Université de Montréal

Algorithmes de construction et correction d'arbres de gènes par la réconciliation

par
Manuel Lafond

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Thèse présentée à la Faculté des études supérieures
en vue de l'obtention du grade de Philosophiæ Doctor (Ph.D.)
en informatique

Août, 2016

Université de Montréal
Faculté des études supérieures

Cette thèse intitulée:

Algorithmes de construction et correction d'arbres de gènes par la réconciliation

présentée par:

Manuel Lafond

a été évaluée par un jury composé des personnes suivantes:

Sylvie Hamel,	président-rapporteur
Nadia El-Mabrouk,	directeur de recherche
Pierre Bellec,	membre du jury
Yann Ponty,	examinateur externe

Thèse acceptée le: .....

RÉSUMÉ

Les gènes, qui servent à encoder les fonctions biologiques des êtres vivants, forment l'unité moléculaire de base de l'hérédité. Afin d'expliquer la diversité des espèces que l'on peut observer aujourd'hui, il est essentiel de comprendre comment les gènes évoluent. Pour ce faire, on doit recréer le passé en inférant leur phylogénie, c'est-à-dire un arbre de gènes qui représente les liens de parenté des régions codantes des vivants. Les méthodes classiques d'inférence phylogénétique ont été élaborées principalement pour construire des arbres d'espèces et ne se basent que sur les séquences d'ADN. Les gènes sont toutefois riches en information, et on commence à peine à voir apparaître des méthodes de reconstruction qui utilisent leurs propriétés spécifiques. Notamment, l'histoire d'une famille de gènes en terme de duplications et de pertes, obtenue par la réconciliation d'un arbre de gènes avec un arbre d'espèces, peut nous permettre de détecter des faiblesses au sein d'un arbre et de l'améliorer. Dans cette thèse, la réconciliation est appliquée à la construction et la correction d'arbres de gènes sous trois angles différents:

- Nous abordons la problématique de résoudre un arbre de gènes non-binaire. En particulier, nous présentons un algorithme en temps linéaire qui résout une polytomie en se basant sur la réconciliation.
- Nous proposons une nouvelle approche de correction d'arbres de gènes par les relations d'orthologie et paralogie. Des algorithmes en temps polynomial sont présentés pour les problèmes suivants: corriger un arbre de gènes afin qu'il contienne un ensemble d'orthologues donné, et valider un ensemble de relations partielles d'orthologie et paralogie.
- Nous montrons comment la réconciliation peut servir à "combiner" plusieurs arbres de gènes. Plus précisément, nous étudions le problème de choisir un superarbre de gènes selon son coût de réconciliation.

Mots clés: arbres de gènes, arbres d'espèces, réconciliation, polytomie, duplication, orthologie, paralogie, superarbre

ABSTRACT

Genes encode the biological functions of all living organisms and are the basic molecular units of heredity. In order to explain the diversity of species that can be observed today, it is essential to understand how genes evolve. To do this, the past has to be recreated by inferring their phylogeny, i.e. a gene tree depicting the parental relationships between the coding regions of living beings. Traditional phylogenetic inference methods have been developed primarily to construct species trees and are solely based on DNA sequences. Genes, however, are rich in information and only a few known reconstruction methods make usage of their specific properties. In particular, the history of a gene family in terms of duplications and losses, obtained by the reconciliation of a gene tree with a tree species, may allow us to detect weaknesses in a tree and improve it. In this thesis, reconciliation is applied to the construction and correction of gene trees from three different angles:

- We address the problem of resolving a non-binary gene tree. In particular, we present a linear time algorithm that solves a polytomy based on reconciliation.
- We propose a new gene tree correction approach based on orthology and paralogy relations. Polynomial-time algorithms are presented for the following problems: modify a gene tree so that it contains a given set of orthologous genes, and validate a set of partial orthology and paralogy relations.
- We show how reconciliation can be used to "combine" multiple gene trees. Specifically, we study the problem of choosing a gene supertree based on its reconciliation cost.

Keywords: gene trees, species trees, reconciliation, polytomy, duplication, orthology, paralogy, supertree

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES SIGLES

ADN	Acide désoxyribonucléique
ARN	Acide ribonucléique
BLAST	Basic Local Alignment Search Tool
DLS	Duplication-Loss-Speciation
LCA	Lowest Common Ancestor
MCL	Markov CLustering algorithm
MPR	Most Parsimonious Reconciliation
MRP	Matrix Representation with Parsimony
MRR	Meilleur Résultat Réciproque
NNI	Nearest Neighbor Interchange
NJ	Neighbor-Joining
RF	Robinson-Foulds
SPR	Subtree Prune and Regraft
TBR	Tree bisection and reconnection
UPGMA	Unweighted Pair Group Method with Arithmetic Mean
UTO	Unité Taxonomique Opérationnelle

REMERCIEMENTS

Je dois d'abord remercier ma directrice de recherche, Nadia El-Mabrouk, sans qui je ne serais peut-être même pas impliqué dans le milieu de la recherche. Elle a fortement suggéré et grandement facilité mon passage de la maîtrise au doctorat, alors que mon plan initial était de m'arrêter au deuxième cycle. Et surtout, elle a assuré un excellent suivi tout au long de mes études, faisant en sorte que le doctorat s'est déroulé sans la moindre embuche majeure.

Merci à tous les chercheurs avec lesquels j'ai eu la chance de collaborer et de qui j'ai pu apprendre énormément, en particulier Cédric Chauve, Riccardo Dondi, Aïda Ouangraoua et Éric Tannier. Un merci spécial à Krister M. Swenson, qui m'a initié aux principes de rigueur et d'éthique scientifique, ainsi qu'à la rédaction d'article. Merci aussi à Gena Hahn, qui m'a invité au sein de son groupe de travail en théorie des graphes malgré mes connaissances plutôt élémentaires sur le sujet.

Je tiens aussi à remercier chaleureusement tous ceux qui ont démontré, de près ou de loin, un intérêt envers mes travaux de recherche. Les études doctorales sont souvent remplies de doutes, et chaque article semble parfois n'être qu'une goutte dans un océan de papiers. Un témoignage d'appréciation est d'une valeur inestimable pour un étudiant, et j'encourage d'ailleurs tout chercheur qui lit ces lignes à ne jamais se gêner pour souligner le travail d'un étudiant, et ce même s'il ne le connaît pas. Merci particulièrement à David Sankoff et Céline Scornovacca à ce niveau.

Merci à Vickie, ma compagne de vie, pour sa remarquable participation à notre projet de gestion de progéniture. Sans elle, cette thèse n'existerait peut-être pas (c'est toutefois débattable, puisque sans elle, la progéniture n'existerait pas non plus - mais nous n'en ferons pas un cas ici). Merci à ma famille et à mes amis hors-science, qui me gardent sur terre et me rappellent régulièrement qu'il existe tout un univers au-delà des lemmes et théorèmes

que je côtoie chaque jour.

Je remercie aussi tous les membres du LBIT que j'ai côtoyés pendant les quatre dernières années: Emmanuel Noutahi, Robin Milosz, Billel Benzaid, Virginie Calderon, et Olivier Tremblay-Savard (le seul qui réussissait à m'empêcher de travailler en prenant un repas).

Merci à tous ceux qui ont lu cette thèse et apporté des suggestions, notamment les membres du jury. Un merci particulier à Yann Ponty pour ses recommandations qui ont, selon moi, permis d'améliorer la qualité de la thèse, et surtout de réduire le nombre de bêtises qui s'y trouvent.

Je remercie finalement tous les organismes qui m'ont supporté financièrement tout au long du doctorat: le Fonds de recherche du Québec - Nature et technologies (FRQNT), le Conseil de recherches en sciences naturelles et en génie du Canada (CRSNG), la Faculté des études supérieures et postdoctorales de l'Université de Montréal (FESP), le Département d'informatique et recherche opérationnelle (DIRO), et la société Hydro-Québec.

CHAPITRE 1

INTRODUCTION

On sait depuis Darwin que les espèces qui coexistent aujourd'hui partagent des relations de parenté qui peuvent être représentées par un *arbre d'espèces*. En théorie, toutes ces espèces descendent d'un ancêtre commun ayant vécu il y a environ 4 milliards d'années, et l'évolution des organismes peut être représentée par un arbre de la vie dans lequel cet ancêtre se trouve à la racine. Les *gènes*, qui servent à encoder les fonctions biologiques des êtres vivants et qui forment l'unité moléculaire d'hérédité, évoluent également au fil du temps. Les gènes peuvent subir des mutations locales au niveau de leur séquence d'ADN, ou bien des mutations à plus grande échelle telles que la duplication ou les réarrangements génomiques. De façon analogue aux espèces, ces événements lient les gènes par des relations de parenté évolutive que l'on représente par des *arbres de gènes*. Bien que les arbres d'espèces aient été étudiés depuis près de deux siècles, il n'y a qu'une quarantaine d'années que l'on s'intéresse aux arbres de gènes. En effet, leur rôle au niveau de l'évolution était sous-estimé, et ce jusqu'à ce que plusieurs ouvrages maintenant devenus classiques - on peut penser à *Evolution by gene duplication* de Ohno [120] ou à *The selfish gene* de Dawkins [40] - placent le gène, et non l'espèce, à l'avant-plan de l'origine de l'évolution et de la bio-diversité.

L'utilité pratique des phylogénies de gènes était toutefois limitée par les difficultés techniques liées à l'obtention de l'ADN des gènes. Mais depuis l'avènement des méthodes de séquençage des vingt dernières années, l'inférence d'arbres de gènes a été utilisée au sein d'une multitude d'études biologiques à plusieurs échelles. Dans les années 90, les reconstructions phylogénétiques de gènes d'une même famille ont permis d'approfondir notre compréhension des mécanismes régissant l'évolution des espèces, donnant lieu à des méthodes permettant de construire des arbres d'espèces plus

précis [109]. Dans les mêmes années, Eisein proposa d'utiliser les arbres de gènes afin d'améliorer notre capacité à prédire les fonctions encodées par les gènes en se basant sur le principe que la proximité évolutive de gènes pouvait être indicatrice de leur similarité fonctionnelle [49]. Plus récemment, les arbres de gènes inter-espèces ont été utilisés afin de développer des méthodes algorithmiques servant à inférer le contenu et la position des gènes dans les génomes des espèces ancestrales [9, 108]. À l'échelle intra-espèce, Cann, Stoneking et Wilson ont été en 1987 à l'origine de la théorie de l'existence d'une *Ève mitochondriale*, une femme considérée comme l'ancêtre commune la plus récente de tous les être humains ayant vécu en Afrique il y a entre 100,000 et 200,000 ans [22]. Cette découverte s'est faite en construisant des arbres de gènes mitochondriaux provenant d'une centaine d'humains d'ethnies différentes (en utilisant les techniques de parcimonie de Fitch, de Neighbor-Joining et de maximum de vraisemblance) et en datant les branches de l'arbre à l'aide de méthodes statistiques. C'est d'une façon similaire que l'on estima, en 2003, qu'une personne sur 200 descendait de l'empereur Genghis Kahn [172] (à l'exception qu'un *réseau* phylogénétique a été utilisé au lieu d'un arbre). Finalement, il est aussi possible de construire des arbres de gènes à l'échelle individuelle, c'est-à-dire pour des gènes provenant d'un même organisme, mais de cellules différentes. De multiples études (e.g. [21, 63]) ont ainsi permet d'obtenir les séquences d'ADN de multiples cellules cancéreuses à différents stades de la maladie d'un patient, ce qui permet de mieux comprendre l'évolution de la maladie.

L'importance de l'inférence de phylogénies de gènes en biologie n'est donc plus à démontrer, d'où la nécessité d'avoir des méthodes qui sont à la fois fiables et rapides. Toutefois, la reconstruction phylogénétique est un problème complexe du point de vue algorithmique et se ramène la plupart du temps à résoudre des problèmes qui ne peuvent être traités dans des temps raisonnables. Les méthodes développées sont donc des heuristiques qui ne permettent pas de garantir que les arbres obtenus soient les meilleurs

par rapport aux critères d'optimisation considérés. Les arbres obtenus contiennent donc généralement des parties erronées. De plus, ces arbres sont construits à partir de séquences d'ADN, alors que celles-ci peuvent être sujettes à toutes sortes d'artefacts menant à des erreurs, et ce même si l'on suppose que l'on a des algorithmes parfaits [123]. En effet, une mauvaise différentiation au niveau des séquences, par exemple des séquences trop similaires, ne permet parfois pas d'inférer une topologie précise. Inversement, des séquences trop éloignées peuvent devenir indistinguables de séquences choisies au hasard, ce qui ne permet pas de préférer une topologie à une autre. Des problèmes de modélisation peuvent également survenir lors du processus d'inférence phylogénétique. Les algorithmiques utilisés optimisent, pour la plupart, des critères combinatoires basés sur des hypothèses évolutives qui ne sont pas nécessairement vérifiées (par exemple que tous les sites d'une séquence évoluent indépendamment, ou bien que le principe de parcimonie ou de maximum de vraisemblance s'applique sur l'évolution). Ces méthodes nécessitent aussi de spécifier des paramètres, par exemple un modèle évolutif spécifiant des taux de mutation entre chaque paire de nucléotides, alors que ceux-ci sont parfois choisis arbitrairement, faute de ne pas savoir comment les déterminer adéquatement. D'autre part, nous ne sommes pas à l'abri d'erreurs au niveau du séquençage, de l'alignement, du regroupement de gènes en familles, de modèles évolutifs variables ou d'évolution convergente. Chacun de ces facteurs peut avoir un impact significatif sur les arbres de gènes inférés par les méthodes actuelles.

Les participants du projet TreeFam [83], une base de données d'arbres de gènes vérifiés manuellement, ont récemment fait état de la problématique sous-jacente aux algorithmes connus: aucun d'entre eux n'est assez fiable pour que l'on puisse se passer d'une étape de "curation" par un expert. Ceci entre en contraste avec le fait qu'énormément de progrès a été fait au niveau du séquençage depuis le *Human Genome Project* en 2001, et le nombre d'arbres de gènes que l'on veut inférer rend impossible la tâche de

la vérification manuelle. À titre d'exemple, la base de données Ensembl des vertébrés recense, pour 65 espèces, un total d'environ 1.2 millions de gènes pour lesquels 110,000 arbres ont été générés automatiquement (mais bien sûr, pas vérifiés manuellement) [61].

Malgré tous les problèmes associés aux séquences pour l'inférence phylogénétique, c'est souvent la seule source d'information qui est utilisée par les algorithmes actuels. Dans cette thèse, nous cherchons à raffiner ces méthodes, à les spécialiser aux arbres de gènes, en y incorporant des propriétés qui sont spécifiques aux gènes. Nous nous intéressons plus particulièrement à l'histoire des gènes en termes de spéciations, duplications et pertes. Un gène subit une spéciation lorsqu'il est transmit à deux espèces descendantes. La duplication correspond plutôt à la création d'un copie identique d'un gène au sein de son génome. La Figure 1.1 illustre les phénomènes de duplication et spéciation au sein d'un arbre de gènes. La duplication est un mécanisme évolutif important et une source majeure d'innovation fonctionnelle [120]. Les gènes peuvent également être perdus au cours de l'évolution. Ainsi, certains gènes peuvent être présents dans certaines espèces, et absents dans d'autres.

