espèces correspondant.

5.2 Mise à disposition de la communauté

La publication de TPMS a suscité un certain intérêt, puisque la revue dans laquelle l'article a été publié l'a classée comme *highly accessed* à titre définitif[1], ce qui signifie qu'il est significativement plus consulté que les articles parus à la même période dans la même revue. Ceci démontre un intérêt pour ce genre de thématique, et signifie qu'il faut faire d'autant plus d'efforts sur la diffusion du programme. Je propose ici

[1] Voir `http://www.biomedcentral.com/about/mostviewed/`.

une réflexion sur les différentes formes que peuvent prendre un programme de calcul scientifique.

5.2.1 Historique des différentes formes de programmes

La forme des programmes informatiques a beaucoup évolué avec le temps. Le fait d'exécuter des programmes localement sur la propre machine de l'utilisateur, ou au contraire le fait de se servir d'un ordinateur comme d'un simple terminal de consultation est une question dont la réponse a pu, selon le contexte technique et idéologique, beaucoup varier.

Aux débuts de l'informatique, les ordinateurs (appelés *mainframes*) étaient volumineux, consommateurs de ressources, très chers. Ils étaient réservés à l'industrie, la banque, la recherche scientifique. Leur fonctionnement se faisait en mode *batch* : comme une tâche était exécutée seulement lorsque la précédente avait pris fin, les résultats n'étaient pas immédiats. Il n'était possible d'accéder à de telles machines que *via* un terminal, c'est-à-dire un ensemble clavier-écran, connecté à l'ordinateur par le réseau. Ce terminal n'était doté d'aucune intelligence, et sa seule tâche était de transmettre correctement ce qui était tapé au clavier, et d'afficher correctement les séquences de caractères reçues. L'utilisation de ces *mainframes* a donné naissance à un premier type d'application : le programme de calcul non interactif. Des données sont entrées, le programme est exécuté, puis des données sont retournées.

Ensuite, les années 1980 ont vu l'avènement de la micro-informatique. Par *micro*, il faut entendre que la machine tenait sur un bureau. Assez vite, les progrès de miniaturisation et d'augmentation de la puissance de calcul ont permis de proposer aux utilisateurs des interfaces graphiques, et la souris. Deux types de programmes sont nés de cette ère : les programmes graphiques utilisables à la souris, et des programmes en ligne de commande interactifs.

Depuis les années 2000, le réseau s'est généralisé. La puissance des processeurs étant plafonnée[2], on a construit des machines avec de nombreux CPU, qu'il convient de mutualiser pour mieux les rentabiliser, et auxquelles on accède par le réseau. Ce genre de machine peut se situer dans le laboratoire de travail, ou de plus en plus fréquemment, distantes géographiquement, et accessibles *via* l'internet. Ces programmes, souvent en

[2]Voir le chapitre sur le parallélisme, 3.1.4.1.

ligne de commande, sont optimisés pour un fonctionnement en parallèle.

Vient enfin une troisième génération de programmes : les applications sur le web. Elles existent depuis la popularisation du web, au début des années 1990, faute de puissance côté client, c'est-à-dire côté utilisateur. Ces applications sont pendant longtemps restées des formulaires qui permettaient d'exécuter des requêtes dans des bases de données [99], ou encore des interfaces sommaires pour des outils en ligne de commande [16, 118]. Puis, dans les années 2010, un effort particulier a été porté par les concepteurs de navigateurs web pour améliorer la puissance disponible dans ces derniers, notemment en ce qui concerne le JavaScript, langage de choix pour écrire des interfaces, côté client. La fondation Mozilla a sorti en 2013 son compilateur de JavaScript, IonMonkey[3], tandis que Google fait des efforts pour améliorer sa propre machine virtuelle JavaScript, nommée V8[4]. Cet effort de rapidité a rendu possible l'émergence d'applications complètes sur le web, qui se manient avec autant de confort qu'un programme exécuté localement.

5.2.2 Critères pour chaque type de programme

Les outils de bioinformatique peuvent donc prendre trois formes : calcul en ligne de commande, programme à interface graphique exécuté localement, ou interface sur le web. La table 5.2.2 résume les avantages et inconvénients de ces trois types de programmes. Quelques détails sur les critères évalués :

Facilité d'utilisation : Un programme en ligne de commande n'est pas forcément abordable. On ne peut connaître les options et la façon d'entrer les données qu'en lisant une documentation. Alors que les programmes graphiques[5] offrent la possibilité de découvrir le fonctionnement du programme d'après son aspect à l'écran.

Contraintes d'installation : Des compétences d'informaticiens sont parfois requises pour installer des programmes sur une machine locale. Les programmes de bioinformatique sont d'autant plus concernés par cette contrainte, car ils sont peu répandus, leur usage étant circonscrit à une petite communauté de scientifiques,

[3] `https://blog.mozilla.org/javascript/2012/09/12/ionmonkey-in-firefox-18/`
[4] `https://developers.google.com/v8/`
[5] du moins ceux qui ont une bonne conception ergonomique

TABLEAU 5.1 – Avantages et inconvénients de différents types d'applications

	Calcul en ligne de commande	Interface graphique locale	Application web
Facilité d'utilisation	Faible	**Bonne**	**Bonne**
Contraintes d'installation	Forte	Forte	**Nulle**
Dépendance à une connexion à l'internet	**Non**	**Non**	Oui
Dépendance à l'hébergeur	**Non**	**Non**	Forte
Possibilité de scripter et combiner les outils	**Oui**	Non	Généralement non
Étude/adaptation possibles du code	**Oui, souvent**	**Oui, souvent**	Non, souvent

et sont parfois peu empaquetés[6] : parfois, seul le code source est laissé à la disposition de la communauté. Une interface web, quant à elle, permet l'accès à tous, du moment que l'administrateur du serveur qui souhaite offrir un service décide de supporter cette charge.

Dépendance à une connexion à l'internet : Même si les situations où on n'a pas accès au réseau sont de plus en plus rares, elles existent, ou l'accès est parfois de piètre qualité. Un programme local est indépendant des contraintes de réseau.

Dépendance à l'hébergeur : Un program local[7] peut subsister même si son/ses auteurs ne le maintiennent plus. Un logiciel qui ne serait que sur le web, et dont l'hébergeur unique décide de ne plus le proposer devient de fait totalement inaccessible.

Étude/adaptation possibles du code : Corollaire du point précédent : les logiciels dont les binaires sont distribués sont souvent distribués avec leur code source, par nécessité, ou par choix philosophique. Cela permet la correction de bugs,

[6] on pense ici à des paquets de distributions linux (`.rpm`, `.deb`), des paquets MacPorts, ou encore des images disque `dmg` pour Mac OS qui sont tous des moyens de fournir une application fonctionnelle.

[7] *a fortiori* s'il est libre ou *open source*

une étude du fonctionnement, ainsi qu'une possible adaptation à des besoins particuliers.

5.2.3 Distribution de TPMS

TPMS est pour l'instant un programme en ligne de commande. Cela le rend facile à intégrer dans un processus de traitement de données. Une des orientations qu'il faudra avoir est de permettre le traitement de fichier d'entrée : pour l'intant, il ne fonctionne qu'en mode interactif, c'est-à-dire qu'il faut répondre à des questions posées par le programme (nom de la requête, opération à effectuer)[8].

Un axe majeur de travail pour une meilleur diffusion de ce code serait l'intégration dans le projet Bio++. Cette intégration aurait plusieurs avantages :

Limiter la fragmentation du code : De nombreuses fonctions de TPMS pourraient concerner bien d'autres programmes que lui-même, comme la gestion des groupes taxonomiques et des espèces, la recherche de motifs.

Limiter la maintenance : TPMS est lié à une version de Bio++. Chaque mise à jour de ce dernier risque de le rendre incompatible. Le fait d'inclure TPMS dans le projet permettrait de synchroniser les mises à jour, même si cela occasione à chaque fois la même charge de travail.

Profiter de la notoriété du projet : Même si les publications scientifiques sont de bonnes sources pour apprendre l'existence d'un programme, la mention de celui-ci dans la documentation d'un projet peut également être un bon moyen de le faire connaître.

Simplifier l'installation à l'utilisateur : Pour un utilisateur, ou même un administrateur système, installer des bibliothèques et des suites de programmes est toujours une tâche source de complications : dépendances à d'autres bibliothèques, prise en compte des bons chemins de compilations, synchronisation des mises à jour sont autant de sources de blocages. N'avoir qu'un seul projet ou suite à installer simplifie énormément les choses.

[8] Cela dit, il est possible d'envoyer une séquence de commandes sur l'entrée standard, même pour les programmes en mode interactif. Mais cette façon de faire rend le procédé peu clair, et les erreurs faciles.

Cependant, l'intégration à un autre projet quel qu'il soit demanderait un grand travail de réécriture pour se conformer aux conventions du projet de destination, et au fait qu'une partie du programme serait déplacé dans des bibliothèques logicielles indépendantes.

5.3 Travail de sécurisation du code

Le programme a été écrit dans une optique d'efficacité et d'optimisation. Néanmoins, depuis le début de son écriture il y a plusieurs années, mon exigeance et mon niveau de maîtrise du C++ et des techniques de programmation ont évolué. Le programme n'est donc pas toujours homogène, et n'a pas satisfait à une étude poussée du code. Plusieurs axes de travail sont possibles :

Ajouter des contraintes sur la modification possible des variables Il faudrait que seules les fonctions qui ont besoin d'agir sur les variables qui leur sont adressées aient le droit de le faire. Le langage C++ permet de le faire avec l'opérateur `const`. Cette rigueur est contraignante pour le programmeur, mais elle rend le fonctionnement du programme beaucoup plus prévisible.

Améliorer le chargement des arbres Pour l'instant, le chargement d'un arbre ou d'une collection non conforme entraîne l'arrêt du programme sans trop de précisions. Il faudrait dans l'idéal que le programme soit i) plus tolérant quant à la forme des données d'entrées, et ii) qu'il indique précisément le problème le cas échéant.

Détection des fuites de mémoire Au fil du chargement des arbres, de la recherche de motifs, de nombreux objets sont créés de façon dynamique, c'est-à-dire que leur existence ne dépend pas d'un contexte particulier du programme : ils existent tant qu'ils ne sont pas supprimés. Si, lors d'une procédure répétitive (succession de recherche de motifs, de parcours de familles), une fonction génère un nombre important d'objet de manière dynamique, et qu'un oubli dans la procédure fait qu'ils ne sont pas supprimés, ces objets vont ainsi s'accumuler, faisant croître l'empreinte mémoire du programme. Autrement dit, il va occuper une place de plus en plus importante dans la mémoire du système tant qu'il n'est pas terminé. Cela peut aboutir à des situations critiques où une mémoire trop importante du système est utilisée.

5.4 Performances

TPMS présente de très bonnes performances. La raison principale est qu'un effort particulier a été apporté à l'optimisation des fonctions utilisées. Une seconde raison non négligeable est que le C++ est un langage compilé : les instructions du binaire obtenu sont directement exécutées par le processeur sans étape intermédiaire. De plus, les instructions sont adaptées au processeur cible, et exploite toutes les optimisations disponibles sur celui-ci. D'autre part, le code est parallélisé, ce qui permet d'utiliser tous les cœurs d'une machine, et de diminuer quasi linéairement le temps de calcul.

À titre d'exemple, sur une machine de bureau de type Xeon 2,67 GHz avec quatre cœurs, les deux recherches de motifs pour répondre à la controverse Coelomata / Ecdysozoa (voir section 4.2.1) ont pris seulement six secondes. La collection utilisée contenait pourtant 14 190 familles de gènes. D'autre part, l'enracinement complet d'une banque de 234 892 familles (avec méthode combinée, voir 3.4) faites de 3 666 568 séquences prend approximativement 30 secondes en utilisant 16 cœurs d'une machine de calcul.

5.5 Conclusion

La suite développée est donc un outil très performant qui a de nombreuses applications dans l'extraction d'informations dans les arbres phylogénétiques. Il a fait l'objet d'une publication scientifique [11] déjà classée *highly accessed*, et de plusieurs utilisations ayant donné lieu à une publication scientifique [2, 6, 66]. De nombreuses idées d'exploration des familles de gènes restent encore à explorer, dans un domaine où aucun programme de ce type n'existe.

Bibliographie

[1] Sophie S. ABBY, Éric TANNIER, Manolo GOUY et Vincent DAUBIN : Detecting lateral gene transfers by statistical reconciliation of phylogenetic forests. *BMC Bioinformatics*, 11:324, 2010.

[2] Sophia AHMED, J Mark COCK, Eugénie PESSIA, Rémy LUTHRINGER, Alexandre CORMIER, Marine ROBUCHON, Lieven STERCK, Akira F PETERS, Simon DITTAMI, Erwan CORRE, Myriam VALERO, Jean-Marc AURY, Denis ROZE, Yves Van de PEER, John H BOTHWELL, Gabriel AB MARAIS et Susana COELHO : The evolution of an ancient pair of uv sex chromosomes. *Science*, 2013, Soumis.

[3] John ALDRICH : RA Fisher and the making of maximum likelihood 1912-1922. *Statistical Science*, 12:162–176, 1997.

[4] Adrian M ALTENHOFF, Adrian SCHNEIDER, Gaston H GONNET et Christophe DESSIMOZ : OMA 2011 : orthology inference among 1000 complete genomes. *Nucleic Acids Research*, 39:D289–D294, 2011.

[5] Stephen F ALTSCHUL, Warren GISH, Webb MILLER, Eugene W MYERS et David J LIPMAN : Basic local alignment search tool. *Journal of molecular biology*, 215:403–410, 1990.

[6] Jeremy ANDRES, Florence ARSÈNE-PLOETZE, Valérie BARBE, Céline BROCHIER-ARMANET, Jessica CLEISS-ARNOLD, Jean-Yves COPPÉE, Marie-Agnès DILLIES, Lucie GEIST, Aurélie JOUBLIN, Sandrine KOECHLER, Florent LASSALLE, Marie MARCHAL, Claudine MÉDIGUE, Daniel MULLER, Xavier NESME, Frédéric PLEWNIAK, Caroline PROUX, Martha Helena. RAMIREZ-BAHENA, Chantal SCHENOWITZ, Sismeiro ODILE, David VALLENET, Joanne M. SANTINI et Philippe BERTIN : Life in an arsenic-containing gold mine : genome and physiology of the autotrophic arsenite-oxidizing bacterium Rhizobium sp. NT-26. *Genome Biology and Evolution*, Accepté, 2013.

[7] Maria ANISIMOVA et Olivier GASCUEL : Approximate likelihood-ratio test for branches : A fast, accurate, and powerful alternative. *Systematic Biology*, 55:539–552, 2006.

[8] Thomas BAYES et Richard PRICE : An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society*, 53:370–418, 1763.

[9] Dennis A BENSON, Ilene KARSCH-MIZRACHI, Karen CLARK, David J LIPMAN, James OSTELL et Eric W SAYERS : Genbank. *Nucleic Acids Research*, 40:D48–D53, 2012.

[10] Roman BIEK, Alexei J. DRUMMOND et Mary POSS : A virus reveals population structure and recent demographic history of its carnivore host. *Science*, 311:538–541, 2006.

[11] Thomas BIGOT, Vincent DAUBIN, Florent LASSALLE et Guy PERRIÈRE : TPMS : a set of utilities for querying collections of gene trees. *BMC Bioinformatics*, 14:109, 2013.

[12] Jaime E BLAIR, Kazuho IKEO, Takashi GOJOBORI et S Blair HEDGES : The evolutionary position of nematodes. *BMC Evolutionary Biology*, 2:7, 2002.

[13] Samuel BLANQUART et Nicolas LARTILLOT : A bayesian compound stochastic process for modeling nonstationary and nonhomogeneous sequence evolution. *Molecular Biology and Evolution*, 23:2058–2071, 2006.

[14] Nicolas BLAVET, Delphine CHARIF, Christine OGER-DESFEUX, Gabriel AB MARAIS et Alex WIDMER : Comparative high-throughput transcriptome sequencing and development of SiESTa, the Silene EST annotation database. *BMC Genomics*, 12:376, 2011.

[15] Bastien BOUSSAU et Manolo GOUY : Efficient likelihood computations with non-reversible models of evolution. *Systematic Biology*, 55:756–768, 2006.

[16] Christopher B BURGE et Samuel KARLIN : Finding the genes in genomic DNA. *Current Opinion in Structural Biology*, 8:346–354, 1998.

[17] Christiam CAMACHO, George COULOURIS, Vahram AVAGYAN, Ning MA, Jason PAPADOPOULOS, Kevin BEALER et Thomas L. MADDEN : BLAST+ : architecture and applications. *BMC Bioinformatics*, 10:421, 2009.

[18] Luca L CAVALLI-SFORZA et Anthony WF EDWARDS : Phylogenetic analysis. models and estimation procedures. *American Journal of Human Genetics*, 19: 233, 1967.

[19] Brian CHARLESWORTH *et al.* : The evolution of sex chromosomes. *Science*, 251:1030, 1991.

[20] Peter JA COCK, Tiago ANTAO, Jeffrey T CHANG, Brad A CHAPMAN, Cymon J COX, Andrew DALKE, Iddo FRIEDBERG, Thomas HAMELRYCK, Frank KAUFF, Bartek WILCZYNSKI *et al.* : Biopython : freely available python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25:1422–1423, 2009.

[21] IUPAC-IUB COMMISSION : Iupac-iub commission on biochemical nomenclature a one-letter notation for amino acid secruences tentative rules. *The Journal of Biological Chemistry*, 243:3557–3559, 1968.

[22] Charles DARWIN : *On the origins of species by means of natural selection*. Murray, London, 1859.

[23] Margaret O DAYHOFF et Robert M SCHWARTZ : A model of evolutionary change in proteins. *In* Margaret O DAYHOFF, éditeur : *Atlas of protein se-*

quence and structure, pages 345–352, Silver Springs, 1978. National Biomedical Researche Foundation.

[24] Dee R DENVER, Krystalynne MORRIS, Michael LYNCH et W Kelley THOMAS : High mutation rate and predominance of insertions in the *Caenorhabditis elegans* nuclear genome. *Nature*, 430:679–682, 2004.

[25] Hernán DOPAZO et Joaquín DOPAZO : Genome-scale evidence of the nematode-arthropod clade. *Genome Biology*, 6:R41, 2005.

[26] Mario dos REIS, Jun INOUE, Masami HASEGAWA, Robert J ASHER, Philip CJ DONOGHUE et Ziheng YANG : Phylogenomic datasets provide both precision and accuracy in estimating the timescale of placental mammal phylogeny. *Proceedings of the Royal Society B : Biological Sciences*, 279:3491–3500, 2012.

[27] Jean-François DUFAYARD, Laurent DURET, Simon PENEL, Manolo GOUY, François RECHENMANN et Guy PERRIÈRE : Tree pattern matching in phylogenetic trees : automatic search for orthologs or paralogs in homologous gene sequence databases. *Bioinformatics*, 21:2596–2603, 2005.

[28] Julien DUTHEIL, Sylvain GAILLARD, Eric BAZIN, Sylvain GLÉMIN, Vincent RANWEZ, Nicolas GALTIER et Khalid BELKHIR : Bio++ : a set of C++ libraries for sequence analysis, phylogenetics, molecular evolution and population genetics. *BMC Bioinformatics*, 7:188, 2006.

[29] Robert C. EDGAR : MUSCLE : a multiple sequence alignment method with reduced time and space complexity. *BMC Bioinformatics*, 5:113, 2004.

[30] Robert C. EDGAR : MUSCLE : multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research*, 32:1792–1797, 2004.

[31] Bradley EFRON : Bootstrap methods : another look at the jackknife. *The Annals of Statistics*, 7:1–26, 1979.

[32] Anton J ENRIGHT, Stijn VAN DONGEN et Christos A OUZOUNIS : An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Research*, 30:1575–1584, 2002.

[33] Oliver EULENSTEIN, Boris MIRKIN et Martin VINGRON : Duplication-based measures of difference between gene and species trees. *Journal of Computational Biology*, 5:135–148, 1998.

[34] Joseph FELSENSTEIN : Evolutionary trees from DNA sequences : a maximum likelihood approach. *Journal of Molecular Evolution*, 17:368–376, 1981.

[35] Joseph FELSENSTEIN : Confidence limits on phylogenies : an approach using the bootstrap. *Evolution*, 39(4):783–791, 1985.

[36] Joseph FELSENSTEIN : PHYLIP - Phylogeny Inference Package (Version 3.2). *Cladistics*, 5:164–166, 1989.

[37] Walter M. FITCH : Toward defining the course of evolution : Minimum change for a specific tree topology. *Systematic Biology*, 20:406–416, 1971.

[38] Paul FLICEK, M Ridwan AMODE, Daniel BARRELL, Kathryn BEAL, Simon BRENT, Denise CARVALHO-SILVA, Peter CLAPHAM, Guy COATES, Susan FAIRLEY, Stephen FITZGERALD *et al.* : Ensembl 2012. *Nucleic Acids Research*, 40:D84–D90, 2012.

[39] James FRANKLIN : *The Science of Conjecture : Evidence and Probability Before Pascal*. Johns Hopkins University Press, 2001.

[40] Nicolas GALTIER : Maximum-likelihood phylogenetic analysis under a covarion-like model. *Molecular Biology and Evolution*, 18:866–873, 2001.

[41] Feng GAO, Elizabeth BAILES, David L ROBERTSON, Yalu CHEN, Cynthia M RODENBURG, Scott F MICHAEL, Larry B CUMMINS, Larry O ARTHUR, Martine PEETERS, George M SHAW *et al.* : Origin of HIV-1 in the chimpanzee Pan troglodytes troglodytes. *Nature*, 397:436–441, 1999.

[42] Olivier GASCUEL et Mike STEEL : Neighbor-joining revealed. *Molecular Biology and Evolution*, 23:1997–2000, 2006.

[43] Manolo GOUY et Stéphane DELMOTTE : Remote access to ACNUC nucleotide and protein sequence databases at PBIL. *Biochimie*, 90:555–562, 2008.

[44] Manolo GOUY, Stéphane GUINDON et Olivier GASCUEL : Seaview version 4 : a multiplatform graphical user interface for sequence alignment and phylogenetic tree building. *Molecular Biology and evolution*, 27:221–224, 2010.

[45] Mendel GREGOR : Versuche über pflanzenhybriden. *Verhandlungen des Naturforschenden Vereines in Brünn*, Bd. IV für das Jahr 1865:3–47, 1866.