Savoir comment une famille de gènes a évolué par spéciations, duplications et pertes est primordial pour l'annotation fonctionnelle des gènes. Lorsque l'on dispose de la phylogénie des espèces contenant les gènes d'intérêt, il est possible d'en déduire une telle histoire évolutive pour la famille de gènes. Dans le cas le plus simple, les gènes évoluent uniquement par spéciation et la phylogénie de ces gènes devrait correspondre à celle de l'arbre d'espèces. Toutefois, la présence de pertes et duplications peut faire en sorte que la topologie de l'arbre de gènes diffère de celle de l'arbre d'espèces. Les deux arbres doivent alors être réconciliés afin de pouvoir expliquer leurs incompatibilités. On appelle donc une *réconciliation* entre l'arbre d'espèces et l'arbre de gènes une histoire par spéciations, duplications et pertes qui explique leurs différences. La Figure 1.1 illustre les notions d'arbre d'espèce, d'arbre de gènes et de réconciliation.

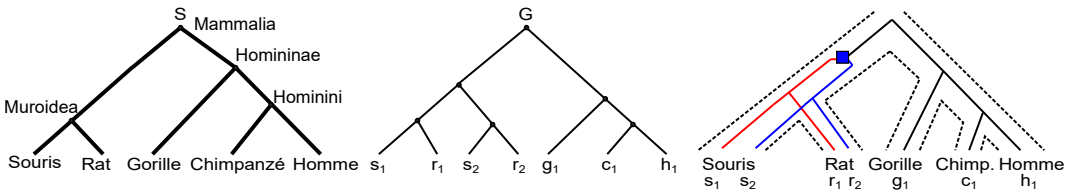

Figure 1.1: L'arbre S est un arbre d'espèces, illustrant les relations de proximité évolutive entre 5 mammifères. Les feuilles de l'arbre représentent des espèces existantes aujourd'hui et les noeuds internes des espèces ancestrales. L'arbre G est l'arbre du gène INS-2, participant à la production d'insuline. Les feuilles représentent des gènes contenus par les 5 espèces existantes, l'étiquette d'un gène correspondant à son espèce (i.e. s_1 est dans la souris, r_1 dans le rat, et ainsi de suite). L'image de droite représente une réconciliation, en "emboîtant" l'arbre de gènes G dans l'arbre d'espèces S (en pointillé). Chaque gène ancestral est ainsi associé à une espèce ancestrale. Une duplication, illustrée par un carré bleu sur l'arbre de gènes, est présente et explique pourquoi le rat et la souris ont deux copie du gène. Les deux lignées créées par la duplication sont représentées par des couleurs différentes. Les autres noeuds internes sont des spéciations.

Nous étudions la question de construction et correction d'arbres de gènes dans le contexte de la réconciliation sous trois aspects différents: la résolution de polytomies, les relations d'orthologie/paralogie et la construction de superarbres. Nous décrivons ci-dessous la structure générale de la thèse.

Chapitres 2 et 3: Mise en contexte

Le chapitre 2 introduit d'abord les notions biologiques essentielles à la compréhension de cette thèse. Nous présentons aussi les notations et définitions formelles qui seront utilisés à travers cette thèse, notamment le modèle DLS (*duplications-loss-speciation*) inhérent à la réconciliation. Nous décrivons également le *lca-mapping* permettant d'inférer une réconciliation parcimonieuse, c'est-à-dire, une réconciliation minimisant duplications et pertes.

Dans le chapitre 3, nous passons en revue les divers algorithmes liés à la correction et à la construction d'arbres de gènes sur lesquels se basent les résultats présentés dans cette thèse. Ceci passe par un bref survol des méthodes de regroupement de gènes en familles, d'inférence phylogénétique (générique et spécifique aux arbres de gènes) et de construction de superarbres. Nous décrivons également comment ces méthodes peuvent mener à des erreurs au sein des arbres de gènes, et présentons les quelques méthodes de correction qui ont été publiées à ce jour.

Les chapitres 4 à 7 présentent mes contributions originales à cette thèse, sous forme de quatre articles publiés. Ils sont regroupés selon les trois thèmes principaux de construction et correction qui suivent.

Chapitre 4 : Résolution de polytomies

Le chapitre 4 présente l'article "*An optimal reconciliation algorithm for gene trees with polytomies*" publié pour la conférence *WABI* en 2012 [101].

Les sources d'erreurs dans la construction d'arbres de gènes peuvent se traduire par la présence de *polytomies* au sein de l'arbre, c'est-à-dire de noeuds non-binaires. En effet, le résultat d'une mauvaise différentiation au niveau des séquences sera de produire des branches mal supportées au niveau de l'arbre. Des tests statistiques tels que le *bootstrapping* peuvent attribuer un degré de confiance à ces branches afin de détecter celles qui sont incertaines [56]. Elles peuvent ensuite être supprimées de l'arbre, ce qui se traduit par la contraction de branches, donnant ainsi lieu à des polytomies [163]. Par ailleurs, certains algorithmes d'inférence phylogénétique, notamment les méthodes Bayésiennes, infèrent une distribution sur les arbres. Une façon d'obtenir un seul arbre à partir d'une telle distribution est de "combiner" les arbres les plus probables. Ceci donne lieu à des parties non-résolues sur l'arbre lorsque les arbres à combiner reflètent des topologies contradictoires.

Bien que des duplications ou des spéciations simultanées donnant lieu à plus de deux descendants pour un gène donné existent dans l'évolution [148,

151], très peu de cas sont répertoriés. Il fait peu de doutes que la présence de noeuds non-binaires est plus souvent causée par des erreurs sous-jacentes aux méthodes de reconstruction utilisées telles que celles citées plus haut. Dans ce travail, nous partons donc de l'hypothèse que les polytomies sont des artefacts dus à une mauvaise résolution de l'arbre. Les arbres non-binaires doivent donc être résolus, c'est-à-dire "binarisés", et on peut se demander comment choisir une bonne résolution étant donné qu'il y en a un nombre exponentiel.

À ce jour, toutes les méthodes de résolution de polytomies spécifiques aux arbres de gènes se basent sur la réconciliation pour effectuer ce choix [23, 47, 96, 101, 176, 177]. Les duplications et pertes sont considérées comme des événements rares, et un choix naturel de résolution est celui qui minimise le nombre de duplications et pertes après réconciliation avec un arbre d'espèces donné. Dans le chapitre 4, nous montrons comment une polytomie peut être résolue selon ce critère en temps linéaire, alors que le meilleur algorithme existant pour résoudre une polytomie sous ce critère nécessitait un temps cubique [23].

Chapitres 5 et 6 : Relations d'orthologie et paralogie

Le chapitre 5 correspond à l'article "*Gene tree correction guided by orthology*" publié dans *BMC Bioinformatics* en 2013. Le chapitre 6 est quant à lui dédié à l'article "*Orthology and paralogy constraints: satisfiability and consistency*", paru dans *BMC Genomics* en 2014.

Les gènes partagent une relation qui est déterminée par l'événement qu'a subi leur ancêtre commun le plus récent. Plus spécifiquement, deux gènes sont *orthologues* si cet ancêtre a subi une spéciation, et *paralogues* s'il a plutôt subi une duplication. Depuis l'avènement du séquençage de génomes entiers, on cherche à distinguer les orthologues des paralogues, notamment à cause des implications fonctionnelles de ces relations [62]. En particulier, la "conjecture des orthologues" stipule que les gènes orthologues sont ceux

qui partagent des fonctions similaires au sein de l'organisme, tandis que les paralogues ont plutôt tendance à diverger [94, 160] d'un point de vue fonctionnel. Toutefois, cette conjecture n'est pas universellement reconnue [3, 29]. Quoi qu'il en soit, confirmer ou infirmer cette conjecture nécessite des outils algorithmiques permettant d'inférer correctement ces relations.

Dans cette thèse, nous établissons des liens formels entre les arbres de gènes et les relations d'orthologie et paralogie, et montrons comment ces deux notions peuvent s'entraider mutuellement dans le processus d'inférence. Dans un sens, si l'on dispose d'un arbre de gènes, alors la réconciliation avec un arbre d'espèces permet d'étiqueter les noeuds de l'arbre de gènes comme des noeuds de duplication ou de spéciation, et ainsi d'inférer un ensemble complet de relations d'orthologie/paralogie entre les gènes deux à deux. Cet ensemble de relations peut cependant entrer en contradiction avec des relations connues entre certains gènes, qui pourraient avoir été inférées par d'autres critères. En effet, il existe divers outils permettant d'inférer des relations d'orthologie et paralogie chez une famille donnée, et ce sans avoir accès à un arbre de gènes. La plupart sont basés sur la similarité des séquences des gènes (e.g. OrthoMCL [106], InParanoid [10], proteinortho [105], eggNOG [88]). D'autres utilisent la *synténie*, c'est-à-dire la conservation de blocs ordonnés de gènes [91, 103]. Bien que ces méthodes ne soient pas parfaites, on peut souvent identifier un sous-ensemble R des relations inférées en lesquelles on a confiance. On doit alors s'attendre à ce qu'un arbre de gènes pour la famille en question contienne les relations de R. Si ce n'est pas le cas, l'arbre doit être corrigé. Nous décrivons dans le chapitre 5 comment, en temps polynomial, modifier un arbre de façon minimale, selon la distance *Robinson-Foulds*, afin qu'il satisfasse un ensemble donné d'orthologies.

Inversement, supposons que l'on dispose d'un ensemble R de relations d'orthologie/paralogie. Si ces relations proviennent d'une vraie histoire de gènes, alors il devrait exister un arbre dont les duplications et spéciations reflètent R. Si un tel arbre n'existe pas, alors nous pouvons en déduire que

l'ensemble des relations est erroné, puisqu'il ne peut être contenu par aucune histoire évolutive de la famille de gènes. On dit que R est *satisfaisable* s'il existe un tel arbre de gènes, et *consistant* s'il existe un arbre d'espèces et un arbre de gènes dont la réconciliation donne lieu à R. Ceci peut s'avérer utile comme outil de validation pour les méthodes d'inférence de relations qui ne se basent pas sur un arbre de gènes réconcilié. Il existe des algorithmes pouvant vérifier la satisfaisabilité et la consistance d'un ensemble *complet* (i.e. lorsque la relation est connue pour chaque paire de gènes) de relations R en temps polynomial [80, 82], mais cette tâche est plus complexe lorsque R est *partiel*, i.e. certaines relations sont inconnues. Dans le chapitre 6, nous verrons que l'on peut, en temps polynomial, vérifier la satisfaisabilité et la consistence d'un tel ensemble.

Chapitre 7 : Superarbres de gènes

Dans le chapitre 7, nous présentons l'article "*Reconstructing a SuperGeneTree minimizing reconciliation*" qui a été publié dans *BMC Bioinformatics* en 2015 [99].

Le problème du *superarbre* est un problème classique en phylogénie. Étant donné un ensemble d'arbres $\mathcal{T}$ qui partagent ou non les mêmes étiquettes aux feuilles, on veut les combiner en trouvant un *superarbre*, c'est-à-dire un arbre qui "contient" chaque arbre de $\mathcal{T}$. En guise de motivation, les auteurs de [13] affirment que les études taxonomiques sont faites de façon non-coordonnée et opportuniste. Le résultat est une dispersion importante des données, un obstacle majeur à la construction de phylogénies d'espèces complètes. Certaines espèces sont largement étudiées et d'autres sont sous-représentées, créant des inégalités quant à nos connaissances sur le contenu en gènes des génomes. Il n'est pas clair que l'on puisse se fier aux méthodes d'inférence connues en présence de telles inégalités au sein des données, surtout pour une grande quantité d'espèces. Par contre, les données peuvent être suffisantes pour produire des phylogénies fiables lorsque l'on se limite à de plus petits groupes. Il

devient alors intéressant de pouvoir combiner les arbres produits sur chaque sous-groupe afin d'obtenir une phylogénie complète. C'est d'ailleurs ce qui a été fait en 2004 dans [39] pour la division taxonomique des *angiospermes*, les plantes à fleurs, où les auteurs on combiné des phylogénies issues de 46 études différentes. Les mêmes idées ont ensuite été appliquées aux mammifères et aux eukaryotes [14, 125].

Comme nous le verrons dans le chapitre 7, les mêmes motivations peuvent inspirer une démarche analogue sur les arbres de gènes. Il est possible que l'on veuille combiner plusieurs arbres de gènes en un seul. Même si on suppose que les arbres sont *compatibles*, i.e. qu'il existe un arbre qui les contient tous, cette tâche peut s'avérer difficile puisqu'il peut exister un nombre exponentiel de superarbres possibles. Comment choisir une bonne solution? Les algorithmes de construction de superarbres sont, pour la plupart, génériques et produisent soit une seule solution, soit toutes les solutions. Nous explorons cette problématique en établissant des critères de sélection spécifiques aux arbres de gènes par l'incorporation de l'arbre des espèces au sein de la construction de superarbres. Plus précisément, parmi tous les superarbres de gènes possibles, on cherche celui qui nécessite le moins de pertes et/ou duplications après réconciliation avec l'arbre d'espèces. On montre dans le chapitre 7 que ce problème est NP-complet et même difficile à approximer à un facteur $n^{1-\epsilon}$ près. On montre aussi que le problème reste NP-complet même si les arbres donnés en entrée ne partagent aucune étiquette (mais où on peut avoir plusieurs gènes de la même espèce). On propose deux algorithmes exacts exponentiels et une heuristique basée sur le problème du Max-Cut.

CHAPITRE 2

CONTEXTE BIOLOGIQUE ET INFORMATIQUE

Ce chapitre introduit les notions de base qui sont nécessaires à la compréhension de cette thèse. La section 2.1 est dédiée aux principaux concepts biologiques liés à l'étude des arbres de gènes. Nous clarifions d'abord ce que veulent dire pour nous un gène, un génome et une espèce. Nous discutons également de l'évolution des gènes par spéciation, duplication et perte, puis définissons les familles de gènes et les relations d'homologie, d'orthologie et de paralogie.

Dans la section 2.3, nous mettons en place le cadre formel dans lequel les arbres de gènes et d'espèces seront traités. Les notations et définitions relatives aux arbres et aux graphes y sont introduites. Nous définissons aussi les notions de polytomie et de résolution de polytomies et présentons les opérations NNI et Robinson-Foulds, qui mènent à des distances entre des arbres.

La section 2.4 est dédiée au modèle DLS (duplication-loss-speciation) et à la réconciliation entre arbres de gènes et arbres d'espèces permettant d'inférer les événements évolutifs ayant eu lieu dans l'histoire d'une famille de gènes.

2.1 Gènes, génomes et espèces

L'ADN est une molécule composée de séquences de nucléotides qui contient les instructions permettant aux être vivants de se développer et de se reproduire. Chacune de ces séquences est appelée un chromosome. On distingue quatre types de nucléotides désignés par A, C, G et T (respectivement Adénine, Cytosine, Guanine et Thymine). L'ADN est donc un ensemble de chaînes de caractères sur un alphabet à quatre lettres.

La notion courante de *gène*, que l'on adopte ici, désigne une région de

l'ADN servant à encoder une protéine, une macromolécule ayant un rôle biologique spécifique au sein de l'organisme. Cette région est transmise par descendance, et un gène forme donc une unité moléculaire d'hérédité.