[46] Fred GRIFFITH : The significance of pneumococcal types. *Journal of Hygiene*, 27:113–159, 1928.

[47] M GROUSSIN, B BOUSSAU et M GOUY : A branch-heterogeneous model of protein evolution for efficient inference of ancestral sequences. *Systematic Biology*, 2013.

[48] Stéphane GUINDON, Jean-François DUFAYARD, Vincent LEFORT, Maria ANISIMOVA, Wim HORDIJK et Olivier GASCUEL : New algorithms and methods to estimate maximum-likelihood phylogenies : assessing the performance of PhyML 3.0. *Systematic Biology*, 59:307–321, 2010.

[49] Laurent GUÉGUEN, Sylvain GAILLARD, Bastien BOUSSAU, Manolo GOUY, Mathieu GROUSSIN, Nicolas C. ROCHETTE, Thomas BIGOT, David FOURNIER, Fanny POUYET, Vincent CAHAIS, Aurélien BERNARD, Céline SCORNAVACCA, Benoît F. NABHOLZ, Annabelle HAUDRY, Loïc DACHARY, Nicolas GALTIER, Khalid BELKHIR et Julien Yann DUTHEIL : Bio++ : efficient, extensible libraries and tools for computational molecular evolution. *Molecular Biology and Evolution*, accepted, 2013.

[50] Mira V. HAN et Christian M. ZMASEK : phyloXML : XML for evolutionary biology and comparative genomics. *BMC Bioinformatics*, 10:356, 2009.

[51] W Keith HASTINGS : Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57:97–109, 1970.

[52] Paulien HOGEWEG et Ben HESPER : The alignment of sets of sequences and the construction of phyletic trees : an integrated method. *Journal of Molecular Evolution*, 20:175–186, 1984.

[53] Thérèse A HOLTON et Davide PISANI : Deep genomic-scale analyses of the metazoa reject coelomata : evidence from single-and multigene families analyzed under a supertree and supermatrix paradigm. *Genome Biology and Evolution*, 2:310, 2010.

[54] Tim HUBBARD, Daniel BARKER, Ewan BIRNEY, Graham CAMERON, Yuan CHEN, L CLARK, T COX, J CUFF, Val CURWEN, Thomas DOWN *et al.* : The Ensembl genome database project. *Nucleic Acids Research*, 30:38–41, 2002.

[55] John P. HUELSENBECK, Fredrik RONQUIST *et al.* : MRBAYES : Bayesian inference of phylogenetic trees. *Bioinformatics*, 17:754–755, 2001.

[56] Jaime HUERTA-CEPAS, Salvador CAPELLA-GUTIERREZ, Leszek P PRYSZCZ, Ivan DENISOV, Diego KORMES, Marina MARCET-HOUBEN et Toni GABALDÓN : PhylomeDB v3. 0 : an expanding repository of genome-wide collections of trees, alignments and phylogeny-based orthology and paralogy predictions. *Nucleic Acids Research*, 39:D556–D560, 2011.

[57] Ravi JAIN, Maria C RIVERA et James A LAKE : Horizontal gene transfer among genomes : the complexity hypothesis. *Proceedings of the National Academy of Sciences of the United States of America*, 96:3801–3806, 1999.

[58] Gangolf JOBB, Arndt VON HAESELER et Korbinian STRIMMER : TREEFINDER : a powerful graphical analysis environment for molecular phylogenetics. *BMC Evolutionary Biology*, 4:18, 2004.

[59] David T JONES, William R TAYLOR et Janet M THORNTON : The rapid generation of mutation data matrices from protein sequences. *Computer Applications in the Biosciences*, 8:275–282, 1992.

[60] Jordi JORDANA, Oriol RIBO et M PELEGRIN : Analysis of genetic relationships from morphological characters in spanish goat breeds. *Small Ruminant Research*, 12:301–314, 1993.

[61] Thomas H JUKES et Charles R CANTOR : *Mammalian protein metabolism*, chapitre Evolution of protein molecules, pages 21–132. Academic Press, New York, 1969.

[62] Pekka KILPELÄINEN et Heikki MANNILA : Retrieval from hierarchical texts by partial patterns. *In Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '93, pages 214–222, New York, 1993.

[63] Motoo KIMURA : A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *Journal of Molecular Evolution*, 16:111–120, 1980.

[64] William KLIMKE, Richa AGARWALA, Azat BADRETDIN, Slava CHETVERNIN, Stacy CIUFO, Boris FEDOROV, Boris KIRYUTIN, Kathleen O'NEILL, Wolfgang RESCH, Sergei RESENCHUK *et al.* : The national center for biotechnology information's protein clusters database. *Nucleic Acids Research*, 37:D216–D223, 2009.

[65] Eugene V. KOONIN : Orthologs, paralogs, and evolutionary genomics. *Annual Review of Genetics*, 39:309–338, 2005.

[66] Jos KÄFER, Martina TALIANOVÁ, Thomas BIGOT, Elleni MICHU, Laurent GUÉGUEN, Alex WIDMER, Sylvain GLÉMIN et Gabriel AB MARAIS : Patterns of molecular evolution in dioecious and non-dioecious silene. *Journal of Evolutionary Biology*, 26:335–346, 2013.

[67] M. A. LARKIN, G. BLACKSHIELDS, N. P. BROWN, R. CHENNA, P. A. MCGETTIGAN, H. MCWILLIAM, F. VALENTIN, I. M. WALLACE, A. WILM, R. LOPEZ, J. D. THOMPSON, T. J. GIBSON et D. G. HIGGINS : Clustal W and Clustal X version 2.0. *Bioinformatics*, 23:2947–2948, 2007.

[68] Nicolas LARTILLOT, Thomas LEPAGE et Samuel BLANQUART : PhyloBayes 3 : a Bayesian software package for phylogenetic reconstruction and molecular dating. *Bioinformatics*, 25:2286–2288, 2009.

[69] Nicolas LARTILLOT et Hervé PHILIPPE : Improvement of molecular phylogenetic inference and the phylogeny of bilateria. *Philosophical Transactions of the Royal Society B : Biological Sciences*, 363:1463–1472, 2008.

[70] Jeffrey G LAWRENCE et Howard OCHMAN : Molecular archaeology of the *Escherichia coli* genome. *Proceedings of the National Academy of Sciences of the United States of America*, 95:9413–9417, 1998.

[71] Si Quang LE, Nicolas LARTILLOT et Olivier GASCUEL : Phylogenetic mixture models for proteins. *Philosophical Transactions of the Royal Society B : Biological Sciences*, 363:3965–3976, 2008.

[72] Joshua LEDERBERG *et al.* : Cell genetics and hereditary symbiosis. *Physiological Reviews*, 32:403–30, 1952.

[73] Joshua LEDERBERG et Edward L TATUM : Gene recombination in *Escherichia coli*. *Nature*, 158:558, 1946.

[74] Shonda A LEONARD : IUPAC/IUB Single-Letter Codes Within Nucleic Acid and Amino Acid Sequences. *Current Protocols in Bioinformatics*, 2002.

[75] Wen-Hsiung LI : So, what about the molecular clock hypothesis ? *Current Opinion in Genetics & Development*, 3:896–901, 1993.

[76] Laurence LOEWE : Negative selection. *Nature Education*, 1, 2008.

[77] Yaniv LOEWENSTEIN, Elon PORTUGALY, Menachem FROMER et Michal LINIAL : Efficient algorithms for accurate hierarchical clustering of huge datasets : tackling the entire protein space. *Bioinformatics*, 24:i41–i49, 2008.

[78] P LOPEZ, D CASANE et H PHILIPPE : Heterotachy, an important process of protein evolution. *Molecular Biology and Evolution*, 19:1–7, 2002.

[79] Michele MAGRANE *et al.* : UniProt Knowledgebase : a hub of integrated protein data. *Database : The Journal of Biological Databases and Curation*, 2011.

[80] Jon MALLATT et Christopher J WINCHELL : Testing the new animal phylogeny : first use of combined large-subunit and small-subunit rRNA gene sequences to classify the protostomes. *Molecular Biology and Evolution*, 19:289–301, 2002.

[81] Michaël MANUEL, Michael KRUSE, Werner EG MÜLLER et Yannick LE PARCO : The comparison of β-thymosin homologues among metazoa supports an arthropod-nematode clade. *Journal of Molecular Evolution*, 51:378–381, 2000.

[82] Gabriel AB MARAIS, Alan FORREST, Esther KAMAU, Jos KÄFER, Vincent DAUBIN et Deborah CHARLESWORTH : Multiple nuclear gene phylogenetic analysis of the evolution of dioecy and sex chromosomes in the genus silene. *PLOS One*, 6:e21915, 2011.

[83] Ken MARTIN, Bill HOFFMAN, Andy CEDILNIK, Brad KING et Alex NUENDORF : *Mastering CMake : A cross-platform build system*. Kitware Incorporated, 2003.

[84] Robert W. MEREDITH, Jan E. JANEČKA, John GATESY, Oliver A. RYDER, Colleen A. FISHER, Emma C. TEELING, Alisha GOODBLA, Eduardo EIZIRIK, Taiz L L. SIMÃO, Tanja STADLER, Daniel L. RABOSKY, Rodney L. HONEYCUTT, John J. FLYNN, Colleen M. INGRAM, Cynthia STEINER, Tiffani L. WILLIAMS, Terence J. ROBINSON, Angela BURK-HERRICK, Michael WESTERMAN, Nadia A. AYOUB, Mark S. SPRINGER et William J. MURPHY : Impacts of the cretaceous terrestrial revolution and kpg extinction on mammal diversification. *Science*, 334:521–524, 2011.

[85] Nicholas METROPOLIS, Arianna W ROSENBLUTH, Marshall N ROSENBLUTH, Augusta H TELLER et Edward TELLER : Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21:1087, 1953.

[86] Vincent MIELE, Simon PENEL et Laurent DURET : Ultra-fast sequence clustering from similarity networks with SiLiX. *BMC Bioinformatics*, 12:116, 2011.

[87] Boris MIRKIN, Ilya MUCHNIK et Temple F SMITH : A biologically consistent model for comparing molecular phylogenies. *Journal of Computational Biology*, 2:493–507, 1995.

[88] William J MURPHY, Eduardo EIZIRIK, Warren E JOHNSON, Ya Ping ZHANG, Oliver A RYDER et Stephen J O'BRIEN : Molecular phylogenetics and the origins of placental mammals. *Nature*, 409:614–618, 2001.

[89] NATIONAL CENTER FOR BIOTECHNOLOGY INFORMATION : NCBI - Taxonomy tools - Statistics, Novembre 2012. `http://www.ncbi.nlm.nih.gov/Taxonomy/taxonomyhome.html/index.cgi?chapter=statistics&uncultured=hide&unspecified=hide`.

[90] Jerzy NEYMAN : Molecular studies of evolution : a source of novel statistical problems. *In* S. S. GUPTA et J. YACKEL, éditeurs : *Statistical decision theory and related topics*, pages 1–27. Academic Press, New York, 1971.

[91] James R NORRIS : *Markov chains. Cambridge series in statistical and probabilistic mathematics*. Cambridge University Press, Cambridge, 1997.

[92] Cédric NOTREDAME, Desmond G HIGGINS, Jaap HERINGA *et al.* : T-Coffee : A novel method for fast and accurate multiple sequence alignment. *Journal of Molecular Biology*, 302:205–218, 2000.