Notons que cette définition n'inclut pas toutes les subtilités liées au fonctionnement des gènes. En fait, depuis le dernier siècle, la définition du mot "gène" évolue plus rapidement que les gènes eux-mêmes. Certaines définitions incluent les notions d'allèles différents (i.e. des "versions" du gène donnant lieu à des phénotypes différents, par exemple la couleur des yeux), de régions régulatrices et promotrices, de régions codantes et non-codantes appelées exons et introns, d'épissage, de gènes d'ARN, de spécialisation cellulaire, etc. Ici, nous faisons abstraction de toute cette complexité sémantique sur la définition d'un gène. Pour nous, un gène correspond simplement à une région spécifique de l'ADN. Il est toutefois intéressant de noter que, dans le contexte de cette thèse, la notion de gène pourrait être étendue à d'autres types d'unité moléculaire telles que les ARN non-codant (en particulier l'ARN de transfert ou l'ARN ribosomique). La propriété essentielle des "gènes" étudiés dans cette thèse est qu'ils peuvent être représentés par une séquence appartenant à une espèce donnée et qu'ils évoluent par spéciation, duplication et perte.

Pour en revenir à l'ADN, afin de former une protéine, la séquence d'ADN du gène est d'abord transcrite en une autre séquence de nucléotides appelée *ARN messager*, une copie transitoire de la séquence du gène[1]. L'ARN messager est divisé en sous-séquences de longueur 3 appelées des *codons*. Chaque codon est traduit en une molécule plus complexe appelée *acide aminé*, et c'est à partir de la séquence d'acides aminés obtenue que se forme une protéine. Un codon spécial appelé *codon stop* sert de signal d'arrêt de la traduction. Les 4 nucléotides donnent 64 codons possibles, alors qu'il n'y a que 20 acides aminés différents. On déduit donc qu'il y a des acides aminés qui sont encodés

[1]L'ARN messager est en fait le *complément* de la séquence du gène, chaque nucléotide ayant un complément, et remplace le nucléotide T par U. Nous n'utilisons toutefois pas ces notions dans cette thèse.

par plusieurs codons. Par exemple, la glutamine peut être formé soit par le codon CAA ou CAG. La séquence d'acides aminés peut servir d'alternative à l'ADN pour représenter le gène.

Chaque gène appartient à un *génome* donné, constitué de tout le matériel génétique d'un organisme. Le génome est habituellement défini comme l'ensemble de toutes les séquences d'ADN formant les chromosomes de l'organisme, qu'elles forment des régions génétiques ou non. Pour nos fins, nous nous intéressons uniquement à l'ensemble des gènes d'un génome, et ce dernier sera donc simplement représenté par des séquences ordonnées de gènes. Remarquons tout de même que cette définition fait abstraction du fait que les gènes ne sont pas répartis de façon homogène le long de leur génome. Certains gènes peuvent donc être plus rapprochés que d'autres au niveau séquentiel ou spatial, et cette proximité peut avoir un impact sur l'organisation du génome, ainsi que sur les niveaux d'expression et de régulation des gènes [30, 128].

Une *espèce* est un groupe d'organismes capables de se reproduire entre eux et d'engendrer une descendance. Nous allons faire la simplification "mathématique" suivante: on peut voir l'ensemble des espèces comme une partition des êtres vivants en classes d'équivalences. Ceci nous permet de désigner, pour chaque espèce, un seul individu pour représenter l'ensemble du groupe. Nous allons donc supposer que chaque espèce est représentée par un seul organisme, et donc un seul génome, et nous confondons espèces et génomes à travers cette thèse. Notons qu'à chaque espèce correspond alors un ensemble unique de gènes.

2.1.1 Évolution des gènes et des espèces

Pendant la transmission d'un gène d'un parent à son enfant, des erreurs de réplication peuvent survenir, modifiant ainsi sa séquence d'ADN. On distingue ici deux types de mutations, soit les *mutations ponctuelles* et les *changements structurels*.

Mutations ponctuelles

Les mutations ponctuelles n'affectent qu'un seul nucléotide de l'ADN d'un gène. Par exemple il se peut qu'un nucléotide A soit muté en G (ou vice-versa), ou qu'un C devienne un T (ou vice-versa). Ces mutations sont appelées des *transitions*, et toutes les autres mutations (e.g. $A \to T, C \to G$, ...) sont appelées *transversions*. Certains de ces changements sont plus fréquents que d'autres. Notamment, les transitions se produisent plus souvent, puisque la structure chimique du A est similaire à celle du G (et celle du C similaire à celle du T). Ces différences deviennent importantes lors de la construction d'une phylogénie, qui utilisent souvent un *modèle d'évolution*. Dans ces modèles, on suppose que les positions d'une séquence évoluent indépendamment les unes des autres, et une matrice de transition 4×4, chaque colonne/rangée représentant un nucléotide différent, spécifie avec quelle probabilité un nucléotide peut être substitué en un autre en un temps donné[2]. Par exemple, le modèle *Jukes-Cantor* [90] affecte la même probabilité μ à toutes les transitions d'un nucléotide N vers un autre nucléotide $N' \neq N$, tandis que le modèle *Kimura* [93] permet d'attribuer une probabilité différente aux transitions et aux transversions. Ces modèles peuvent alors servir à définir une notion de distance entre deux séquences basée sur la probabilité d'obtenir les différences observées, ou bien de calculer une probabilité d'observer un certain ensemble de séquences aux feuilles d'un arbre évolutif. Notons que la matrice de substitution est constituée des paramètres qui doivent être spécifiés par l'utilisateur, et il peut s'avérer difficile de les choisir adéquatement.

Certaines mutations sont *synonymes*, c'est-à-dire qu'elles n'affectent pas l'acide aminé formé par le codon auquel le nucléotide appartient. Par exemple, si une mutation locale change un codon CAA pour CAG, l'acide aminé

[2]En fait, pour des raisons techniques, un modèle est plutôt spécifié par ses *taux* de mutation, ce qui le libère de l'aspect temporel des substitutions. Nous ne nous avançons pas dans ces considérations ici.

correspondant demeure la glutamine, et la protéine n'est donc pas modifiée. Les mutations *non-synonymes* modifient quand à elles l'acide aminé traduit par le codon.

Il se peut aussi qu'un nucléotide soit supprimé (c'est-à-dire qu'il est "oublié" pendant la réplication), ou encore qu'un nouveau nucléotide soit inséré entre deux autres nucléotides.

L'accumulation de mutations ponctuelles peut modifier la protéine encodée par le gène, voire changer sa fonction. Il se peut aussi que la transcription de cette protéine devienne impossible, rendant ainsi le gène non-fonctionnel - on dira alors qu'il est *perdu*. La perte de gènes est généralement considérée comme rare. En effet, un tel événement est habituellement accompagné par la perte d'une fonction biologique. Si cette fonction était vitale, ou du moins assez importante, l'organisme qui en est privé aura du mal à se reproduire, mettant ainsi un frein à la transmission du génome désavantagé. Dans l'autre sens, une mutation génétique peut être bénéfique et améliorer une fonction biologique. Les organismes ayant subi de telles mutations peuvent alors se voir avantagés du point de vue de la survie ou bien de la reproduction, amenant un tel gène à se fixer dans la population.

Changements structurels

En plus des mutations ponctuelles qui n'affectent que les nucléotides individuels, l'ordre et le contenu des gènes d'un génome peut également évoluer. Des mutations appelées *réarrangements* peuvent affecter l'ordre des gènes. Certains de ces réarrangements peuvent être intra-chromosomiques, c'est-à-dire n'affecter qu'un seul chromosome. Par exemple, les inversions renversent l'ordre des gènes dans une sous-région du chromosome, et les transpositions déplacent de telles sous-régions d'un endroit à l'autre sur le chromosome. Les génomes peuvent aussi subir des réarrangements inter-chromosomiques. Les événements de translocation effectuent un échange réciproque de matériel génétique entre deux chromosomes distincts, tandis que les recombinaisons

échangent le matériel génétique de deux mêmes chromosomes, mais provenant d'individus différents.

D'autres événements évolutifs peuvent affecter le contenu en gènes des génomes, notamment les duplications et les pertes de gènes. Dans cette thèse, nous nous intéressons plus particulièrement à ce deuxième type d'événements, qui est à la base de l'évolution de familles de gènes. La *duplication* consiste en la création et la transmission de deux copies identiques d'une région d'ADN lors de la descendance. La duplication est reconnue depuis longtemps comme un des principaux mécanismes régissant l'évolution [120]. Après une duplication, un gène g engendre deux copies g_1 et g_2 chez l'enfant. On appelle g_1 et g_2 les *descendants directs* de g. La Figure 2.1.3 illustre ce phénomène.

Notons que les changements structurels peuvent aussi entraîner la perte d'un gène donné. Par exemple, il se peut qu'une région du génome contenant seulement la moitié d'un gène soit transposée, coupant ainsi le gène en deux et le rendant non-fonctionnel. Aussi, une duplication pourrait insérer la copie au milieu d'un autre gène sur le génome avec le même résultat.

Spéciation

Dans cette thèse, on distingue les événements de spéciation à deux niveaux: au niveau génomique pour les espèces et au niveau moléculaire pour les gènes.

Dans le premier cas, les espèces évoluent au fur et à mesure que leurs génomes subissent des mutations. Lorsqu'un grand nombre de changements est accumulé après plusieurs générations, on peut assister à la naissance de nouvelles espèces. La plupart de temps, celles-ci sont créées suite à l'apparition d'une barrière géographique qui sépare une population en deux sous-groupes. Les deux populations évoluent alors indépendamment et peuvent subir assez de modifications génétiques pour ne plus être interfécondes, menant ainsi à la création de nouvelles espèces. On dit alors que l'espèce parente ayant donné lieu à deux espèces enfants a subi un événement de

spéciation [3].

Une spéciation peut aussi affecter un gène, cette fois au niveau moléculaire. Lorsqu'une espèce s subit un événement de spéciation, chaque gène g de s est normalement transmis aux espèces descendantes s_1 et s_2, le but étant de passer les fonctions biologiques importantes de l'organisme à ses descendants. Le gène g devient alors le gène g_1 dans s_1 et g_2 dans s_2. On dit alors que g subit un événement de *spéciation*, et que g_1 et g_2 sont ses *descendants directs*. La Figure 2.1 illustre un événement de spéciation. Notons que g_1 et g_2 peuvent être différents au niveau de leur ADN - après tout ce sont des mutations génétiques qui ont créé de nouvelles espèces. Aussi, il est possible qu'il y ait "perte en cours de chemin", c'est-à-dire que l'un des deux sous-groupes ait perdu g mais pas l'autre - ce qui peut se produire lorsque la fonction biologique encodée par g n'est pas particulièrement importante, voire même nuisible pour l'une des espèces enfant suite à un changement d'environnement. Dans ce cas, g_1 ne se trouve pas dans le génome de l'espèce s_1, ou bien c'est g_2 qui n'est pas dans s_2. Voir la Figure 2.1 pour un exemple.

Le lien entre la spéciation au niveau génomique et au niveau moléculaire est assez direct: les gènes subissent une spéciation lorsque l'espèce à laquelle ils appartiennent subissent une spéciation.

Notons que la spéciation d'un gène peut cacher une duplication. Plus précisément, supposons qu'un gène g de l'espèce s soit dupliqué en les copies g_1 et g_2. Il est tout à fait possible que l'une des deux copies redondante et qu'elle soit éventuellement perdue. Il se peut alors que lors de la spéciation de s en s_1 et s_2, les deux espèces conservent une copie différente du gène. Par exemple s_1 conserve g_1 et perd g_2, et s_2 conserve g_2 et perd g_1. De tels phénomènes rendent difficile la tâche d'établir une correspondance entre les gènes des organismes.

[3]Il est aussi possible que des populations qui ne sont pas isolées évoluent en espèces distinctes. Par exemple, certains végétaux ont la capacité d'*hybridation*, i.e. deux plantes non-interfécondes peuvent parfois créer un enfant dont le génome est formé d'un hybride des deux génome. Nous n'allons pas explorer ces phénomènes dans cette thèse.

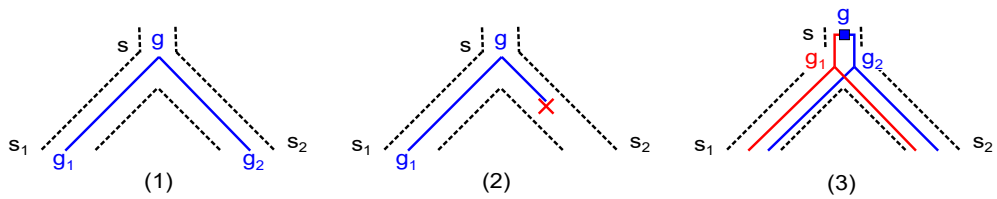

Figure 2.1: Illustration d'une spéciation, perte et duplication. En pointillé, l'arbre d'espèces illustrant une spéciation de l'espèce s en les deux espèces s_1 et s_2. L'arbre emboîté dans l'arbre d'espèces est l'arbre du gène g contenu dans s. En (1), g subit une spéciation et créé les gènes descendants g_1 dans s_1 et g_2 dans s_2. En (2), il y perte en cours de chemin vers s_2, et donc s_2 se retrouve sans copie de g. En (3), il y a duplication de g dans s. L'espèce s se retrouve donc avec deux copies g_1 et g_2 dans son génome. Ces deux copies subissent ensuite une spéciation, et ainsi s_1 et s_2 se retrouvent aussi avec deux copies.

2.1.2 Familles de gènes et relations

Une *famille* de gènes est un ensemble de gènes qui descendent tous d'un ancêtre commun. Deux gènes faisant partie de la même famille sont appelés *homologues*. Dans cette thèse, on suppose que la relation d'homologie forme une relation d'équivalence - en particulier, les familles forment une partition de l'ensemble des gènes. Les gènes homologues partagent une relation déterminée par leur ancêtre commun le plus récent. Deux gènes sont *orthologues* si cet ancêtre a subi une spéciation, et *paralogues* s'il a plutôt subi une duplication [60]. L'importance de distinguer ces deux types de relations est principalement liée à la *conjecture des orthologues* [94].

Conjecture des orthologues

La conjecture des orthologues stipule que les gènes orthologues ont tendance à être plus similaires en termes de séquence et fonction que les paralogues.

Prenons un gène ancestral g, et supposons qu'il effectue une fonction

vitale chez son espèce s. Si g subit un événement de spéciation, ses descendants directs g_1 et g_2 sont envoyés chez les espèces enfants à s. Si la fonction qu'effectuait g était vitale à s, cette fonction devrait aussi l'être pour ses descendants. On peut donc s'attendre à ce que g_1 et g_2 n'aient pas subi de mutations majeures pendant la spéciation. En étendant ce raisonnement à deux gènes g_1 et g_2 existant aujourd'hui et dont l'ancêtre commun est une spéciation, on s'attendrait à ce que g_1 et g_2 soient similaires en terme de séquence et de fonction.

Mais si g subit plutôt une duplication, l'organisme s se retrouve alors avec une copie supplémentaire g'. Normalement, s n'a besoin que d'une seule copie, disons g, pour combler la fonction vitale (il se pourrait très bien que g' prenne le relais pour effectuer cette fonction). L'autre copie g' est donc "libre" d'évoluer sans que ce ne soit léthal à l'espèce. On peut donc s'attendre à un taux de mutation plus élevé chez g', voire même à une modification de sa fonction.

Formation de nouvelles familles

La formation de nouvelles familles se produit généralement suite à une duplication de gène [74]. Il se peut qu'à la suite d'une duplication, un gène g faisant partie d'une famille F accumule des mutations au point de devenir méconnaissable par rapport aux autres membres de F. Si g n'est pas perdu et demeure effectivement un gène après ces mutations, il aura alors fondé sa propre famille dont les membres sont tous ses descendants.

La définition d'homologie est relativement vague à ce sujet, ce qui a d'ailleurs suscité quelques problèmes quant à son utilisation [60]. Dans l'exemple du dernier paragraphe, g était initialement issu d'une duplication d'un autre gène g' de F. Mais alors, on peut argumenter que g descend quand même du plus vieil ancêtre de F, même s'il a accumulé beaucoup de mutations. À quel moment peut-on alors parler de création d'une nouvelle famille? Est-ce qu'une différentiation suffisante au niveau des séquences peut justifier

un tel événement? Ou encore, est-ce que c'est le changement de fonction qui entraîne la création d'une famille? Bien qu'intéressantes, nous n'aborderons pas ces questions philosophiques ici. Plutôt, comme nous le verrons dans la section 3.1, ces familles sont typiquement inférées à partir des séquences d'ADN des gènes, en supposant que les gènes d'une même famille sont similaires. Donc à toutes fins pratiques, les familles sont simplement définies par notre capacité à distinguer une réelle similarité entre deux gènes d'une ressemblance due au hasard.

2.1.3 Synténie

Tel que mentionné plus haut, l'ordre des gènes peut être modifié au cours de l'évolution suite à des événements de réarrangement. Deux génomes ayant le même contenu génétique ne sont donc pas garantis d'avoir les mêmes gènes dans le même ordre. D'ailleurs, il y a généralement peu de conservation dans l'ordre des gènes chez des espèces distantes [95]. On dit qu'il y a *synténie* lorsqu'il y a une conservation d'ordre sur un ensemble de gènes. Un *bloc synténique* est une sous-séquence de gènes commune à deux génomes s_1 et s_2. Par exemple, si s_1 a les gènes (g_1, g_2, g_3, g_4) dans cet ordre sur un de ses chromosomes, que s_2 a les gènes (h_1, h_2, h_3, h_4) dans cet ordre, et que g_i et h_i sont homologues pour chaque i, alors les deux sous-séquences forment un bloc synténique. Il n'y a pas de taille précise pour décrire un bloc synténique - généralement trois gènes consécutifs suffisent.

Notons que cette définition de synténie ignore la *distance intergénique*, c'est-à-dire le nombre de nucléotides entre deux gènes adjacents sur le génome. Il y a un lien intime entre la synténie et la distance intergénique: si peu de nucléotides séparent deux gènes, il y a moins de chances qu'un changement structurel vienne "casser" l'adjacence entre les deux gènes. On rapporte d'ailleurs dans [126] que les gènes adjacents dans des blocs synténiques ont une distance intergénique plus petite que les gènes adjacents en moyenne.

2.2 Alignement de séquences

Les méthodes traditionnelles de contruction d'arbres nécessitent en entrée un ensemble de séquences *alignées*. Soit S_1, S_2 deux séquences de caractères sur l'alphabet $\mathcal{A}$ [4]. Un alignement entre S_1 et S_2 met en correspondance les positions de S_1 et de S_2 afin de connaître celles qui ont préservées, insérées ou supprimées. Il est plus facile de voir un alignement en ajoutant des *trous*, c'est-à-dire des caractères '-', aux positions de S_1 et S_2 qui n'ont pas de correspondance.

Un *alignement global* S'_1, S'_2 est donc une obtenu en insérant des trous à travers S_1 et S_2 de façon à ce que les deux séquences S'_1 et S'_2 résultantes soient de la même longueur n, et qu'il n'existe aucune position i telle que $S'_1[i] = `-'$ et $S'_2[i] = `-'$. On déduit que $n \leq |S_1| + |S_2|$, i.e. la somme des tailles des deux séquences. On cherche l'alignement qui minimise le *coût d'alignement* $c(S'_1, S'_2)$. On suppose que l'on connaît une fonction de coût $f : \mathcal{A}' \times \mathcal{A}' \mapsto \mathbb{R}$, où $\mathcal{A}' = \mathcal{A} \cup \{`-'\}$. Le coût à la position i est donné par $c_i(S'_1, S'_2) = f(S'_1[i], S'_2[i])$, où $S[i]$ dénote le caractère à la position i de S. Le coût total est donc $c(S'_1, S'_2) = \sum_{i=1}^{n} c_i(S'_1, S'_2)$. L'algorithme Needleman–Wunsch, un incontournable de la bioinformatique, permet de retrouver un alignement de coût minimum entre S_1 et S_2 en temps $O(|S_1||S_2|)$ par programmation dynamique [114]. Ce coût minimum est parfois interprété comme une *distance* entre S_1 et S_2. Notons aussi que ce coût peut être interprété comme un score en ajustant c adéquatement.

Alignement multiple : Si on a plutôt un ensemble de séquences $S = \{S_1, \ldots, S_k\}$ dont les tailles peuvent varier, un *alignement multiple* $S' = \{S'_1, \ldots, S'_k\}$ de S est obtenu en insérant des trous à travers les séquences de S, encore une fois de façon à ce qu'elles soient toutes de la même longueur n. Il y a plusieurs façons d'attribuer un coût à S', la plus commune étant le "sum-

[4] L'alphabet est habituellement dénoté Σ, mais nous réservons cette notation aux ensembles d'espèces

of-pairs" dans lequel $c(S') = \sum_{1 \leq i < j \leq n} c(S_i, S_j)$. Trouver un alignement multiple minimisant le sum-of-pairs sous les mesures classiques (e.g. distance d'édition, distance Hamming) est NP-complet [166]. Il existe tout de même une multitude de logiciels permettant de retrouver un alignement multiple de qualité (Clustal [104], T-Coffee [118], MUSCLE [48], HMMER [58], ...).

Alignement local : Étant donné deux séquences S_1 et S_2, on peut se demander quelles sous-chaînes de S_1 et S_2 s'alignent le mieux. Par exemple, S_1 et S_2 peuvent être des séquences de gènes différents, mais qui partagent des domaines (i.e. des sous-séquences encodant des tâches similaires) que l'on aimerait retrouver. À chaque sous-chaînes S'_1 de S_1 et S'_2 de S_2 correspond donc un score $sc(S'_1, S'_2)$. Ce score est typiquement donné par un alignement global entre S'_1 et S'_2 dans lequel les positions ayant le même caractère contribuent positivement au score, et les autres positions négativement. L'alignement local peut retrouver les sous-chaînes S'_1 et S'_2 telles que $sc(S'_1, S'_2)$ est maximisé. L'algorithme Smith-Waterman [149], qui est une variante de l'algorithme Needleman–Wunsch, permet d'accomplir une telle tâche en temps $O(|S_1||S_2|)$.

BLAST : L'algorithme BLAST [4], qui tient lieu de *Basic Local Alignment Search Tool*, est une heuristique permettant d'approximer l'alignement local. La complexité quadratique de Smith-Waterman rend parfois impossible la tâche de retrouver un gène à travers une base de données de plusieurs génomes, alors que BLAST est en mesure de le faire. L'algorithme procède par filtrage en sélectionnant des régions susceptibles d'être intéressantes, ce qui réduit l'espace de recherche et accélère l'alignement local. L'idée de la méthode est, étant donné une séquence S_1 à retrouver dans une séquence S_2 de grande taille, de d'abord décomposer S_1 en ses $|S_1| - k + 1$ sous-chaînes de taille fixe k et, pour chacune d'entre elles, retrouver les positions de S_2 où ces sous-chaînes apparaissent (peut-être à quelques erreurs près). BLAST

essaie ensuite d'aligner S_1 par extension aux positions trouvées. C'est-à-dire, à partir de l'occurrence S_2' trouvée dans S_2, on essaie d'étendre S_2' de un caractère (à gauche ou à droite) et de refaire un alignement avec S_1. Ces opérations d'extension continuent jusqu'à ce que l'ajout d'un caractère réduise le score d'alignement. Ceci permet d'attribuer un score à toutes les régions rapportées, et celles-ci sont ensuite triées. BLAST donne aussi, pour chaque alignement avec un score s, une valeur appelée *e-value*, qui décrit le nombre de sous-chaînes attendues (au sens statistique) avec score s dans une sous-chaîne aléatoire de taille $|S_2|$. On peut modéliser une chaîne aléatoire de plusieurs façons, la plus simple étant de faire apparaître, à chaque position, chaque caractère avec la même probabilité. On peut aussi attribuer une probabilité distincte à chaque nucléotide ou acide aminé, par exemple basée sur le fréquence du caractère dans le génome. Il est toutefois difficile de déterminer si une e-value donnée est significative ou non selon l'usage qu'on veut en faire. À titre d'exemple, la base de données HOGENOM [122] utilise le seuil de e-value de 10^{-4} pour décider si deux gènes peuvent être homologues ou non.

2.3 Phylogénies, arbres de gènes et arbres d'espèces

Nous introduisons dans cette section les notations et définitions préliminaires utilisées dans cette thèse, ainsi que certaines opérations permettant de passer d'un arbre à l'autre. Ces opérations donnent lieu à des notions de distances entre des arbres. Nous formalisons ensuite les arbres de gènes, d'espèces ainsi que l'association (que l'on appelle un *mapping*) qui les unit.

2.3.1 Notations et définitions

Soit G un graphe simple non-orienté. On désigne par $V(G)$ l'ensemble de ses sommets et par $E(G)$ l'ensemble de ses arêtes. Nous noterons simplement

par $|G|$ le nombre de sommets $|V(G)|$ de G. Le *degré* d'un sommet v est le nombre de voisins de v dans G (ou de façon équivalente, le nombre d'arêtes qui sont incidentes à v). Une arête joignant deux sommets x et y est dénotée xy. Un *arbre* T est un graphe connexe sans cycle (notons que les arêtes d'un arbre ne sont pas orientées). Les sommets d'un arbre sont parfois appelés des *noeuds*. On dit que T est *enraciné* s'il contient un sommet spécial $r(T)$ appelé la *racine*. Sauf indication contraire, tous les arbres considérés dans cette thèse sont enracinés. Dans ce cas, un noeud x est un *descendant* d'un noeud y si y est sur le chemin unique entre x et $r(T)$. On dit alors que y est un *ancêtre* de x. On peut écrire $y \geq x$ lorsque y est un ancêtre de x (et $y > x$ si $y \neq x$). Si $y > x$ et xy est une arête de T, alors x est un *enfant* de y. Une *feuille* de T est un noeud de degré 1. Les noeuds qui ne sont pas des feuilles sont des noeuds *internes*. On suppose que chaque feuille d'un arbre est étiquetée de façon unique. L'ensemble des feuilles de T est dénoté $\mathcal{L}(T)$. Si x est un noeud de T, le *sous-arbre enraciné en* x, dénoté par T_x, désigne l'arbre contenant x et tous ses descendants et dont la racine est x. Si $v \in V(T)$, le *clade de* v, dénoté $clade(v)$, est le sous-ensemble de feuilles $\mathcal{L}(T_v)$. L'ensemble des clades de T est $clades(T) = \{clade(v) : v \in V(T)\}$. Il est possible de démontrer qu'un arbre est entièrement défini par $clades(T)$. Deux arbres T_1 et T_2 sont dits *égaux* si $clades(T_1) = clades(T_2)$, et dans ce cas on écrit $T_1 = T_2$.

L'*ancêtre commun le plus récent* d'un ensemble de noeuds $X \subseteq V(T)$ est l'ancêtre de tous les noeuds de X qui le plus éloigné de la racine. Il est dénoté par $lca_T(X)$ (lca tient lieu de *lowest common ancestor*). On peut écrire $lca_T(x, y)$ lorsque $|X| = 2$. Par exemple, dans l'arbre T_2 de la Figure 2.2, $lca_{T_2}(\{a, b, c\}) = y$ et $lca_{T_2}(d, e) = z$. La notion de *lca* ne s'applique pas à des arbres non-enracinés.

On dit que T est *binaire* si chaque noeud interne a exactement 2 enfants (i.e. chaque noeud est de degré 3 ou 1, excepté $r(T)$ qui est de degré 2). Sauf indication contraire, on écrit x_1 et x_2 pour désigner les deux enfants d'un

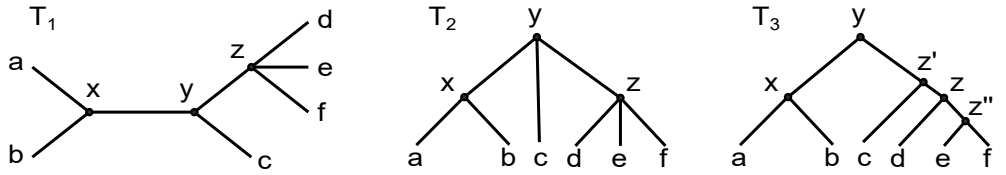

Figure 2.2: Illustrations des notions principales sur les arbres. Les trois arbres T_1, T_2 et T_3 ont le même ensemble de feuilles $\mathcal{L}(T_1) = \mathcal{L}(T_2) = \mathcal{L}(T_3) = \{a, b, c, d, e, f\}$. T_1 est un arbre non-enraciné et non-binaire. T_2 est un arbre enraciné et non-binaire. En fait, T_1 et T_2 sont identiques, à l'exception que l'on a désigné y comme étant la racine de T_2 et représenté l'arbre en conséquence. L'arbre T_3, quant à lui, est un arbre binaire enraciné. Outre les clades correspondant aux feuilles (e.g. $clade(a) = \{a\}$), les clades de T_2 sont $clade(x) = \{a, b\}, clade(z) = \{d, e, f\}$ et $clade(y) = \{a, b, c, d, e, f\}$, et donc $clades(T_2) = \{\{a\}, \{b\}, \{c\}, \{d\}, \{e\}, \{f\}, \{a, b\}, \{d, e, f\}, \{a, b, c, d, e, f\}\}$. On a $clades(T_2) \subset clades(T_3)$, et les seuls clades de $clades(T_3) \setminus clades(T_2)$ sont $\{e, f\}$ et $\{c, d, e, f\}$. L'arbre T_3 est donc une résolution de T_2.

noeud x. Si T est plutôt non-enraciné, on dit que T est binaire si chaque noeud est de degré 1 ou 3. Une *arbre étoile* T est un arbre enraciné qui a un seul noeud interne, soit $r(T)$, dont les enfants sont toutes les feuilles de T. La Figure 2.3 illustre les opérations de greffe et contraction.

Si T est un arbre (enraciné ou non), la *contraction* d'une arête xy consiste à supprimer y de T et à ajouter aux voisins de x les voisins de y (autres que x). Si T est enraciné, cette opération correspond à donner à x les enfants de y. La *greffe* d'une feuille ℓ à T sur l'arête xy correspond à subdiviser xy, créant ainsi un noeud z, puis à ajouter ℓ comme enfant de z. On peut aussi greffer un sous-arbre T' en remplaçant ℓ par $r(T')$ dans cette définition.