[93] Howard OCHMAN, Jeffrey G LAWRENCE et Eduardo A GROISMAN : Lateral gene transfer and the nature of bacterial innovation. *Nature*, 405:299–304, 2000.

[94] Susumu OHNO *et al.* : *Sex chromosomes and sex-linked genes.(Monographs on endocrinology, Vol. 1.)*. Berlin, Heidelberg, New York : Springer Verlag, 1967.

[95] Maureen A. O'LEARY, Jonathan I. BLOCH, John J. FLYNN, Timothy J. GAUDIN, Andres GIALLOMBARDO, Norberto P. GIANNINI, Suzann L. GOLDBERG, Brian P. KRAATZ, Zhe-Xi LUO, Jin MENG, Xijun NI, Michael J. NOVACEK, Fernando A. PERINI, Zachary S. RANDALL, Guillermo W. ROUGIER, Eric J. SARGIS, Mary T. SILCOX, Nancy B. SIMMONS, Michelle SPAULDING, Paúl M. VELAZCO, Marcelo WEKSLER, John R. WIBLE et Andrea L. CIRRANELLO : The placental mammal ancestor and the post-k-pg radiation of placentals. *Science*, 339:662–667, 2013.

[96] Roderic DM PAGE et Michael A CHARLESTON : From gene to organismal phylogeny : reconciled trees and the gene tree/species tree problem. *Molecular Phylogenetics and Evolution*, 7:231–240, 1997.

[97] Simon PENEL, Anne-Muriel ARIGON, Jean-François DUFAYARD, Anne-Sophie SERTIER, Vincent DAUBIN, Laurent DURET, Manolo GOUY et Guy PERRIÈRE : Databases of homologous gene families for comparative genomics. *BMC Bioinformatics*, 10:S3, 2009.

[98] Guy PERRIÈRE et Céline BROCHIER-ARMANET : *Concepts et méthodes en phylogénie moléculaire*. Springer, Paris, 2010.

[99] Guy PERRIERE, Christophe COMBET, Simon PENEL, Christophe BLANCHET, Jean THIOULOUSE, Christophe GEOURJON, Julien GRASSOT, Celine CHARAVAY, Manolo GOUY, Laurent DURET *et al.* : Integrated databanks access and sequence/structure analysis services at the PBIL. *Nucleic Acids Research*, 31:3393–3399, 2003.

[100] Sean POWELL, Damian SZKLARCZYK, Kalliopi TRACHANA, Alexander ROTH, Michael KUHN, Jean MULLER, Roland ARNOLD, Thomas RATTEI, Ivica LETUNIC, Tobias DOERKS *et al.* : eggNOG v3. 0 : orthologous groups covering 1133 organisms at 41 different taxonomic ranges. *Nucleic Acids Research*, 40:D284–D289, 2012.

[101] Morgan N PRICE, Paramvir S DEHAL et Adam P ARKIN : Fasttree : computing large minimum evolution trees with profiles instead of a distance matrix. *Molecular Biology and Evolution*, 26:1641–1650, 2009.

[102] Marco PUNTA, Penny C COGGILL, Ruth Y EBERHARDT, Jaina MISTRY, John TATE, Chris BOURSNELL, Ningze PANG, Kristoffer FORSLUND, Goran CERIC, Jody CLEMENTS *et al.* : The Pfam protein families database. *Nucleic Acids Research*, 40:D290–D301, 2012.

[103] Adolphe QUETELET : *Correspondance Mathématique Et Physique, Volume 3 (Réimpression Fac Simile 2010)*, volume 4. Observatoire Royal de Belgique, 1838.

[104] Anja RAUTENBERG, Louise HATHAWAY, Bengt OXELMAN et Honor C. PRENTICE : Geographic and phylogenetic patterns in Silene section Melandrium (Caryophyllaceae) as inferred from chloroplast and nuclear DNA sequences. *Molecular Phylogenetics and Evolution*, 57:978–991, 2010.

[105] Susanne S RENNER et Robert E RICKLEFS : Dioecy and its correlates in the flowering plants. *American Journal of Botany*, 1995.

[106] Fredrik RONQUIST et John P HUELSENBECK : MrBayes 3 : Bayesian phylogenetic inference under mixed models. *Bioinformatics*, 19:1572–1574, 2003.

[107] Jue RUAN, Heng LI, Zhongzhong CHEN, Avril COGHLAN, Lachlan James M COIN, Yiran GUO, Jean-Karim HERICHE, Yafeng HU, Karsten KRISTIANSEN, Ruiqiang LI *et al.* : Treefam : 2008 update. *Nucleic Acids Research*, 36:D735–D740, 2008.

[108] Iñaki RUIZ-TRILLO, Jordi PAPS, Mercè LOUKOTA, Carles RIBERA, Ulf JONDELIUS, Jaume BAGUÑÀ et Marta RIUTORT : A phylogenetic analysis of myosin heavy chain type ii sequences corroborates that acoela and nemertodermatida are basal bilaterians. *Proceedings of the National Academy of Sciences*, 99:11246–11251, 2002.

[109] Andre RZHETSKY et Masatoshi NEI : Theoretical foundation of the minimum-evolution method of phylogenetic inference. *Molecular Biology and Evolution*, 10:1073–1095, 1993.

[110] Andrey RZHETSKY et Masatoshi NEI : A simple method for estimating and testing minimum-evolution trees. *Molecular Biology and Evolution*, 9:945–967, 1992.

[111] Eric W. SAYERS, Tanya BARRETT, Dennis A. BENSON, Stephen H. BRYANT, Kathi CANESE, Vyacheslav CHETVERNIN, Deanna M. CHURCH, Michael DICUCCIO, Ron EDGAR, Scott FEDERHEN, Michael FEOLO, Lewis Y. GEER, Wolfgang HELMBERG, Yuri KAPUSTIN, David LANDSMAN, David J. LIPMAN, Thomas L. MADDEN, Donna R. MAGLOTT, Vadim MILLER, Ilene MIZRACHI, James OSTELL, Kim D. PRUITT, Gregory D. SCHULER, Edwin SEQUEIRA, Stephen T. SHERRY, Martin SHUMWAY, Karl SIROTKIN, Alexandre

SOUVOROV, Grigory STARCHENKO, Tatiana A. TATUSOVA, Lukas WAGNER, Eugene YASCHENKO et Jian YE : Database resources of the national center for biotechnology information. *Nucleic Acids Research*, 37:D5–15, 2009.

[112] Florence SERVANT, Catherine BRU, Sébastien CARRÈRE, Emmanuel COURCELLE, Jérôme GOUZY, David PEYRUC et Daniel KAHN : ProDom : automated clustering of homologous domains. *Briefings in Bioinformatics*, 3:246–251, 2002.

[113] Claude E SHANNON et Warren WEAVER : *The Mathematical Theory of Communication*. University of Illinois Press, 1971.

[114] Fabian SIEVERS, Andreas WILM, David DINEEN, Toby J. GIBSON, Kevin KARPLUS, Weizhong LI, Rodrigo LOPEZ, Hamish MCWILLIAM, Michael REMMERT, Johannes SÖDING, Julie D. THOMPSON et Desmond G. HIGGINS : Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. *Molecular Systems Biology*, 7:539, 2011.

[115] Temple F SMITH et Michael S WATERMAN : Comparison of biosequences. *Advances in Applied Mathematics*, 2:482–489, 1981.

[116] Mark S SPRINGER : Molecular clocks and the timing of the placental and marsupial radiations in relation to the cretaceous–tertiary boundary. *Journal of Mammalian Evolution*, 4:285–302, 1997.

[117] Alexandros STAMATAKIS, Thomas LUDWIG et Harald MEIER : RAxML-III : a fast program for maximum likelihood-based inference of large phylogenetic trees. *Bioinformatics*, 21:456–463, 2005.

[118] Mario STANKE, Oliver KELLER, Irfan GUNDUZ, Alec HAYES, Stephan WAACK et Burkhard MORGENSTERN : AUGUSTUS : ab initio prediction of alternative transcripts. *Nucleic Acids Research*, 34:W435–W439, 2006.

[119] Gergely J SZÖLLŐSI, Bastien BOUSSAU, Sophie S ABBY, Eric TANNIER et Vincent DAUBIN : Phylogenetic modeling of lateral gene transfer reconstructs the pattern and relative timing of speciations. *Proceedings of the National Academy of Sciences*, 109:17513–17518, 2012.

[120] Roman L TATUSOV, Eugene V KOONIN et David J LIPMAN : A genomic perspective on protein families. *Science*, 278:631–637, 1997.

[121] Roman L TATUSOV, Arcady R MUSHEGIAN, Peer BORK, Nigel P BROWN, William S HAYES, Mark BORODOVSKY, Kenneth E RUDD, Eugene V KOONIN *et al.* : Metabolism and evolution of Haemophilus influenzae deduced from a whole-genome comparison with *Escherichia coli*. *Current Biology*, 6:279–291, 1996.

[122] John S. TAYLOR et Jeroen RAES : Duplication and divergence : the evolution of new genes and old ideas. *Annual Review of Genetics*, 38:615–643, 2004.

[123] Julie D THOMPSON, Desmond G HIGGINS et Toby J GIBSON : CLUSTAL W : improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22:4673–4680, 1994.

[124] Jeffrey L THORNE, Hirohisa KISHINO et Ian S PAINTER : Estimating the rate of evolution of the rate of molecular evolution. *Molecular Biology and Evolution*, 15:1647–1657, 1998.

[125] Jana C VAMOSI, Sally P OTTO et Spencer CH BARRETT : Phylogenetic analysis of the ecological correlates of dioecy in angiosperms. *Journal of Evolutionary Biology*, 16:1006–1018, 2003.

[126] Vito VOLTERRA et Marcel BRELOT : *Leçons sur la Théorie mathématique de la lutte pour la vie (Réimpression fac simile 1990)*. Jacques Gabay, 1931.

[127] Rutger A. VOS, James P. BALHOFF, Jason A. CARAVAS, Mark T. HOLDER, Hilmar LAPP, Wayne P. MADDISON, Peter E. MIDFORD, Anurag PRIYAM, Jeet SUKUMARAN, Xuhua XIA et Arlin STOLTZFUS : Nexml : rich, extensible, and verifiable representation of comparative data and metadata. *Systems Biology*, 61:675–689, 2012.

[128] Peter J WADDELL et Shawn SHELLEY : Evaluating placental inter-ordinal phylogenies with novel sequences including RAG1, γ-fibrinogen, ND6, and mt-

tRNA, plus MCMC-driven nucleotide, amino acid, and codon models. *Molecular Phylogenetics and Evolution*, 28:197–224, 2003.

[129] Simon WHELAN et Nick GOLDMAN : A general empirical model of protein evolution derived from multiple protein families using a maximum-likelihood approach. *Molecular Biology and Evolution*, 18:691–699, 2001.

[130] (Auteurs Divers) WIKIPEDIA : Ordre de grandeur (nombres), 2012 (consulté en octobre 2012). http://fr.wikipedia.org/wiki/Ordre_de_grandeur_%28nombres%29.

[131] Yuri I WOLF, Igor B ROGOZIN et Eugene V KOONIN : Coelomata and not ecdysozoa : evidence from genome-wide phylogenetic analysis. *Genome Research*, 14:29–36, 2004.

[132] Ziheng YANG : Maximum likelihood phylogenetic estimation from dna sequences with variable rates over sites : approximate methods. *Journal of Molecular Evolution*, 39:306–314, 1994.

[133] Ziheng YANG : Likelihood ratio tests for detecting positive selection and application to primate lysozyme evolution. *Molecular biology and evolution*, 15:568–573, 1998.

[134] Ziheng YANG : PAML 4 : phylogenetic analysis by maximum likelihood. *Molecular Biology and Evolution*, 24:1586–1591, 2007.

[135] Ziheng YANG et Bruce RANNALA : Bayesian phylogenetic inference using DNA sequences : a Markov chain Monte Carlo method. *Molecular Biology and Evolution*, 14:717–724, 1997.

[136] Louxin ZHANG : On a Mirkin-Muchnik-Smith conjecture for comparing molecular phylogenies. *Journal of Computational Biology*, 4:177–187, 1997.

[137] Jie ZHENG, Igor B ROGOZIN, Eugene V KOONIN et Teresa M PRZYTYCKA : Support for the coelomata clade of animals from a rigorous analysis of the pattern of intron conservation. *Molecular Biology and Evolution*, 24:2583–2592, 2007.

[138] Norton D ZINDER et Joshua LEDERBERG : Genetic exchange in Salmonella. *Journal of Bacteriology*, 64:679, 1952.

[139] Emile ZUCKERKANDL et Linus PAULING : *Molecular disease, evolution and genetic heterogeneity*. Horizons in Biochemistry. Academic Press, New York, 1962.

A

Articles publiés

A.1 TPMS, a set of utilities for querying collections of gene trees

Auteurs : Thomas Bigot, Vincent Daubin, Florent Lassalle, Guy Perrière

Revue : BMC Bioinformatics

Statut : Publié

URL : `http://www.biomedcentral.com/1471-2105/14/109/abstract`

Référence Bibliographique : [11]

Bigot *et al. BMC Bioinformatics* 2013, **14**:109
http://www.biomedcentral.com/1471-2105/14/109

BMC Bioinformatics

SOFTWARE Open Access

TPMS: a set of utilities for querying collections of gene trees

Thomas Bigot, Vincent Daubin, Florent Lassalle and Guy Perrière*

Abstract

Background: The information in large collections of phylogenetic trees is useful for many comparative genomic studies. Therefore, there is a need for flexible tools that allow exploration of such collections in order to retrieve relevant data as quickly as possible.

Results: In this paper, we present TPMS (Tree Pattern-Matching Suite), a set of programs for handling and retrieving gene trees according to different criteria. The programs from the suite include utilities for tree collection building, specific tree-pattern search strategies and tree rooting. Use of TPMS is illustrated through three examples: systematic search for incongruencies in a large tree collection, a short study on the Coelomata/Ecdysozoa controversy and an evaluation of the level of support for a recently published Mammal phylogeny.

Conclusion: TPMS is a powerful suite allowing to quickly retrieve sets of trees matching complex patterns in large collection or to root trees using more rigorous approaches than the classical midpoint method. As it is made of a set of command-line programs, it can be easily integrated in any sequence analysis pipeline for an automated use.

Background

Comparative genomics is a central approach in sequence analysis, and many important biological results have been obtained through its use. Among the different programs and packages developed for comparative genomics, those using the information contained in phylogenetic trees are of special interest. Indeed, the explicative power brought by trees has still no match for problems like orthology detection, Horizontal Gene Transfer (HGT) prediction, or large-scale evolutionary studies. For orthologs detection, the most rigorous approach to determine whether homologous genes are orthologous or paralogous consists in comparing a gene tree to the species tree considered as a reference [1-4]. Similarly, the same approach can be used for HGT detection [5]. Lastly, various kind of evolutionary studies have been made possible only through the use of large collection of trees, *e.g.*, detection of positive selection in vertebrate evolution [6], analysis of DNA methylation levels in mammals [7], or GC-content evolution [8].

The problem is that manual browsing of collections containing thousands of gene trees is a tedious – and now, almost impossible to perform – task, and automated tools are required for large scale studies. Therefore, we have developed TPMS, a set of programs devoted to tree manipulation and query. Its central functionality is a pattern-matching algorithm aimed at finding gene trees containing specific motifs. Those patterns are written using an extended Newick format and they usually include some kind of constraint, such as node nature (duplication, speciation), subtree content, or level of statistical support for the nodes (*e.g.*, bootstrap, jackknife or aLRT). The second main functionality brought by TPMS is a tree-rooting tool based on the use of a gene unicity score and taxonomic criteria. Its aim is to place the root of a gene tree in order to minimize the incongruencies observed between this tree and a reference species tree.

The tree pattern search algorithm used by TPMS is an improved version of the one we previously published [1]. The main improvement is the possibility to query tree collections built by the users. Our previous version was implemented in FamFetch, a Graphical User Interface (GUI) that was only able to query a set of predefined databases installed on a centralized server [9].

*Correspondence: guy.perriere@univ-lyon1.fr
Laboratoire de Biométrie et Biologie Évolutive, UMR CNRS 5558, Université Claude Bernard – Lyon 1, 43 bd. du 11 Novembre 1918, 69622 Villeurbanne Cedex, France

Implementation

TPMS is a set of C++ command-line programs that require the Bio++ [10] and Boost (http://www.boost.org/) libraries to run and it is not dependent on the use of the gene families databases developed in our group [9]. Binaries are provided for Linux and MacOSX (Intel architectures only), as well as C++ source code and documentation.

Tree collection building

TPMS needs to be run on a collection built in the RAP format [1]. With this format, all the trees are included in a single text file. The header of this file consists in a reference species tree in Newick format that contains taxa names on its internal nodes and leaves. Names on leaves correspond to species and names on internal nodes correspond to higher taxonomic groups. All taxonomic groups allow the use of synonyms. For instance, in the example file distributed with TPMS, the leaf corresponding to *Escherichia coli* strain K12 substrain W3110 is written as:

```
"ESCHERICHIA COLI W3110"/"ESCHERICHIA COLI
STR. K12 SUBSTR. W3110"/
"ESCHERICHIA COLI STR.W3110"/"ESCHERICHIA
COLI STRAIN W3110"
```

The different synonyms are separated by a slash and it is possible to use any of them when building a query.

Individual gene trees are listed after the reference species tree. Each entry is written in Newick format and has associated information consisting in the tree name and a list of associations between the sequences names used at the leaves and their corresponding species names:

```
HBG469283.phb
[
ECO7I_1_PE647"ESCHERICHIA COLI IAI39"
ESCOL4_2_PE678"ESCHERICHIA COLI STR. K-12
SUBSTR. MG1655"
ESCOL5_1_PE669"ESCHERICHIA COLI STR. K-12
SUBSTR. W3110
SHIBS_2_PE510"SHIGELLA BOYDII SB227"
]
(ECO7I_1_PE647:0.00591,((ESCOL4_2_PE678:0.0,
ESCOL5_1_PE669:0.0)-1:0.0,
SHIBS_2_PE510:0.0)-1:0.00591)-1;
```

The species tree can contain unresolved nodes (multifurcations), but not the individual gene trees since the tree pattern matching algorithm used only supports binary trees [1].

The program *tpms_mkdb* is provided in order to produce a collection in the suitable format. To run this program, the minimal requirements are a species tree file and a collection of gene trees in Newick format. Using the information provided in these two files, the user must then build another file containing the association between all the sequences names used in the gene trees and the species to which they belong. Note that *tpms_mkdb* provides a functionality for facilitating this task. This functionality returns a ready-to-complete list of all the sequences names found in the gene trees files. Then the user has to fill the blanks with corresponding species names.

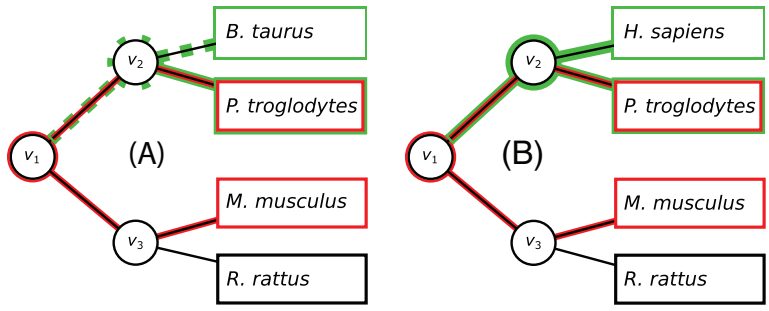
Tree pattern-matching

The tree pattern-matching algorithm itself is implemented in the *tpms_query* program. This is a C++ version of the one used in FamFetch, and a complete description of the algorithm can be found in [1]. Like in the original program, both the target tree and the tree pattern need to be rooted.

Tree patterns have to be written in an extended Newick format, and they can include different criteria. Let υ be a node from the pattern P, and let ω and φ be the two sons of υ. In our format, υ is described by three strings separated by slashes. The first two strings contains the constraints put on υ, whereas the third one – which is only suitable for leaves – describes the set of taxa allowed at this level. The simplest pattern only contains the list of taxa to be found on the leaves with their respective positions. For instance, the pattern `((//Homo sapiens, //Pan troglodytes), //Rodentia)` will find all the gene trees in which a subtree with sequences from *Homo sapiens* and *Pan troglodytes* species are grouped, while sequences from Rodents are located outside of this group. It is possible to accurately ask for the inclusion or the exclusion of specific taxa through the use of + and − signs. In this case, the pattern `((//Homo sapiens, //Pan troglodytes), //Mammalia−Primates)` will find all the gene trees in which a subtree with sequences from *H. sapiens* and *P. troglodytes* species are grouped, while any sequences from Mammals that are not from Primates are located outside this group. This pattern is less constrained than the first one, and the trees it selects will include all those selected by the first one.

It is also possible to introduce taxonomic constraints on one of the two sons of the node matching to υ in the gene tree. This constraint is written on ω, but is put on its father node since the node matching on ω is not necessarily the direct son of the node matching on υ. This kind of constraint is put after the first slash, and examples of accepted and rejected topologies for a simple pattern are given in Figure 1.

Constraints other than purely taxonomic ones can be introduced on nodes. First, it is possible to search for patterns including a threshold for internal branches statistical




```
(/Primates/Pan troglodytes, //Mus musculus);
```

Figure 1 Example of a subtree constraint. In the pattern shown in the lower part of the figure, a constraint has been put on the node leading to *P. troglodytes*. The pattern is shown in red, and the subtree constraint is shown in green. This pattern with this constraint will find all the trees in which a subtree with sequences from *P. troglodytes* and *M. musculus* species are grouped. Non-Primates are forbidden from v_1, the common ancestor of *P. troglodytes* and *M. musculus*, to the node matching to *P. troglodytes*. Tree (**A**) is rejected, since the pattern does not allow *B. taurus* (which is not a Primate) to be in the subtree containing *P. troglodytes* (generated by v_1). On the other hand, tree (**B**) is accepted, as the subtree generated by v_1 contains only primates.

support. For instance, the pattern `(//Homo sapiens, //Mus musculus)$90` will select all the gene trees in which a subtree with sequences from *H. sapiens* and *Mus Musculus* are grouped, this with an associated bootstrap $\geq$ 90%. Also, constraints on speciation or duplication can be set on nodes if the program is running on a reconciled trees collection. This kind of nodes can be specified by the use of letters S or D. The pattern `(//Homo sapiens, //Mus musculus)D` will find all the gene trees in which a subtree with sequences from *H. sapiens* and *M. musculus* are grouped, while the node that groups them is a duplication node. Lastly, it is possible to specify that a connection between two nodes in the gene tree is direct. This is specified through the use of the exclamation mark. Therefore, the pattern `(!//Homo sapiens, //Mus musculus)` will retrieve only gene trees in which the common ancestor of *H. sapiens* and *Mus musculus* has the leaf *H. sapiens* as a direct son.