2.3.2 Polytomies et résolutions

Une *polytomie* est un arbre étoile ayant plus de 2 feuilles. On dit qu'un arbre G contient des polytomies s'il est non-binaire, i.e. si au moins un noeud v a 3 enfants ou plus. Dans ce cas, la *polytomie en* v est l'arbre étoile induit

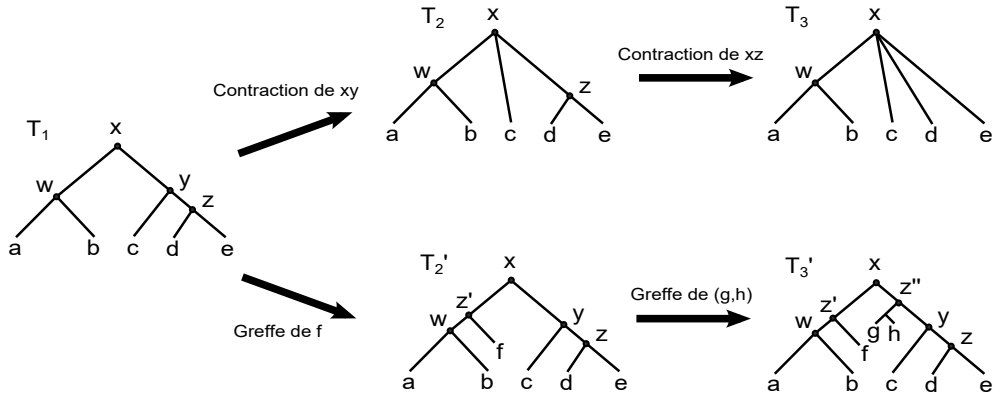

Figure 2.3: Illustrations des opérations de contraction et de greffe sur les arbres enracinés. Pour passer de T_1 à T_2, l'arête xy est contractée, et ainsi les enfants c et z de y deviennent les enfants de z. L'arête xz de T_2 est ensuite contractée pour passer à T_3, les enfants d, e de z étant donnés à x. En bas, le passage de T_1 à T_2' se fait par la greffe de la feuille f sur l'arête xw, créant ainsi un nouveau noeud interne z'. Pour passer de T_2' à T_3', le sous-arbre (g, h) est greffé à l'arête xy, créant un nouveau noeud interne z''.

par v et ses enfants. Une *résolution* de G (aussi appelée une *binarisation* ou encore un *raffinement binaire*) est un arbre binaire G' qui conserve les mêmes relations ancestrales que G. Plusieurs définitions équivalentes de résolution sont utilisées à travers la littérature (et à travers cette thèse). Les définitions principales sont énoncées ci-bas. Il n'est pas difficile (mais non-trivial) de montrer leur équivalence, exercice que nous laissons au lecteur.

Définition 2.1. *Soit G un arbre non-binaire. Un arbre binaire G' est une* résolution *de G si et seulement si G' satisfait l'un des trois énoncés suivants, qui sont tous équivalents:*

1. *(préservation des ancêtres) Une* résolution *de G est un arbre binaire G' tel que $\mathcal{L}(G) = \mathcal{L}(G')$ et pour tout $u, v \in V(G)$ avec $u > v$, $lca_{G'}(\mathcal{L}(G_u)) > lca_{G'}(\mathcal{L}(G_v))$.*

2. *(préservation des clades) Une* résolution *de G est un arbre binaire G'*

tel que $clades(G) \subseteq clades(G')$.

3. *(définition récursive) Si* G *n'a qu'un seul sommet, alors* $G' = G$ *est l'unique* résolution *de* G. *Sinon, soient* $v_1, \ldots, v_k$ *les enfants de* $r(G)$. *Alors une* résolution G' *de* G *est un arbre binaire dont les feuilles sont* $G_1, \ldots, G_k$, *où* G_i *est une résolution de* G_{v_i}, $1 \leq i \leq k$.

La Figure 2.2 illustre un exemple de résolution. L'arbre T_2 contient deux polytomies, soient la polytomie en y et la polytomie en z. Il contient trois clades non-triviaux définis par $clade(x) = \{a, b\}, clade(z) = \{d, e, f\}$ et $clade(y) = \mathcal{L}(T_2)$. T_3 est une résolution, ou une binarisation, de T_2. Il contient tous les clades de T_2, ainsi que deux clades supplémentaires créés par la résolution.

Si P est une polytomie à $n+1$ feuilles, il y a un nombre exponentiel de façons de résoudre P - ce nombre est en fait le nombre de Catalan $C_n = \frac{1}{n+1}\binom{2n}{n} = \prod_{k=2}^{n} \frac{n+k}{k}$, qui correspond au nombre d'arbres binaires sur $n+1$ feuilles. Dans le langage asymptotique, ceci donne $\Theta(\frac{4^n}{n^{3/2}})$ (ce qui peut s'obtenir par l'approximation de Stirling de $n!$). Nous discutons dans le chapitre 4 de l'origine des polytomies dans les arbres de gènes, et des critères permettant de choisir une "bonne" résolution, notamment celle qui donne la meilleure réconciliation.

2.3.3 Opérations et distances sur les arbres

Nous présentons ici les opérations NNI et RF, qui donnent lieu à une distance entre deux arbres. La distance NNI a d'abord été introduite en 1971 afin d'évaluer si deux arbres sur le même ensemble de feuilles, binaires et non-enracinés, sont significativement plus similaires que deux arbres choisis aléatoirement [134]. Quant à la distance RF, elle a été développée en 1981 dans un effort pour généraliser la comparaison aux paires d'arbres pouvant être enracinés ou non, binaires ou non (en fait, le nombre de noeuds internes des deux arbres n'a pas à être égal), et dans lesquels les noeuds peuvent avoir

plus d'une étiquette (incluant les noeuds internes) [133]. Dans le cadre de cette thèse, l'opération NNI est surtout utilisée en tant que manipulation élémentaire permettant de passer d'un arbre à un autre, par exemple dans les algorithmes d'exploration d'espace d'arbres. Quant à la distance RF, elle est utilisée en tant que mesure de similarité entre deux arbres, par exemple lorsque l'on a un arbre de référence et on veut savoir si notre arbre lui ressemble ou non.

Nearest Neighbor Interchange (NNI) [112, 134]: soit T un arbre binaire non-enraciné et uv une arête dont u et v ne sont pas des feuilles. Alors u a deux voisins u_1, u_2 distincts de v, et v a deux voisins v_1, v_2 distincts de u. Il y a deux façons d'interchanger une paire de ces quatre sommets afin d'obtenir un arbre différent. Par exemple, interchanger u_1 et v_1 correspond à remplacer l'arête u_1u par u_1v et l'arête v_1v par v_1u. On peut aussi interchanger u_1 et v_2 - les autres interchangements donnent un arbre équivalent lorsqu'on ne tient compte que des étiquettes aux feuilles (voir la Figure 2.4 pour un exemple). Cet interchangement est ce qu'on appelle une opération NNI. Si T est plutôt enraciné, une opération NNI s'effectue sur une arête uv où u a deux enfants v et u_1, et v a deux enfants v_1 et v_2. Les interchangements possibles sont alors u_1 avec v_1 ou encore u_1 avec v_2.

La *distance NNI* entre deux arbre binaires T_1 et T_2, enracinés ou non, est le nombre minimum d'opérations NNI à effectuer sur T_1 afin d'obtenir T_2. Trouver la distance NNI entre deux arbres est un problème NP-complet [89].

Robinson-Foulds (RF) [133]: une opération RF peut être définie sur un arbre non-binaire, enraciné ou non, dont les feuilles sont étiquetées. Une opération peut être la contraction d'une arête *interne* (i.e. une arête dont aucun des bouts n'est une feuille)[5], ou bien l'expansion d'une arête. La contraction d'une arête est telle que définie dans la section 2.3, i.e. contracter

[5]L'article original de Robinson et Foulds permettait la contraction des arêtes non-internes et introduisait la notion de *contraction étiquetée*. Mais il a été démontré plus tard que la contraction de ces arêtes n'est jamais nécessaire pour minimiser la distance RF.

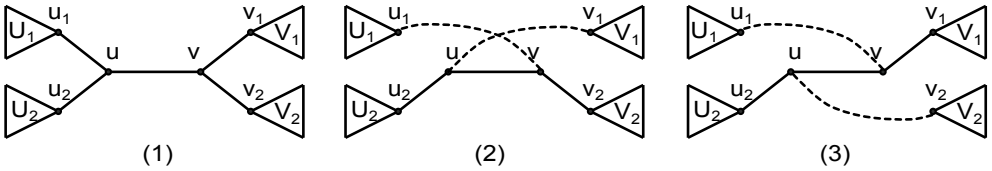

Figure 2.4: Illustrations des opérations NNI sur un arbre non-enraciné. (1) Un arbre non-enraciné et une arête uv, avec u et v des noeuds internes. (2) L'arbre obtenu en interchangeant u_1 et v_1. (3) L'arbre obtenu en interchangeant u_1 et v_2.

une arête uv d'un arbre T correspond à supprimer v de T et à donner à u les voisins de v. L'expansion d'une arête uv consiste à subdiviser uv, créant ainsi un nouveau noeud z, puis à donner à z soit un sous-ensemble des voisins de u, soit un sous-ensemble des voisins de v. Par exemple, sur la Figure 2.5, l'arête de T_1 qui lie la racine et son fils gauche (en rouge) est contractée, donnant lieu à l'arbre T'. Pour passer de T'' à T''', l'arête de T'' liant la racine et son fils droit (en bleu) subit une expansion: un nouveau noeud est créé sur l'arête, et le fils b de la racine est "donné" à ce nouveau noeud.

La *distance RF* entre deux arbres T_1 et T_2 est le minimum d'opérations de contraction et d'expansion nécessaires à effectuer sur T_1 afin d'obtenir T_2.

Bien qu'amusantes, les opérations RF sont très peu utilisées en pratique, et c'est souvent la distance qui nous intéresse, particulièrement à cause de ses propriétés théoriques intéressantes [41]. Soit xy une arête d'un arbre T. Si on retire xy de T, on déconnecte T et on obtient deux arbres A et B. L'arête définit donc une bipartition $\{\mathcal{L}(A), \mathcal{L}(B)\}$ que l'on appelle le *split* de xy. L'ensemble des *splits* de T est l'ensemble des $|T| - 1$ splits de T. Les propriétés des splits ont d'abord été introduites dans [19]. Il s'avère que la distance RF est égale au nombre de *splits* présents dans T_1 mais pas dans T_2, plus le nombre de splits dans T_2 mais pas dans T_1. Si T_1 et T_2 sont deux arbres binaires enracinés, cette distance est égale à $2 \cdot |clades(T_1) \setminus clades(T_2)|$, i.e. deux fois le nombre de clades présents dans

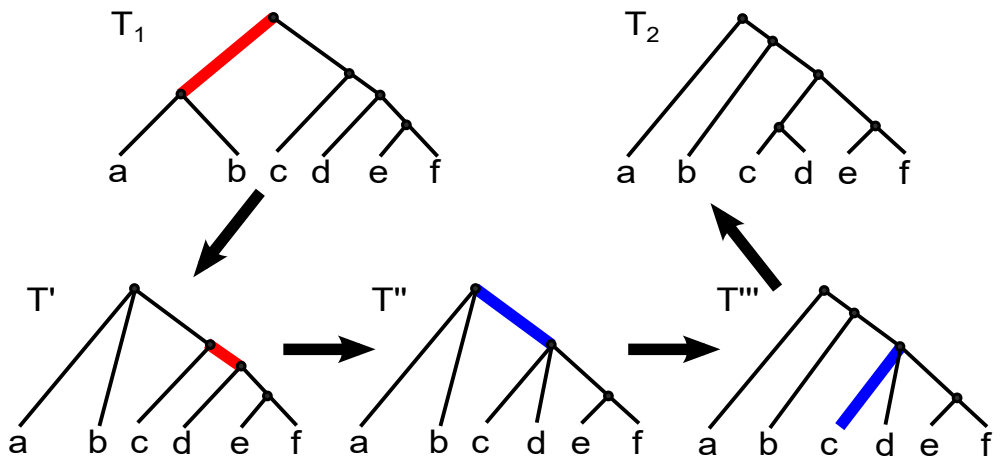

Figure 2.5: Illustrations opérations RF sur un arbre enraciné. T_1 et T_2 sont deux arbres donnés, et T', T'' et T'' illustrent une séquence ayant un minimum d'opérations pour passer de T_1 à T_2: deux contractions (les arêtes rouges indiquent l'arête qui est contractée à l'étape subséquente) et deux expansions (sur les arêtes bleues).

T_1 mais pas dans T_2. Cette propriété est une conséquence du fait qu'il y a une bijection entre les splits de T et $clades(T) \setminus \mathcal{L}(T)$. Sur la Figure 2.5, $clades(T_1) \setminus clades(T_2) = \{\{a, b, \}, \{d, e, f\}\}$ contient deux clades, et donc la distance RF entre les deux arbres est de 4 - ce qui correspond au nombre de contractions et d'expansions nécessaires pour passer de T_1 à T_2. Ces propriétés permettent de calculer la distance RF en temps linéaire. Malgré ses défauts - déplacer une seule feuille d'un arbre peut changer complètement sa distance - la distance RF est la plus utilisée étant donné la vitesse à laquelle elle peut être calculée.

Il existe d'autres opérations dignes de mention auxquelles on peut rattacher une distance. Par exemple, l'opération *Subtree prune and regraft* (SPR) consiste à enlever un sous-arbre et à le greffer sur une autre branche [77, 78]. Cette opération a été utilisée dans le contexte des modèles évolutifs permettant la recombinaison (le mixage de matériel génétique de

parents, ce qui peut donner lieu à un enfant qui possède des traits qui ne sont dans aucun des parents). L'opération *tree bisection and reconnection* (TBR), quant à elle, supprime une arête créant ainsi deux arbres. Ceux-ci sont rejoint en ajoutant un sommet sur une arête quelconque à chacun des deux arbres, et on ajoute une arête joignant ces deux sommets. L'opération TBR est surtout utilisée pour énumérer le voisinage d'un arbre de façon exhaustive, puisque le nombre d'opérations TBR possibles ($O(n^3)$) sur un arbre est plus élevé que le nombre de mouvements NNI ($O(n)$) ou SPR ($O(n^2)$) possibles [156].

2.3.4 Arbres de gènes, arbres d'espèces et lca-mapping

D'un point de vue formel, une *famille de gènes homologues* $\Gamma = \{g_1, \ldots, g_n\}$ (ou simplement une famille de gènes) est un ensemble de gènes (où les g_i peuvent être représentés par leur nom, leur séquence, etc.). Chaque gène $g \in \Gamma$ appartient à une espèce donnée $\sigma(g)$, et l'ensemble des espèces ayant un gène dans Γ est dénoté par Σ. À travers cette thèse, on suppose que $\sigma(g)$ est connu pour tout gène g de Γ.

Un arbre G est un *arbre de gènes* pour Γ si $\mathcal{L}(G) = \Gamma$, et un arbre S est un *arbre d'espèces* pour Σ si $\mathcal{L}(S) = \Sigma$. Les gènes représentés par les feuilles de G sont appelés les gènes *existants* et ceux représentés par les noeud internes les gènes *ancestraux*. Dans cette thèse, une *phylogénie* désigne un arbre de gènes ou un arbre d'espèces. Sauf indication contraire, toutes les phylogénies sont binaires. D'un point de vue biologique, si G est une phylogénie, chaque gène ancestral g a deux enfants g_1 et g_2 qui représentent ses deux descendants directs suivant un événement de spéciation ou duplication. D'une façon similaire, si S est une phylogénie, les deux enfants s_1 et s_2 d'une espèce ancestrale s représentent les espèces engendrées par un événement de spéciation ayant affecté s. On suppose dans le reste de cette section que G et S sont des phylogénies (donc binaires).