Tree rooting

Tree rooting is an essential step in phylogeny as it orients the tree and enables the evolutionary history it summarizes to be interpreted. We have therefore implemented two rooting strategies in the *tpms_computation* program: the first one aims at maximizing the size of subtrees with unicopy sequences while the second one tries to maximize the accuracy of taxonomic affectations for nodes. The first approach introduces a score on the internal nodes of a rooted tree. Given a node v_i ($i = 1, 2, \ldots, n-1$), its score u_i is obtained by the product of the number of sequences by species encountered in the subtree generated by the node considered. A score equal to 1 means that the corresponding subtree only contains one sequence for each species. Now, let T be a rooted tree, its unicity score $U^{(T)}$ is equal to:

$$U^{(T)} = \sum_{i=1}^{n-1} \ln\left(u_i^{(T)}\right) \quad (1)$$

In a unrooted tree with n leaves, there are $2n-3$ possible positions for the root. TPMS will try every position, and for each one, computes the unicity score of the corresponding rooted tree. Then, the rooting that is kept is the one that minimizes $U^{(T)}$. This approach favors duplications closer to the root than to the leaves. It minimizes the number of duplication events, and is therefore based on a parsimony reasoning. This method can be useful to extract unicopy subtrees from gene families containing many copies for some species. It can automatically split trees containing paralogs into several subtrees. Figure 2 shows two rooting examples of a simple gene tree containing six sequences from three different species.

The second rooting method uses the reference species tree included in the header of the tree collection. During the rooting procedure, each internal node v_i of the gene tree is associated with the taxonomic group corresponding to the Lowest Common Ancestor (LCA) in the species tree. Then, a distance d_i is computed for each node. This distance is equal to the number of nodes between the root of the species tree, and the node of the taxonomic group in the species tree associated with the node in the gene tree. Note that the taxa from the species tree that are not present in the gene tree are not taken into account in this distance computation. Therefore, the sum of node distances for a rooted tree T is equal to:

$$D^{(T)} = \sum_{i=1}^{n-1} d_i^{(T)} \quad (2)$$

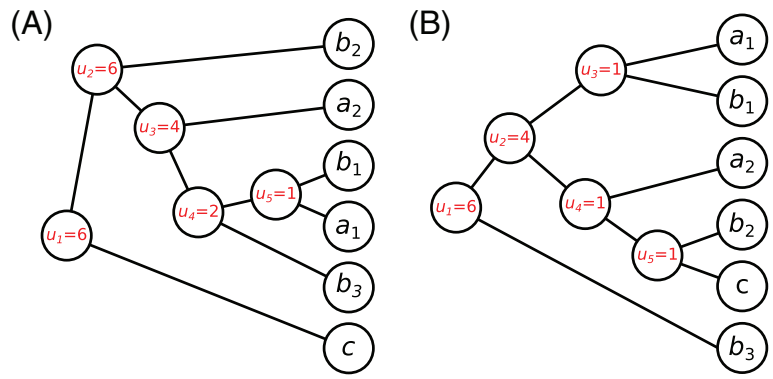


Figure 2 Example of unicity score computing for a gene tree with two different rootings. In this tree, we have six sequences a_1, a_2, b_1, b_2, b_3 and c. Sequences a_1 and a_2 belong to species A; b_1, b_2 and b_3 belong to species B; and c belongs to species C. Scores of internal nodes are given in the circles. In this example, the rooting for topology (**A**) gives the unicity score $U^{(T)} = \ln(6) + \ln(6) + \ln(4) + \ln(2) + \ln(1) \approx 5.663$, and the rooting for topology (**B**) gives the score $U^{(T)} = \ln(6) + \ln(4) + \ln(1) + \ln(1) + \ln(1) \approx 3.178$. The rooting in (**B**) is therefore preferred over the one in (**A**).

Again, TPMS will try all the $2n - 3$ rooting positions and will keep the one maximizing $D^{(T)}$. Indeed, higher $D^{(T)}$ values correspond to better taxonomic affectations (*i.e.*, resulting trees that are more congruent to the reference species tree). Figure 3 shows an example of two possible rootings for a bacterial gene tree, this considering a reference species tree.

Usually, the two approaches are used in sequence, the first one is efficient to isolate clusters of unique sequences, but it can lead to many *ex aequo*. The problem is that, among those *ex aequo*, some of them can contain obviously erroneous polyphyletic groups. On the other hand, the second approach will find the solutions that are more consistent to the species tree. It can be therefore used to solve those *ex aequo* in the right way, as it will remove the solutions that are in violation of the reference. If more candidates still remains after those two steps, the root is placed on the solution having the longest branch. In case of branch lengths equality, placement is arbitrary on one of the remaining possibilities.

Data sets

For the detection of tree incongruencies in complete genomes, we performed the search on the 128674 gene trees available from HOGENOM release 5 (http://pbil.univ-lyon1.fr/databases/hogenom/). Those trees include a total of respectively 2789275 bacterial, 138474 archaeal and 738819 eukaryotic sequences. For the Ecdysozoa/Coelomata controversy and the mammals phylogeny study, we used the 14190 gene trees available

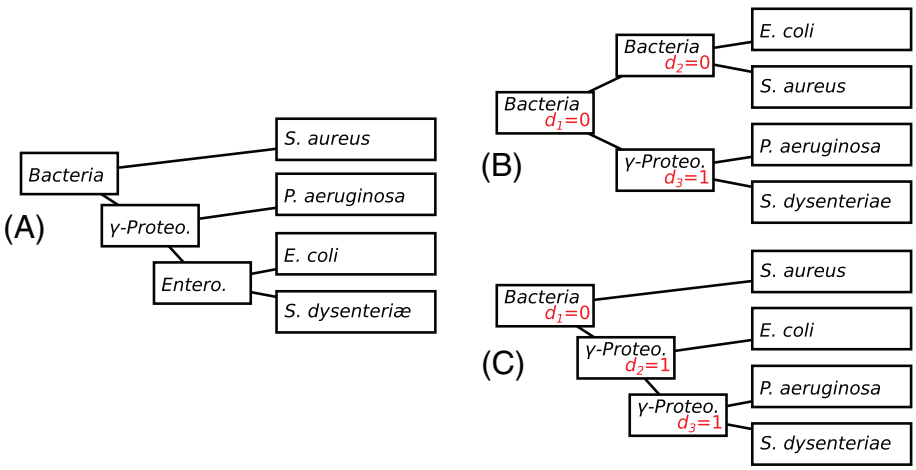


Figure 3 Example of distance score computing for a gene tree with two different rootings. The reference species tree is shown in (**A**), with the names of the taxonomic groups written on the nodes (Bacteria, γ-Proteobacteria and Enterobacteria). The rooting given in (**B**) gives a sum of distances $D^{(T)} = 1 + 0 + 0 = 1$, while the rooting given in (**C**) gives a sum of distances $D^{(T)} = 1 + 1 + 0 = 2$. The rooting in (**C**) is therefore preferred over the one in (**B**).

from HOMOLENS release 5 (http://pbil.univ-lyon1.fr/databases/homolens.php). For the first two studies, the reference species tree used was the one provided by NCBI (ftp://ftp.ncbi.nih.gov/pub/taxonomy/) and for the mammal phylogeny we used a slightly modified version of the tree published by Dos Reis *et al.* [11]. Those modifications were made to match the species tree available in HOMOLENS as three species from [11] were not available in this database: *Pongo abelii*, *Vicugna pacos* and *Canis familiaris*. They were replaced respectively by *Pongo pygmaeus*, *Lama pacos* and a clade grouping *Canis lupus familiaris* and *Ailuropoda melanoleuca*.

Results

Incongruencies detection

The program *tpms_computation* can be used to detect taxonomic incongruencies. As seen above, each node in a gene tree can be assigned to its LCA, this using the information provided by the reference species tree. To detect the incongruencies, the approach used is to see if a leaf or a node in a gene tree induces a perturbation in this taxonomy affectation. Exploration of the tree starts from the leaves and goes up to the root. For each node, the algorithm masks the subtree it generates. If this masking leads to a taxonomic assignation change on its grandfather node, then this subtree is pruned. A perturbation index is associated to this incongruency, it correspond to the distance (in nodes numbers and with ignoring the taxa that are not present in the gene tree) between the former taxonomic affectation and the one realized after the masking process. This operation is repeated until all the existing incongruencies have been detected. Lastly, to be considered as a true incongruency, the involved branch must have a support value greater than a threshold given by the user.

A simple example of detection is given in Figure 4. In this example, *Staphylococcus aureus*, a Firmicute, is wrongly grouped with *Escherichia coli*, an Enterobacteria. Masking the node leading to *S. aureus* leads to a change from Bacteria to Enterobacteria in the taxonomic assignation of its grandfather node. As the number of nodes separating Bacteria from Enterobacteria in the species tree used, the perturbation index associated with this incongruency is equal to 2.

Over the 128674 gene trees available in HOGENOM, *tpms_computation* found 110359 incongruencies supported by bootstrap values $\geq$ 90%, this corresponds to an average of 0.86 errors per tree. Different events could explain the incongruencies observed: phylogenetic reconstruction errors (such as long branch attraction artefacts), errors in the NCBI species tree, wrong gene tree rooting by TPMS, hidden paralogies (as the gene trees from HOGENOM are not reconciled), incomplete lineage sorting and HGTs. Due to that fact, it is remarkable that their frequency is relatively low, with less than 1 error per tree on average.

The number of intra and inter-domain incongruencies detected is shown in Figure 5. An intra-domain incongruency is when a species is placed inside its domain but outside the taxonomic group it belongs to and an inter-domain incongruency is when a species is wrongly located in another domain than the one it belongs to. A good way to see if our method can be used to detect HGTs is to look at its ability to identify well-known transfers such as the ones observed between mitochondria or chloroplast and the nucleus of eukaryotes. For that purpose, we looked at

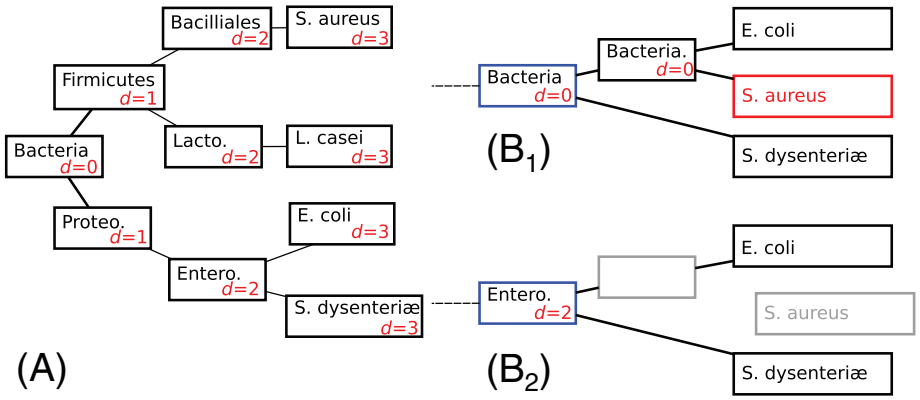


Figure 4 Example of incongruency detection in a subtree. The reference species tree is shown in (**A**), with the names of the taxonomic groups written on the nodes. For each internal node, the distance *d* to the root is written in red. The subtree containing the incongruency is shown in (**B1**), with *S. aureus* being the misplaced taxon. In this subtree, the taxonomic group associated to the grandfather of *S. aureus* is Bacteria and its distance to the root in the species tree is $d = 0$. In (**B2**) the father node of *S. aureus* is removed. The root of this subtree has now two sons *E. coli* and *S. dysenteriae*. The new taxonomic affectation of this node becomes Enterobacteria, which corresponds to a distance to the root $d = 2$, in the species tree. Therefore, perturbation induced by the misplacement of *S. aureus* is equal to $2 - 0 = 2$.

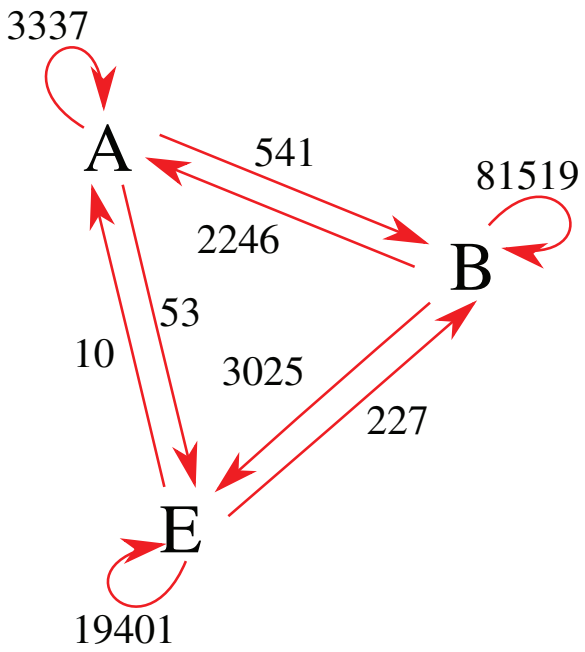


Figure 5 Number of incongruencies inferred between and inside the three domains of life. The three domains (Archaea, Bacteria and Eukaryota) are represented respectively by the letters A, B and E. For inter-domain incongruencies, the arrows give the direction of the switch observed.

the incongruences found with the 374 eukaryotic genes labelled as mitochondrial in HOGENOM 5. Among those genes, 252 (67.4%) were found to be involved in an incongruency placing them among Bacteria. Similarly, among the 377 genes labelled as chloroplastic, 249 (66%) were found to be involved in an incongruency placing them among Bacteria.

Coelomata/Ecdysozoa controversy

The issue of whether coelomates form a single clade, the Coelomata, or whether all animals that moult an exoskeleton (such as the coelomate arthropods and the pseudocoelomate nematodes) form a distinct clade, the Ecdysozoa, is still a major open-ended subject in evolutionary biology. While single-gene based phylogenies supported the Ecdysozoa hypothesis [12-14], a first wave of phylogenomic analyses mostly favored the Coelomata [15-17]. More recently, those results were challenged by other genomic-scale studies [18-20], shifting again the balance toward Ecdysozoa. Here we present a short example showing how it is possible to retrieve the trees supporting either hypothesis.

Using TPMS, the patterns to find are `(//Fungi,(/Nematoda/Nematoda,(/Chordata/Chordata,/Arthropoda/Arthropoda)$80)$80)` and `(//Fungi,(/Chordata/Chordata,(/Nematoda/Nematoda,/Arthropoda/Arthropoda)$80)$80)`, respectively for Coelomata and Ecdysozoa hypotheses with a bootstrap support $\geq$ 80%. We performed the search into a collection built with the HOMOLENS database, this with several thresholds for bootstrap values (Table 1). Excepted for the search performed with no minimal value for bootstrap scores, the results returned were systematically in favor of the Coelomata hypothesis.

Note that this small example is only aimed at showing the possibilities provided by TPMS in terms of tree pattern search. To fully validate the result presented above it would be necessary to go back to the sequence alignments and test for the relative evolutionary rates of individual genes, this in order to avoid systematic biases leading to reconstruction artefacts such as long branch attraction. Indeed, the only nematode genome available in HOGENOM 5 is *Caenorhabditis elegans*, and this organism is known to have genes with high evolutionary rates, leading frequently to such kind of artefacts [21].

Mammals phylogeny

In order to find sets of orthologs in a specific taxon it is possible to query a collection using a pattern corresponding to the subtree of a species tree containing the given taxon. If a gene tree contains exactly this pattern, then the most parsimonious hypothesis is that the matching sequences are orthologous. The search for exact patterns in TPMS requires a direct link between nodes, which is specified by the exclamation mark. For instance, to query a collection to find orthologous genes inside the *Rodentia* order, we must select all the gene trees containing the TPMS pattern `((((!//Rattus Norvegicus,!//Mus Musculus)!,!//Dipodomys ordii)!,!// Cavia Porcellus)!,!// Spermophilus Tridecemlineatus)`. This feature can be used to test the support level for nodes in a species tree. As an example, we used the mammals phylogeny published by Dos Reis *et al.* [11]. For that purpose, we extracted all possible subtrees in this tree and used the corresponding patterns to perform searches in HOMOLENS. Complete listing of the patterns in TPMS format is provided as Additional file 1.

Table 1 Number of tree patterns matching Coelomata or Ecdysozoa hypotheses

	no thr.	≥80%	≥85%	≥90%	≥95%
Coelomata	768	241	174	110	54
Ecdysozoa	1309	149	83	45	20

The queries corresponding to the two hypotheses have returned the numbers of patterns shown in the table. Five different thresholds have been used for bootstrap, from no threshold to values higher than 95%.

Figure 6 shows the tree topology of the Dos Reis *et al.* phylogeny in which the node level of support is given as the ratio of the number of families matching the pattern over the total number of eligible families (*i.e.*, the HOMOLENS families containing the species found in the subtree). As expected, both numbers decrease as we move from the leaves to the root of the tree due to the fact that deep nodes correspond to subtrees containing a growing number of taxa. A consequence is that the level of support also decreases when moving to the deepest parts of the tree. This goes to an extreme with the nodes corresponding to *Boreoeutheria*, *Eutheria* and *Theria* as they have a support equal to zero in terms of families matching the pattern. This is due to the fact that it is not possible to specify fuzzy patterns in which some species are optional for the search. This would allow to put constraint on the placement of a species in a tree, without rejecting patterns which do not include this species. As we are restricted to strict patterns, finding families containing a large number of species and exactly matching this pattern is difficult.

This method can also be used to test alternative hypotheses in a tree. For instances, the *Pegasoferae* node has the lowest support (168/6466) when looking at all the other nodes located at the same level in the tree. Moreover, Waddell *et al.* [22] made the hypothesis that *Chiroptera* is in fact a sister group of *Equus caballus*. To test this hypothesis we searched for the number of gene families supporting it. For that purpose, we used the pattern `((!//Equus caballus,(!//Pteropus vampyrus,!//Myotis lucifugus)!)!,((!//Canis lupus familiaris,!//Ailuropoda melanoleuca)!,!//Felis catus)!)`. With this pattern, we obtained a support level of 6/6466 for *Pegasoferae*, which is significantly lower than the one observed with the original topology.

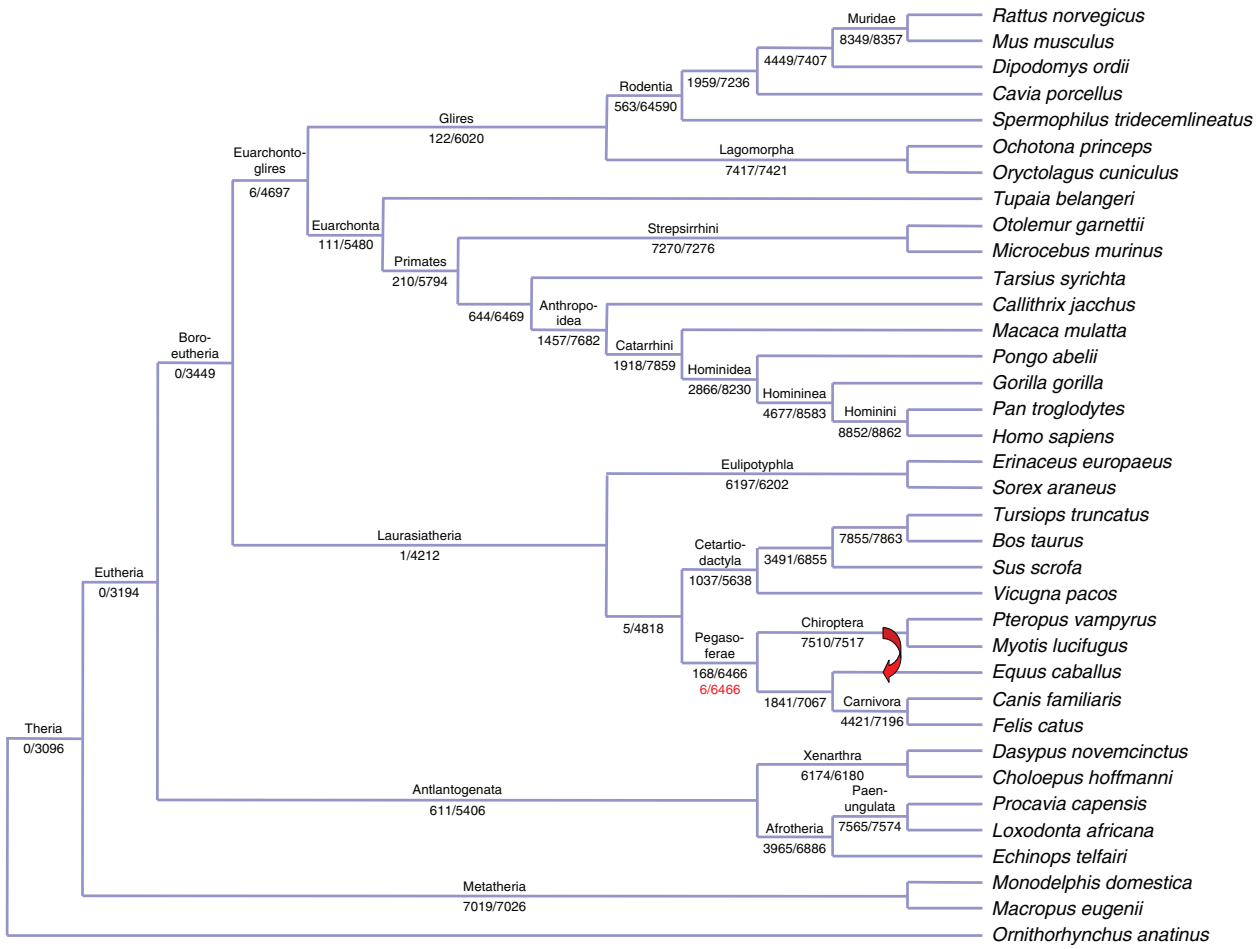


Figure 6 Mammal phylogeny support through the number of matching gene trees. For each node, the number of families matching the pattern over the number of eligible families is given. Red arrow shows the branch displacement required to obtain an alternative topology in which *Chiroptera* is a sister group of *E. caballus* and their father node is a sister group of *Carnivora*.