LCA-mapping

Puisque l'on suppose que σ est connu, on peut associer chaque gène existant $g \in \mathcal{L}(G)$ à son espèce $\sigma(g)$, qui est une feuille de S. Par contre, il nous est souvent nécessaire d'associer aussi chaque gène ancestral à une espèce dans S. Cette information nous est inconnue, et elle doit être inférée. On écrit $s(g)$ pour désigner l'espèce associée à un noeud g. On impose à $s(g)$ deux conditions:

1. si g est une feuille de G, alors $s(g) = \sigma(g)$;
2. si $g \geq g'$, alors $s(g) \geq s(g')$.

La deuxième condition nous assure que les contraintes temporelles représentées par G et S sont satisfaites, i.e. qu'un descendant de g ne se retrouve pas dans une espèce ayant existé avant l'apparition de g. Dénotons $\Sigma(g) = \{\sigma(g') : g' \in \mathcal{L}(G_g)\}$ l'ensemble des espèces dans lesquelles les descendants de g se retrouvent. Alors une conséquence immédiate de nos contraintes sur s est que $s(g) \geq lca_S(\Sigma(g))$, puisque $s(g)$ doit être un ancêtre de chaque espèce présente dans $\Sigma(g)$. Nous nous intéressons au cas particulier du *lca-mapping*, dans lequel cette inégalité devient une égalité.

Définition 2.2. *Le* lca-mapping $s : V(G) \to V(S)$ *associe chaque gène existant et ancestral de* G *à une espèce de* S *de la façon suivante:*

1. *si* $g \in \mathcal{L}(G)$, *alors* $s(g) = \sigma(g)$;
2. *sinon,* $s(g) = lca_S(\Sigma(g))$

Bien sûr, rien ne nous garantit que le lca-mapping identifie correctement l'espèce contenant chaque gène ancestral. Cette association peut être vue comme *parcimonieuse*, dans le sens où elle choisit pour le gène g l'espèce la plus récente qui pourrait le contenir (selon nos contraintes). Un des avantages principaux du lca-mapping est qu'il nous permet d'inférer une réconciliation

minimale, c'est-à-dire une histoire de famille de gènes qui nécessite un minimum d'explications en termes de pertes et de duplications de gènes. Il faut tout de même noter que dans les modèles où les transferts horizontaux de gènes sont permis, le lca-mapping n'est plus parcimonieux. En effet, il est possible que pour un gène ancestral g d'une espèce s_1, une copie g' soit transmise à une autre espèce s_2 ayant co-existé à la même époque (i.e. de façon "horizontale", c'est-à-dire que s_2 n'est pas un descendant de s_1). Or, dans l'arbre de gènes on aura que g est le parent de g', alors que $s(g) = s_1$ et $s(g') = s_2$, violant ainsi la deuxième condition d'un mapping. D'autres complexités surviennent en présence de transferts, notamment les "transferts des morts". Nous ne traitons pas de ces cas ici, et référons le lecteur à [158] pour plus de détails.

Il est aussi important de mentionner que même sans présence de transferts, l'utilisation du lca-mapping a été remise en question à quelques reprises, notamment dans le contexte de *réconciliation probabiliste* [45, 110]. Certains modèles permettent d'ailleurs d'associer un gène à une arête s_1s_2 de S (plutôt qu'à un noeud), ce qui peut représenter soit l'existence d'une espèce s entre s_1 et s_2 dont nous ignorions l'existence, ou bien l'apparition du gène pendant la transition de s_1 vers s_2.

2.4 Modèle DLS et réconciliation

Nous utilisons le modèle DLS (*Duplication, Loss, Speciation*) dans lequel chaque gène ancestral g subit exactement un de ces trois événements[6]:

- la *duplication* se produit lorsque g créé une copie g' de lui-même dans le génome de son espèce s, et chaque copie évolue indépendamment;
- la *spéciation* se produit lorsque l'espèce s contenant g se divise en deux

[6]On suppose ici que l'histoire d'une famille de gènes est régie par ces trois événements. D'autres types d'événements tels que le transfert horizontal et l'hybridation peuvent survenir, mais nous n'en tenons pas compte dans cette thèse.

espèces s_1 et s_2, et que le gène g transmet une copie g_1 à s_1 et une copie g_2 à s_2;

- la *perte* de g se produit généralement par une accumulation de mutations qui le rendent non-fonctionnel.

La *réconciliation*, établie depuis 1979 [67], vise à identifier les gènes ancestraux d'un arbre de gènes G ayant subi une duplication ou une spéciation, et d'identifier les gènes ancestraux qui ont été perdus. On peut trouver dans la littérature (et dans cette thèse) une multitude de définitions équivalentes de la réconciliation. Nous donnons ici celle qui, selon l'auteur, est la plus simple à concevoir.

Avant de procéder, il est à noter que l'on introduit dans cette section la notion de *consistance* d'une spéciation. Une autre définition de 'consistance' est donnée au chapitre 3, celle-ci applicable à un ensemble d'arbres. Qui plus est, on définit un ensemble de relations 'consistantes' au chapitre 6. L'utilisation de même terme pour désigner trois concepts différents est certes regrettable, mais c'est la terminologie qui a été utilisée dans les articles de cette thèse, et ceux-ci sont présentés tels que publiés.

Duplication et spéciation

Un *étiquetage* de G est une fonction $ev : V(G) \setminus \mathcal{L}(G) \rightarrow \{Duplication, Speciation\}$. Si ev est donné, on dit que g de G est une spéciation (respectivement, une duplication) pour signifier si $ev(g) = Speciation$ (respectivement, $ev(g) = Duplication$).

Les étiquetages ne sont pas tous valides par rapport à S. En effet, lorsqu'un gène ancestral g contenu dans l'espèce $s = s(g)$ subit une spéciation, ses enfants g_1 et g_2 sont, respectivement, envoyés dans les espèces s_1 et s_2. On doit donc s'attendre à ce que tout gène descendant de g_1 soit dans une espèce descendante de s_1, et tout gène descendant de s_2 soit dans

dans une espèce descendante de s_2. Cette condition détermine la *consistance avec* S telle que définie ci-bas:

Définition 2.3. *Soient S un arbre d'espèces et G un arbre de gènes étiqueté. Un sommet de spéciation g avec $s = s(g)$ est dit* consistant avec S *si $s(g_1) \leq s_1$ et $s(g_2) \leq s_2$.*

On dit que G est consistant *avec S si tous ses sommets de spéciation sont consistants avec S.*

Notons qu'il n'y a pas de contraintes quant à l'étiquetage d'un noeud par *Duplication*. Un exemple d'étiquetage de G consistant S est donné à la Figure 2.6.

Une question naturelle survient alors: étant donné un arbre d'espèce S et un arbre de gènes G, quel est l'étiquetage de G qui est consistant avec S et qui minimise le nombre de duplications? Il s'avère que la réponse passe par le *lca*-mapping.

Théorème 2.4. *[69] Un étiquetage de G qui minimise le nombre de duplications est obtenu par le lca-mapping $s(g) = lca_S(\Sigma(g))$ pour tout sommet interne g de G, et l'étiquetage $ev(g) = Duplication$ si $s(g) = s(g_1)$ ou $s(g) = s(g_2)$, et $ev(g) = Speciation$ sinon.*

L'arbre G' dans la Figure 2.6 est étiqueté de façon à minimiser le nombre de duplications. Notons que le lca-mapping n'est pas nécessairement la seule association gène-espèce permettant de minimiser les duplications. Mais comme nous le verrons, il devient nécessaire lorsque l'on veut aussi minimiser les pertes.

Perte de gènes

L'étiquetage ne suffit pas à expliquer complètement l'histoire d'une famille de gènes. Reprenons la Figure 2.6 en exemple. L'espèce ancestrale y de S a comme enfants x et f. Si g est un gène de y ayant subit une spéciation,

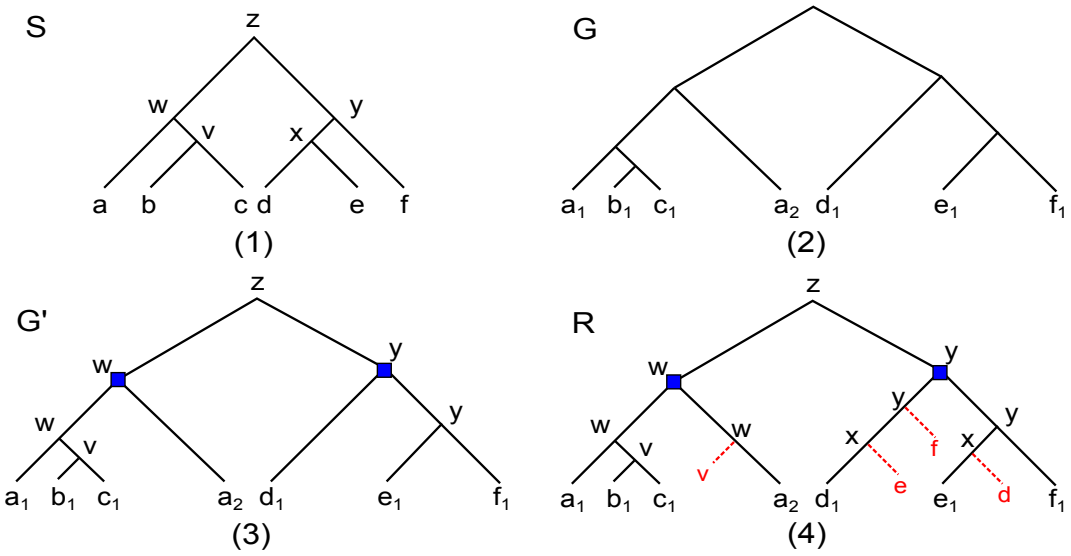

Figure 2.6: (1) Un arbre d'espèces S. (2) Un arbre de gènes G. Le nom d'un gène à une feuille correspond à l'espèce contenant le gène (e.g. $s(a_1) = a$). (3) G' est l'arbre G dont les noeuds internes sont étiquetés par Duplication et Speciation. Chaque noeud ancestral de G' est annoté par son lca-mapping. Les carrés bleus indiquent une duplication, et les autres noeuds internes sont des spéciations. Cet étiquetage est consistant avec S, et changer l'une des duplications en spéciation produirait un noeud inconsistant. (4) R est la réconciliation entre G et S minimisant pertes et duplications. Les pertes sont indiquées par les lignes pointillées rouges.

on s'attend donc à ce que g, dans son arbre de gènes, ait un enfant dans l'espèce x et un autre dans f. Ce n'est toutefois pas le cas dans G', où il semble manquer un gène de l'espèce x entre e_1 et son parent. Informellement, un gène g de y se serait transmit à e en "sautant" l'espèce x. Ceci peut s'expliquer par une perte du gène dans l'espèce d, puisque si ce dernier était ajouté, on observerait alors le gène dans l'espèce x, tel que démontré sur R dans la Figure 2.6. Dans le cas où g subit une duplication, on s'attend à ce que les deux gènes enfants se retrouvent dans la même espèce que g. Sinon, des pertes doivent être ajoutées, tel qu'illustré par la perte ajoutée entre a_2

et son parent sur R.

On peut donc expliquer les incongruences de G et son étiquetage en y ajoutant les gènes perdus manquants. Une *augmentation* de G est un arbre R obtenu en greffant à G des feuilles que l'on appelle les *pertes*. Chaque perte p est associée à une espèce $\sigma(p) \in V(S)$, en on appelle p une *perte en* $\sigma(p)$. Notons que $\sigma(p)$ peut être un noeud interne de S, contrairement aux gènes existants. Nous pouvons maintenant définir formellement la réconciliation entre un arbre de gènes et un arbre d'espèces dans le modèle DLS.

Définition 2.5. *Une réconciliation entre G et S est une augmentation R de G accompagnée d'un étiquetage ev satisfaisant, pour tout sommet interne g de R:*

1. *si $ev(g) = Duplication$, alors $s(g) = s(g_1) = s(g_2)$;*
2. *si $ev(g) = Speciation$ avec $s = s(g)$, alors $s(g_1) = s_1$ et $s(g_2) = s_2$.*

On dénote par $d(R)$ le nombre de noeuds de duplications de R et par $\ell(R)$ son nombre de pertes[7]. Le *coût de réconciliation* est $c(R) = d(R) + \ell(R)$. On peut alors se demander comment trouver la réconciliation R qui minimise ce coût. Le théorème suivant stipule qu'il suffit de minimiser les duplications tel que décrit plus haut, puis d'ajouter les pertes requises par la définition d'une réconciliation.

Théorème 2.6. *[69] La réconciliation R entre G et S qui minimise $d(R) + \ell(R)$ est unique et peut être obtenue de la façon suivante:*

1. *Appliquer le lca-mapping s à G;*
2. *Étiqueter G: $g \in V(G)$ est une duplication si $s(g) = s(g_1)$ ou $s(g) = s(g_2)$, et une spéciation sinon;*

[7]Notons que certaines définitions de réconciliation remplacent chaque perte p en s par le sous-arbre S_s, faisant en sorte que toutes les feuilles de R sont associées à des feuilles de S. Dans ce cas, $\ell(R)$ est donné par le nombre de sous-arbres de S greffés à R.

3. Ajouter les pertes: pour chaque branche gg', où g est le parent de g',

- *si g est une spéciation, greffer une perte en s pour chaque espèce s présente sur le chemin entre $s(g)$ et $s(g')$, excluant $s(g)$ et $s(g')$;*
- *si g est une duplication et $s(g) \neq s(g')$, greffer une perte en s pour chaque espèce s présente sur le chemin entre $s(g)$ et $s(g')$, incluant $s(g)$ et excluant $s(g')$.*

Duplication apparentes et non-apparentes

Les duplications d'un arbre réconcilié R peuvent être *apparentes* ou *non-apparentes*. Un noeud de duplication g ayant deux enfants g_1 et g_2 est une duplication apparente si $\Sigma(g_1) \cap \Sigma(g_2) \neq \emptyset$, et une duplication non-apparente sinon. Les duplications apparentes sont généralement considérées comme plus plausibles: s'il y a eu duplication en g, on s'attend à ce qu'au moins une espèce ait deux copies de g (ou deux copies d'un gène descendant de g). Les duplications non-apparentes, quant à elles, laissent croire qu'il y a eu duplication mais qu'une seule copie ait survécu dans chacune des espèces, impliquant la présence d'une perte dans chacune d'entre elles. L'arbre R de la Figure 2.6 illustre les deux types de duplication. La duplication en w est apparente, puisqu'un gène de l'espèce a est trouvé de part et d'autre du noeud. La duplication en y est quant à elle non-apparente, puisque ses feuilles descendantes représentent toutes des gènes d'espèces différentes. Notons que la réconciliation R est la plus parcimonieuse, et il est donc possible qu'une réconciliation optimale dans le modèle DLS contienne des duplications non-apparentes.

Les duplications non-apparentes sont qualifiées de "douteuses" dans la base de données Ensembl [61].

Réconciliations de grande taille

Si R est une réconciliation entre G et S, la taille de R peut être significativement plus grande que celle de G à cause de l'ajout des pertes. Ce fait est utilisé sans preuve dans le chapitre 4, et nous en faisons donc la démonstration ici.

Théorème 2.7. *Il existe une infinité d'arbres d'espèces S et d'arbres de gènes G tels que la réconciliation la plus parcimonieuse R entre G et S a $\Omega(|G||S|)$ noeuds.*

Démonstration. Soit S l'arbre binaire à n feuilles obtenu d'un chemin de longueur $n-1$ en ajoutant à chaque sommet une feuille voisine, puis une feuille voisine a supplémentaire à une des deux extrémités du chemin (on appelle parfois S une chenille, ou *caterpillar* en anglais). L'autre extrémité r est l'unique sommet de degré 2 et forme donc la racine. Soit b la feuille dont r est le parent. Notons que $n-2 \geq |S|/3$ sommets séparent a de r (puisque $|S| = 2n-1$). Soit (a,b) l'arbre de gènes à deux feuilles a_1 et b_1, tel que $s(a_1) = a$ et $s(b_1) = b$. Alors G est obtenu avec $|S|$ copies disjointes de (a,b) dont les racines sont jointes arbitrairement afin d'obtenir un arbre binaire. Il s'ensuit que $|G|$ a $2|S|$ feuilles et donc $4|S|-1$ noeuds. Dans la réconciliation R, chaque copie de (a,b) nécessite $n-2$ pertes sur la branche entre a_1 et son parent (une perte pour chaque noeud entre a et r dans S). Donc, le nombre de sommets de R est $|S|(n-2)+|G| \geq |S||S|/3 \in \Omega(|S||G|)$ car $|G| \in \Theta(|S|)$.[8] □

2.4.1 Parcimonie et autres réconciliations

La réconciliation décrite ci-haut, qui est utilisée à travers cette thèse, s'appuie sur le principe de la parcimonie. La minimisation des duplications

[8]Notons qu'au sens strict, G doit contenir un gène pour chaque espèce de S - nous laissons le soin au lecteur de vérifier qu'ajouter une copie de S dans G ne change pas le résultat)

et pertes est justifiée par l'idée que l'explication la plus simple est celle qui devrait être préférée. On l'appelle d'ailleurs la MPR dans la littérature, pour *Most Parsimonious Reconciliation*. La vie serait bien évidemment trop facile si la parcimonie était toujours respectée, et quelques autres modèles de réconciliation ont été imaginés. Dans [145], on propose une méthode Bayésienne permettant d'inférer une distribution de probabilités sur les réconciliations possibles. Le modèle étudié est plus complexe qu'ici et se base sur la vraisemblance statistique d'un mapping gène-espèce et d'une réconciliation. La fonction de vraisemblance est calculée à partir des séquences des gènes, d'un modèle d'évolution et de taux de substitutions sur les branches de l'arbre de gènes. Dans ce modèle, des événements sous-optimaux au point de vue parcimonie sont permis, puisqu'ils peuvent être mieux supportés au point de vue de la vraisemblance. Il est donc possible que le mapping gène-espèce le plus vraisemblable ne soit pas le lca-mapping, et que des pertes et duplications n'apparaissant pas dans la MPR soient inférées. Le modèle nécessite l'apprentissage de taux de pertes et duplications, ainsi qu'une estimation des temps représentés par les branches des arbres. Les auteurs de [45] ont étudié la distribution des réconciliations obtenues et ont montré qu'elle est dominée par un petit ensemble de réconciliations qui sont très proches de la MPR. L'exactitude de la MPR est toutefois débattue. Dans [131], une étude empirique sur des arbres simulés montre que dans 98% des cas, la réconciliation MPR est la plus statistiquement vraisemblable. Ceci contraste avec les résultats dans [110], où les auteurs obtiennent, sur des arbres de vertébrés, que dans 19% des cas, la réconciliation MPR n'est pas la plus vraisemblable - les auteurs expliquent cette différence par leur utilisation d'un modèle évolutif plus réaliste. Mentionnons aussi que la première étude a été faite sur des arbres simulés alors que la seconde a été faite sur des données réelles. Nous n'allons pas nous engluer davantage dans ce débat ici et allons nous contenter de la réconciliation MPR.

Notons par ailleurs qu'il existe des méthodes de réconciliation perme-

ttant d'incorporer d'autres événements évolutifs. En particulier, on peut s'intéresser au modèle DTLS, où les transferts horizontaux de gènes sont permis, i.e. le transfert de gènes d'une espèce à une autre espèce ayant existé à la même époque. La plupart des travaux faits dans ce modèle se basent sur la parcimonie. On cherche alors à minimiser les événements de duplications et transferts, et parfois aussi de pertes. Ce type de réconciliation étend la notion d'arbre d'espèces à celle de *réseaux phylogénétiques* [86], qui peut être vu comme un arbre orienté auquel on ajoute des arcs entre des noeuds qui sont séparés. On ne connaît habituellement pas ces arcs, et ils doivent souvent être inférés pendant la réconciliation. Toutefois, on veut éviter de produire des scénarios dans lesquels une espèce transmet un gène à une espèce ayant autre existé à une autre époque. En particulier, le réseau phylogénétique d'espèces doit être sans cycle. Le problème de minimiser le nombre de duplications et de transferts seulement devient NP-complet lorsqu'on ne sait pas quelles espèces ont cohabité [73], mais peut se faire en temps polynomial si les branches de l'arbre d'espèces sont datées [44]. Tel que mentionné plus haut, le lca-mapping n'est plus valable en présence de transferts, et il y a une multitude d'espèces auxquelles un gène ancestral a pu appartenir. L'algorithme de minimisation de duplications et transferts accomplit sa tâche en remplissant une table de programmation dynamique à deux dimensions: une pour les gènes ancestraux et une pour les espèces ancestrales. Un coût de réconciliation est calculé pour chaque paire (g, s) de gène et d'espèce représentant le mapping $s(g) = s$.

Aussi, on fait état dans [170] et [132] de la réconciliation avec pertes, duplications et *coalescence profonde*. Cette notion explique les différences entre G et S par la coexistence de plusieurs allèles d'un gène qui évoluent différemment. L'article propose une heuristique dont le temps et la précision théoriques restent vagues, et la complexité du problème demeure inconnue.

CHAPITRE 3

CONSTRUCTION ET CORRECTION D'ARBRES DE GÈNES

Nous décrivons dans ce chapitre les méthodes algorithmiques liées à la construction et à la correction d'arbres de gènes. Nous adoptons ici le processus d'inférence d'arbre de gènes basé sur les trois grandes étapes suivantes:

1. inférer une famille F de gènes homologues;
2. construire un arbre de gènes à partir des séquences de F alignées;
3. détecter les erreurs présentes dans G et le corriger/l'améliorer.

Nous survolons dans ce chapitre les méthodes permettant de réaliser chacune de ces étapes. Dans cette thèse, nous nous concentrons surtout sur la détection et la correction d'erreurs (chapitres 4, 5 et 6). Une bonne connaissance des méthodes actuelles d'inférence de familles et arbres de gènes est toutefois nécessaire afin de comprendre d'où proviennent les erreurs et ainsi motiver la nécessité d'algorithmes de correction. Nous décrivons donc brièvement dans la section 3.1 comment les familles de gènes sont généralement inférées et, dans la section 3.2, survolons les méthodes d'inférence phylogénétique traditionnelles. Celles-ci sont génériques et applicables à des jeux de données qui ne sont pas nécessairement issus de la biologie. Nous complétons donc cette section en présentant les récentes méthodes d'inférence qui s'appliquent spécifiquement aux arbres de gènes.

Aussi, comme nous le verrons dans le chapitre 7, il est possible de partitionner une famille en sous-groupes de gènes qui sont plus faciles à traiter (par exemple en groupe d'orthologues), puis d'inférer un arbre pour chacun de ces groupes. On obtient ainsi des arbres de gènes en lesquels on a un meilleur degré de confiance, que l'on peut ensuite combiner en un seul

"superarbre". Il existe bien des façons de définir ce qu'est un 'bon' superarbre et, conséquemment, une multitude de méthodes existent pour combiner plusieurs arbres en un. Nous passons en revue ces méthodes dans la section 3.3, en mettant l'emphase sur l'algorithme Build. Ces méthodes sont générales et peuvent s'appliquer à n'importe quel type d'arbre, et nous montrons comment Build peut être étendu au cas spécifique de superarbre de gènes dans le chapitre 7.

Nous terminons ce chapitre par la section 3.4, qui décrit les principales sources d'erreur au sein d'un arbre de gènes, puis propose une revue des méthodes actuelles de correction d'arbres de gènes. C'est dans le cadre du développement de telles méthodes de correction que viennent se placer les algorithmes présentés dans cette thèse.

3.1 Inférence de familles de gènes

Afin de regrouper des gènes en famille, on effectue typiquement une recherche BLAST "all-against-all" entre les gènes provenant de plusieurs génomes. C'est-à-dire, pour chaque gène g, on fait une recherche BLAST de sa séquence à travers tous les génomes (incluant son propre génome). Chaque gène g' trouvé ayant produit une e-value dépassant un certain seuil (e.g. 10^{-20}) est retenu comme étant un membre potentiel de la famille de g. Ces relations de similarité sont représentées sous la forme d'une *graphe BLAST* dont les sommets sont les gènes, et où g et g' forment une arête s'ils ont été retenus comme "frères" potentiels. Ces arêtes peuvent être étiquetées par le score BLAST correspondant. La dernière étape consiste à former des *clusters* dans ce graphe, c'est-à-dire partitioner les sommets en composantes fortement connexes. Il y a plusieurs façon d'accomplir cette tâche, mais la plus utilisée est sans doute l'algorithme du *Markov clustering* (MCL) [162]. En bref, l'idée derrière MCL est d'effectuer des marches aléatoires sur le graphe. Une telle marche aura du mal à sortir d'une composante fortement connexe du graphe, et avec suffisamment d'itérations on peut ainsi les

détecter [50].

À travers cette thèse, nous supposons que le travail d'identification de familles a été fait et que les gènes homologues nous sont donnés. Il est toutefois important de garder en tête que les méthodes actuelles d'inférence ne sont pas parfaites. Il y a effectivement une multitude de sources d'erreurs possibles: il peut y avoir eu des erreurs de séquençage, BLAST peut avoir manqué des occurrences de gènes homologues, les seuils BLAST utilisés peuvent avoir été mal choisis, ou encore une fausse similarité peut avoir été inférée (par exemple si beaucoup de familles partagent des régions d'ADN similaires "génériques", c'est-à-dire des sous-séquences réutilisables [37]). Un des impacts de telles erreurs est que l'on peut croire qu'un gène a été perdu chez un ancêtre puisqu'on ne le trouve pas dans la famille, alors que la cause est simplement un problème d'inférence.

3.2 Méthodes d'inférence phylogénétique

Les méthodes décrites dans cette section peuvent servir à reconstruire l'arbre évolutif de n'importe quel type d'éléments auxquels on peut associer une séquence ou une distance, ce qui inclut les gènes et les espèces. Nous ne faisons qu'un bref survol des méthodes génériques de reconstruction, et réferrons le lecteur à [57] pour de plus amples détails.

Afin de faire abstraction du type d'élément étudié, on appelle un élément particulier (e.g. un gène ou une espèce) une *unité taxonomique opérationnelle*, ou UTO. On divise les méthodes traditionnelles d'inférence de phylogénie en deux grandes catégories: les méthodes basées sur les caractères et les méthodes basée sur les distances.

3.2.1 Méthodes basées sur les caractères

On suppose ici que l'on peut associer à chaque UTO x un vecteur de traits de caractères $(x_1, \ldots, x_k)$ de même taille. Dans les premières reconstruc-

tions d'arbres d'espèces, ces vecteurs correspondaient aux phénotypes des espèces (i.e. les caractéristiques observables), chaque position i représentant un trait particulier et x_i la valeur pour ce trait (par exemple 0 ou 1 pour l'absence/présence du trait). Dans les méthodes d'inférence phylogénétique modernes, les traits considérés sont généralement les colonnes de l'alignement multiple de séquences d'ADN ou d'acides aminés.

Dans le reste de cette section, on suppose que l'on dispose d'un ensemble de séquences $\mathcal{S} = \{S_1, \ldots, S_n\}$, chacune de taille m, et qu'on veut reconstruire une phylogénie dont les feuilles sont étiquetées par $\mathcal{S}$.

Méthode de parcimonie maximum

Soit T un arbre ayant $\mathcal{L}(T) = \mathcal{S}$. Alors $\mathcal{S}$ peut servir à attribuer un score à T. Chaque position i est traitée indépendamment et permet d'assigner un coût $c(T, i)$ à T, le coût total de T étant $\sum_{i=1}^{n} c(T, i)$. On suppose, par le principe de parcimonie, que les séquences subissent un minimum de variation au cours de l'évolution. Pour chaque position i, on veut donc assigner à chaque noeud ancestral un caractère de façon à minimiser les changements de caractères entre les branches. Chaque changement de caractère de a à b induit un coût $c(a, b)$ donné. Soit xy une branche de T où $x \geq y$, avec c_x et c_y les caractères respectivement assignées à x et y. Le coût de xy est donné par $c(c_x, c_y)$, et le coût $c(T, i)$ de T est la somme du coût de chacune de ses branches. Le coût optimal de T est le plus petit coût possible donné par une assignation des noeuds internes. La Figure 3.1 montre un exemple d'assignation de caractères aux noeuds ancestraux d'un arbre donné, pour les caractères à une position donnée. L'arbre le plus parcimonieux est celui dont le coût optimal est le minimum parmi tous les arbres possibles. On limite typiquement la parcimonie aux arbres binaires, bien que la définition nous permette quand même d'attribuer un coût aux arbres non-binaires.

Si $c(a, b) = 1$ pour toute paire (a, b) de caractères telle que $a \neq b$, l'algorithme de Fitch [59] permet de trouver le coût optimal d'un arbre binaire

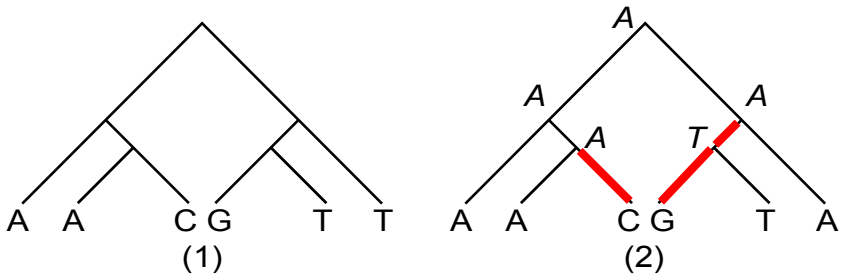

Figure 3.1: Exemple d'étiquetage parcimonieux pour un seul caractère. (1) Un arbre ainsi qu'un nucléotide assigné à chaque feuille. Cette assignation correspond typiquement au caractère à la i-ème position de chaque séquence assignée aux feuilles. (2) Un étiquetage des noeuds internes minimisant les changements sur les branches (en gras). Cet arbre nécessite au minimum trois changements.

T donné en temps $O(nm|\mathcal{A}|)$, où $\mathcal{A}$ est l'alphabet. Pour des coûts arbitraires de paires de caractères, l'algorithme Sankoff-Rousseau permet de trouver le coût optimal de l'arbre en temps $O(nm|\mathcal{A}|^2)$ [139]. Trouver l'arbre le plus parcimonieux pour un ensemble de séquences de caractères donné est NP-difficile, par une réduction du problème d'arbre de Steiner [92], qui consiste à trouver un sous-graphe connexe dont les arêtes ont un poids minimum, et qui contient un ensemble prescrit de sommets (ce sous-graphe doit être un arbre, d'où le nom 'arbre' de Steiner). Une méthode naïve est d'énumérer tous les $\prod_{i=1}^{k}(2k-5)$ arbres possibles, mais des heuristiques de *branch-and-bound* et de *hill-climbing* existent [159].