Discussion and conclusion

TPMS is a useful set of programs that can be used in a wide range of phylogenetic problems. It has high performance due to its implementation in a compiled language and the fact that all individual programs are multithreaded. Multithreading allows an optimal use on the multi-core architectures that are now common even for desktop computers. For instance, the two pattern searches (with no threshold for bootstrap values) performed on a collection containing 14190 trees to identify the ones supporting the Coelomata or the Ecdyzosoa hypotheses took only six seconds on an Intel Xeon 2.67 GHz with four cores.

For the tree pattern search, the most recent program available to perform this kind of task is PhyloPattern [23]. It is implemented in Java and Prolog and its users have to write their queries as Prolog predicates, which can be very difficult for non-experts. Also, this program cannot be used to root trees or to search for HGTs. Moreover, its performance will be far below the ones of TPMS, due to the fact it is written with interpreted languages.

For the automated tree rooting procedure, different alternatives to the two strategies proposed by TPMS are available. Due to its simplicity, the classical midpoint method looks like a good choice, but this is usually a poor solution. Indeed, midpoint rooting implies that the molecular clock hypothesis is approximately true for the sequences used and, most of the time, this is not the case [24]. Other common approaches are mostly based on algorithms identifying roots that minimize the number of gene duplication and losses, but they require reconciled trees and therefore they need a reliable reference species tree [1,25,26]. In the case of TPMS, the rooting strategy based on the use of an unicity score does not require the availability of such reference tree. This alternative is useful in the case of gene trees containing prokaryotic sequences, as no trusted species tree exist for these organisms. Also, reconciliation methods usually performs poorly with prokaryotic sequences as they do not introduce the possibility of HGTs events.

The application of TPMS for efficient large-scale identification of putative HGTs would require some improvements. Presently, the algorithm is already able to search for incongruent patterns in which a subtree is responsible for a topological shift from the reference species tree. The problem is that the automated procedure provided does not guarantee that the rooting is optimal for prokaryotic species when performing "generic" searches (*e.g.*, searches among the whole Bacteria or Archaea). If there is a lot of HGTs in a tree, our algorithm can lead to a wrong rooting, causing partially wrong taxonomic assignments, and therefore false positives. On the other hand, using TPMS with an explicit transfer pattern (a taxon nested in another when it is not supposed to be according to the species tree) can be very efficient. A method like Prunier, which is specifically devoted to the detection of HGTs is of course more efficient than TPMS in that area but the drawback is that it is limited to unicopy gene families [5].

In the future, we have planned to add functionalities mainly to increase the user-friendliness. For instance, a module automatically completing the species names when building the file containing the associations between sequence names and the species to which they belong is under development. This module works with the remote-acnuc library [27] and requires that the considered species are referenced in the general sequence collections.

Availability and requirements

Project name: TPMS
Project home page: http://pbil.univ-lyon1.fr/software/tpms/
Operating system(s): Unix-like operating systems such as Linux and MacOSX (Intel)
Progamming language: C++
Other requirements: Bio++ and Boost libraries
License: GNU GPL

Additional file

Additional file 1: Set of TPMS patterns corresponding to all possible subtrees from the tree shown in Figure 6.

Competing interests

The authors declare that they have no competing interests.

Authors' contributions

TB did all software development, documentation writing and web site set-up, FL participated to conception of the rooting algorithms, VD participated to the incongruencies detection research, GP supervised TPMS development and wrote the manuscript. All authors read and approved the final manuscript.

Acknowledgements

This work has been supported by ANR grant ANR-08-GENM-025 for the MetaSoil project and ANR grant ANR-10-BINF-01-01 for the Ancestrome project.

Received: 5 November 2012 Accepted: 12 March 2013
Published: 27 March 2013

References

1. Dufayard JF, Duret L, Penel S, Gouy M, Rechenmann F, Perrière G: **Tree pattern matching in phylogenetic trees: automatic search for orthologs or paralogs in homologous gene sequence databases.** *Bioinformatics* 2005, **21:**2596–2603.
2. Gabaldón T: **Large-scale assignment of orthology: back to phylogenetics?** *Genome Biol* 2008, **9:**235.
3. Altenhoff AM, Dessimoz C: **Phylogenetic and functional assessment of orthologs inference projects and methods.** *PLoS Comput Biol* 2009, **5:**e1000262.
4. Pryszcz LP, Huerta-Cepas J, Gabaldón T: **MetaPhOrs: orthology and paralogy predictions from multiple phylogenetic evidence using a consistency-based confidence score.** *Nucleic Acids Res* 2010, **39:**e32.

5. Abby SS, Tannier E, Gouy M, Daubin V: **Detecting lateral gene transfers by statistical reconciliation of phylogenetic forests.** *BMC Bioinformatics* 2010, **11:**324.
6. Studer RA, Penel S, Duret L, Robinson-Rechavi M: **Pervasive positive selection on duplicated and nonduplicated vertebrate protein coding genes.** *Genome Res* 2008, **18:**1393–1402.
7. Meunier J, Khelifi A, Navratil V, Duret L: **Homology-dependent methylation in primate repetitive DNA.** *Proc Natl Acad Sci USA* 2005, **102:**5471–5476.
8. Belle EM, Duret L, Galtier N, Eyre-Walker A: **The decline of isochores in mammals: an assessment of the GC content variation along the mammalian phylogeny.** *J Mol Evol* 2004, **58:**653–660.
9. Penel S, Arigon AM, Dufayard JF, Sertier AS, Daubin V, Duret L, Gouy M, Perrière G: **Databases of homologous gene families for comparative genomics.** *BMC Bioinformatics* 2009, **10**(Suppl 6):S3.
10. Dutheil J, Gaillard S, Bazin E, Glemin S, Ranwez V, Galtier N, Belkhir K: **Bio++: a set of C++ libraries for sequence analysis, phylogenetics, molecular evolution and population genetics.** *BMC Bioinformatics* 2006, **7:**188.
11. Dos Reis M, Inoue J, Hasegawa M, Asher RJ, Donoghue PCJ, Yang Z: **Phylogenomic datasets provide both precision and accuracy in estimating the timescale of placental mammal phylogeny.** *Proc R Soc B* 2012, **279:**3491–3500.
12. Manuel M, Kruse M, Muller WE, Le Parco Y: **The comparison of beta-thymosin homologues among metazoa supports an arthropod-nematode clade.** *J Mol Evol* 2000, **51:**378–381.
13. Ruiz-Trillo I, Paps J, Loukota M, Ribera C, Jondelius U, Baguna J, Riutort M: **A phylogenetic analysis of myosin heavy chain type II sequences corroborates that Acoela and Nemertodermatida are basal bilaterians.** *Proc Natl Acad Sci USA* 2002, **99:**11246–11251.
14. Mallatt J, Winchell CJ: **Testing the new animal phylogeny: first use of combined large-subunit and small-subunit rRNA gene sequences to classify the protostomes.** *Mol Biol Evol* 2002, **19:**289–301.
15. Blair JE, Ikeo K, Gojobori T, Hedges SB: **The evolutionary position of nematodes.** *BMC Evol Biol* 2002, **2:**7.
16. Wolf YI, Rogozin IB, Koonin EV: **Coelomata and not Ecdysozoa: evidence from genome-wide phylogenetic analysis.** *Genome Res* 2004, **14:**29–36.
17. Zheng J, Rogozin IB, Koonin EV, Przytycka TM: **Support for the Coelomata clade of animals from a rigorous analysis of the pattern of intron conservation.** *Mol Biol Evol* 2007, **24:**2583–2592.
18. Dopazo H, Dopazo J: **Genome-scale evidence of the nematode-arthropod clade.** *Genome Biol* 2005, **6:**R41.
19. Lartillot N, Philippe H: **Improvement of molecular phylogenetic inference and the phylogeny of Bilateria.** *Philos Trans R Soc B* 2008, **363:**1463–1472.
20. Holton TA, Pisani D: **Deep genomic-scale analyses of the metazoa reject Coelomata: evidence from single- and multigene families analyzed under a supertree and supermatrix paradigm.** *Genome Biol Evol* 2010, **2:**310–324.
21. Denver DR, Morris K, Lynch M, Thomas WK: **High mutation rate and predominance of insertions in the Caenorhabditis elegans nuclear genome.** *Nature* 2004, **430:**679–682.
22. Waddell PJ, Shelley S: **Evaluating placental inter-ordinal phylogenies with novel sequences including RAG1, γ-fibrinogen, ND6, and mt-tRNA, plus MCMC-driven nucleotide, amino acid, and codon models.** *Mol Phylogenet Evol* 2003, **28:**197–224.
23. Gouret P, Thompson JD, Pontarotti P: **PhyloPattern: regular expression to identify complex patterns in phylogenetic trees.** *BMC Bioinformatics* 2009, **10:**298.
24. Li WH: **So, what about the molecular clock hypothesis?** *Curr Opin Genet Dev* 1993, **3:**896–901.
25. Eulenstein O, Mirkin B, Vingron M: **Duplication-based measures of difference between gene and species trees.** *J Comput Biol* 1998, **5:**135–148.
26. Górecki P, Eulenstein O: **Algorithms: simultaneous error-correction and rooting for gene tree reconciliation and the gene duplication problem.** *BMC Bioinformatics* 2012, **13**(Suppl 10):S14.
27. Gouy M, Delmotte S: **Remote access to ACNUC nucleotide and protein sequence databases at PBIL.** *Biochimie* 2008, **90:**555–562.

doi:10.1186/1471-2105-14-109
Cite this article as: Bigot *et al.*: TPMS: a set of utilities for querying collections of gene trees. *BMC Bioinformatics* 2013 **14**:109.

A.2 Patterns of molecular evolution in dioecious and non-dioecious silene

Auteurs : Jos Kafer, Martina Talianová, Thomas Bigot, Elleni Michu, Laurent Guéguen, Alex Widmer, Sylvain Glémin et Gabriel AB Marais

Revue : Journal of Evolutionary Biology

Statut : Publié

URL : `http://onlinelibrary.wiley.com/doi/10.1111/jeb.12052/abstract`

Référence Bibliographique : [66]

Le contrat d'édition du journal *Journal of Evolutionary Biology* ne permet pas la libre diffusion des articles.

A.3 Bio++ : efficient, extensible libraries and tools for computational molecular evolution

Auteurs : Laurent Guéguen, Sylvain Gaillard, Bastien Boussau, Manolo Gouy, Mathieu Groussin, Nicolas Rochette, Thomas Bigot, David Fournier, Fanny Pouyet, Vincent Cahais, Aurélien Bernard, Céline Scornavacca, Benoît Nabholz, Annabelle Haudry, Loïc Dachary, Nicolas Galtier, Khalid Belkhir, Julien Dutheil.

Revue : Molecular Biology and Evolution

Statut : Publié

URL : `http://mbe.oxfordjournals.org/content/30/8/1745`

Référence Bibliographique : [49]

Le contrat d'édition du journal *Molecular Biology and Evolution* ne permet pas la libre diffusion des articles.