La méthode de parcimonie a perdu en popularité depuis que Felsenstein a démontré qu'elle n'est pas *statistiquement consistante* [54]. En effet, la probabilité d'obtenir le bon arbre par parcimonie ne tend pas nécessairement vers 1 lorsque la taille des séquences tend vers l'infini, et tend même vers 0 sur certains exemples.

Méthode de maximum de vraisemblance

En général, la méthode d'estimation par maximum de vraisemblance vise à inférer les paramètres d'un modèle statistique à partir de données observées. Dans le contexte phylogénétique, les données sont $\mathcal{S}$ et les paramètres à inférer sont l'arbre et le modèle évolutif des séquences. Nous donnons ici une explication abrégée de cette méthode.

Supposons que l'on connaisse un arbre T avec $\mathcal{L}(T) = \mathcal{S}$. Supposons aussi que l'on connaisse la fréquence $f(a)$ de chaque caractère a, ainsi que le modèle de substitution M, où $M(a, b)$ est la probabilité qu'un caractère a devienne le caractère b après une mutation. Nous sommes alors en mesure d'estimer la probabilité $P(\mathcal{S}|T, M)$ que l'on observe les séquences $\mathcal{S}$ aux feuilles de T. On appelle $P(\mathcal{S}|T, M)$ la fonction de *vraisemblance*. L'idée est d'essayer, pour chaque position i, toutes les combinaisons d'assignation de caractère possibles aux noeud internes de T (les assignations aux feuilles sont forcées par $\mathcal{S}$). À chaque assignation correspond une probabilité $P_i(\mathcal{S}|T, M)$ calculée à partir de f et M, et la probabilité d'observer la position i est la somme de chacune de ces probabilités (chaque assignation est un événement indépendant). Par exemple, si T a deux feuilles étiquetées en position i par les caractères a et b, et que r est l'étiquette de $r(T)$, on a $P_i(\mathcal{S}|T, M) = \sum_c f(c)M(c, a)M(c, b)$, où c prend les valeurs de tous les caractères possibles, $f(c)$ représentant la probabilité d'avoir c à la racine. La vraisemblance est $P(\mathcal{S}|T, M) = \prod_{i=1}^{n} P_i(\mathcal{S}|T, M)$.

Dans le cas des nucléotides, pour un seul caractère il y a $O(4^k)$ assignations possibles sur un arbre T à k noeuds internes. Toutefois l'algorithme de 'pruning' de Felsenstein [55], qui s'apparente à la programmation dynamique de Sankoff-Rousseau, permet de calculer cette probabilité en temps cubique.

Mais bien évidemment, on ne connaît pas T, et peut-être même pas M. On cherche alors à trouver les paramètres T et M qui maximisent la vraisemblance $P(\mathcal{S}|T, M)$. On fixe souvent M à un modèle connu tel que Jukes-Cantor, dans lequel chaque mutation est équiprobable, ou HKY85 dans lequel

les transversions ont κ fois plus de chances de se produire que les transversions [75]. Mais dans les cas plus complexes, d'autres paramètres tels que les longueurs de branches ou les taux de substitution par position peuvent venir s'ajouter.

Sans surprises, trouver T et les paramètres qui maximisent la vraisemblance est un problème NP-complet [31]. Comme la parcimonie, des heuristiques existent afin de limiter l'espace de recherche des arbres (e.g. [51, 71]). La méthode de maximum de vraisemblance est devenue plus populaire que la parcimonie, notamment parce qu'elle possède la propriété désirable de consistance statistique: si M est bien choisi, la probabilité d'inférer le bon arbre tend vers 1 lorsque la taille des séquences tend vers l'infini [171].

Méthode d'inférence bayésienne

Alors que la méthode de maximum de vraisemblance cherche l'arbre le plus susceptible d'avoir généré les séquences observées, l'inférence bayésienne s'attarde aux arbres les plus probables étant donné ces séquences. En d'autres termes, la vraisemblance évalue $P(\mathcal{S}|T)$ alors que les bayésiens s'attardent à $P(T|\mathcal{S})$. Plus précisément, $P(T|\mathcal{S})$ est une distribution sur les arbres que l'on cherche à inférer afin de trier les arbres. Pour ce faire, on utilise le théorème de Bayes stipulant que $P(T|\mathcal{S}) = \frac{P(\mathcal{S}|T)P(T)}{P(\mathcal{S})}$, d'où le nom de la méthode. La *probabilitité à priori* $P(T)$ doit être donnée, et est typiquement 1 divisé par le nombre total d'arbre.

Sans entrer plus loin dans les détails, cette distribution est presque toujours impossible à inférer de façon exacte, et on l'estime typiquement à l'aide de méthodes "Markov Chain Monte Carlo" (MCMC). Par exemple, l'algorithme Metropolis-Hastings[76] est une méthode MCMC qui explore l'espace des arbres aléatoirement de façon à converger vers la distribution originale. À chaque itération, un nouvel arbre candidat est proposé selon une distribution donnée, et ce candidat peut être accepté ou rejeté avec une certaine probabilité. Sur n itérations, l'algorithme explorera un arbre T ap-

proximativement $P(T|\mathcal{S}) \cdot n$ fois. On peut conserver l'historique de visite des arbres pour obtenir une estimation de la distribution. Plusieurs logiciels implémentent cette méthodologie [46, 135].

Le principe de l'inférence bayésienne est au coeur de plusieurs débats et controverses [84]. Certains critiquent la subjectivité inhérente à la probabilité à priori alors que d'autres estiment qu'elle permet d'inclure plus d'information basée sur les observations. L'histoire de l'inférence phylogénétique est d'ailleurs parsemée de discordes entre les adhérants à la parcimonie, au maximum de vraisemblance et à l'inférence Bayésienne. Une étude empirique a montré en 2006 que la méthode Bayésienne se comparait à celle du maximum vraisemblance en termes de précision, et que les deux étaient meilleures à ce niveau que la méthode de parcimonie [119].

Un des points à retenir par rapport aux méthodes bayésiennes est qu'elles n'infèrent pas un arbre mais plutôt un ensemble d'arbres. Puisque l'on cherche plus souvent à analyser un seul arbre, une pratique commune est de "combiner" les k arbres les plus probables en un seul - d'en faire un *superarbre consensus*, qui n'est pas nécessairement binaire. Les superarbres sont discutés à la section 3.3.

3.2.2 Méthodes basées sur les distances

Les méthodes de distance réduisent les séquences à une matrice D de taille $n \times n$ dans laquelle $D_{i,j}$ est une *distance* entre les UTO i et j. Cette distance peut-être calculée de plusieurs façons. Il peut par exemple s'agir du nombre de caractères différents entres les séquences S_i et S_j (la distance de Hamming), ou bien d'une distance d'alignement entre S_i et S_j. L'idée générale des méthodes par distance est de rassembler les UTO les plus "proches" dans l'arbre, la notion de proximité pouvant varier d'une méthode à l'autre. Le but principal de ces méthodes est de pouvoir affecter aux branches de l'arbre T des distances qui reflètent celles de D. Plus précisément, la somme des distances des branches sur le chemin entre i et j définit leur distance sur T,

et on cherche T et une affectation aux branches qui minimise les erreurs par rapport à D.

UPGMA

La méthode UPGMA [150], qui tient lieu de "Unweighted Pair Group Method with Arithmetic Mean", est une méthode de regroupement hiérarchique qui construit un arbre itérativement, en joignant à chaque étape deux sous-arbres pour n'en former qu'un. L'algorithme termine lorsqu'il n'y a plus qu'un seul arbre. Dans le cas de base, on ne peut joindre que des feuilles, on on choisit les feuilles i et j qui minimisent $D_{i,j}$ pour former le sous-arbre (i, j). Aux étapes suivantes, ce sont deux sous-arbres A et B que l'on doit choisir. La distance entre ces deux sous-arbres est donnée par $\frac{|\mathcal{L}(A)|}{|\mathcal{L}(B)|} \sum_{a \in \mathcal{L}(A), b \in \mathcal{L}(B)} D_{a,b}$, c'est-à-dire la moyenne des distances entre les feuilles de A et celles de B. Le temps requis pour construire un arbre est $O(n^3)$.

UPGMA n'est plus vraiment utilisé pour construire des arbres de gènes ou d'espèces. Un de ses désavantages est que la méthode suppose une horloge moléculaire, c'est-à-dire que chaque UTO évolue à la même vitesse, et donc que chaque gène/espèce observé a exactement le même âge.

Neighbor-Joining

Neighbor-Joining [138], ou NJ, peut être vu comme un successeur de UPGMA. C'est également une méthode de regroupement hiérarchique. La différence est dans le choix des sous-arbres qui sont joints à chaque étape. Au lieu de choisir les deux sous-arbres les plus proches en moyenne, NJ cherche les deux sous-arbres qui sont à la fois proches et *éloignés des autres sous-arbres*. Pour trouver les deux feuilles à joindre à la première étape, cette idée est exprimée en trouvant i et j qui minimisent $(n-2)D_{i,j} - \sum_{k=i}^{n}(D_{i,k} + D_{j,k})$. Ces feuilles i et j sont remplacées par le sous-arbre (i, j) qui sera ensuite traité

comme une feuille. Les distances entre (i, j) et les autres feuilles doivent être mises à jour de façon adéquate (voir [138] pour plus de détails). L'algorithme s'arrête lorsque l'arbre courant est binaire et non-enraciné.

NJ a l'avantage que si les distance de D reflètent exactement l'évolution des séquences ayant réellement eu lieu (et même peut-être à quelques erreurs près), alors l'arbre retourné est le vrai arbre - du moins, il existe un enracinement donnant le vrai arbre [6]. La méthode est aussi statistiquement consistante. Toutefois, les distances sont souvent erronées en pratique, et sont en plus moins informatives que les séquences, et par conséquent, NJ est moins précis que les méthodes de maximum vraisemblance et d'inférence bayésienne [119]. Néanmoins, NJ offre un excellent rapport entre la précision et la rapidité d'exécution, ce qui fait en sorte que c'est toujours un des algorithmes les plus utilisés.

3.2.3 Méthodes spécifiques à la construction d'arbres de gènes

La quasi-totalité des méthodes phylogénétiques conçues spécifiquement pour la construction d'arbres de gènes suivent le même principe: intégrer l'information de l'arbre d'espèces et/ou de la réconciliation aux méthodes décrites ci-haut. Ces méthodes, que nous survolons ici, sont assez récentes, datant d'au plus une dizaine d'années.

SYNERGY [167] (2007): cette méthode basée sur les distances incorpore, à la reconstruction de l'arbre de gènes, des informations basées sur l'arbre d'espèces et la synténie. C'est d'ailleurs la seule qui, à notre connaissance, considère cette dernière source d'information. La synténie, avec les distances d'alignement entre les séquences, est utilisée dans le calcul de la distance entre deux gènes - les paires de gènes faisant partie de blocs synténiques sont considérés comme plus proches, la taille du bloc étant considérée dans cette mesure de proximité. Une mesure de distance entre deux groupes de gènes est également proposée. SYNERGY construit récursivement un ensemble

d'arbres de gènes à partir de la racine de l'arbre d'espèces. Si s_1 et s_2 sont les enfants de la racine, un ensemble d'arbres de gènes G_1 (resp. G_2) est construit à partir des gènes présents dans un descendant de s_1 (resp. s_2). Une distance est calculée entre chaque paire d'arbres de $G_1 \cup G_2$, et Neighbor-Joining est ensuite utilisé pour joindre les arbres de $G_1 \cup G_2$ en un seul.

Prime-GSR [2] (2009): l'algorithme Prime-GSR (la signification de l'acronyme GSR n'est spécifiée nulle part) est une méthode Bayésienne qui étend celle décrite plus haut en y incorporant la réconciliation. Plus précisément, l'arbre d'espèces est utilisé pour rattacher à un arbre candidat G une probabilité à chacune de ses réconciliations possibles. Cette distribution fait partie de la probabilité associée à G et sert à déterminer la distribution de transition et d'acceptation/rejet dans les méthodes MCMC.

GIGA [161] (2010): l'algorithme GIGA (Gene tree Inference in the Genomic Age) est une méthode basée sur les distances qui incorpore l'arbre d'espèces et la réconciliation à la construction. La procédure s'apparente à UPGMA mais définit un ensemble de règles basées sur des informations génomiques permettant de réarranger l'arbre pendant sa construction. Par exemple, si deux sous-arbres A et B doivent être joints en un seul arbre AB à une certaine étape, et que A et B n'ont qu'un seul gène par espèce, alors la topologie de AB est exactement celle de l'arbre d'espèces (alors que UPGMA aurait joint $r(A)$ et $r(B)$ sous une racine commune). Sinon, B est greffé à A de façon à minimiser le coût de réconciliation. D'autres règles s'appliquent et permettent, entre autre, d'inférer les longueurs des branches sur l'arbre trouvé.

SPIMAP [131] (2011): les auteurs de SPIMAP définissent un modèle évolutif permettant de calculer la probabilité d'un arbre de gènes (et autres paramètres tels que la longueur des branches et les taux de duplications et pertes) étant donné un arbre d'espèces. La méthode construit d'abord un arbre Neighjor-Joining et, d'une façon similaire aux heuristiques pour le

maximum de parcimonie/vraisemblance, effectue une recherche à partir de cet arbre afin de trouver celui qui maximise la probabilité.

PHYLDOG [15] (2013): cet algorithme est le seul de cette liste à ne pas prendre en entrée un arbre d'espèces. En fait, étant donné un ensemble de familles de gènes (avec leurs séquences), PHYLDOG fait de la *co-estimation*, c'est-à-dire qu'il infère à la fois un arbre d'espèces et un arbre de gènes pour chaque famille. Ceci se traduit par la recherche des arbres de gènes, de l'arbre d'espèces et des événements de duplication et pertes qui sont les plus probables. Un arbre d'espèces initial est inféré et sert à estimer un premier ensemble d'arbres de gènes. Ceux-ci servent ensuite à améliorer l'arbre d'espèces, qui sert à ré-inférer les arbres de gènes, et ainsi de suite.

TERA [141] (2014): à partir d'un ensemble d'arbres de gènes (par exemple un échantillon obtenu par une méthode d'inférence bayésienne), cette méthode construit un arbre de gènes dans lequel chaque clade fait partie d'au moins un des arbres donnés en entrée (ce que les auteurs appellent un amalgame). Le choix de cet arbre est basé sur une combinaison du score de vraisemblance et du coût de réconciliation en tenant compte des duplications, pertes et transferts horizontaux. Un algorithme de programmation dynamique permet d'explorer l'espace des arbres.

3.2.4 Bootstrapping

Les méthodes de construction qui utilisent les séquences (incluant les méthodes par distances) permettent d'effectuer un *bootstrapping*, un des tests statistiques les plus répandus afin d'évaluer la robustesse d'un arbre inféré par un algorithme A [56]. Si M est la matrice des séquences, on obtient une nouvelle matrice M' en choisissant aléatoirement un certain nombre $n' < n$ de colonnes de M. La même colonne peut se répéter et les colonnes peuvent être ordonnées au hasard. On exécute ensuite A sur cette nouvelle matrice M' pour obtenir un arbre bootstrap T'. On répète ensuite cette

opération un grand nombre de fois, puis on évalue les branches de T. La valeur de bootstrap $b(xy)$ d'une branche xy est donnée par la proportion des arbres bootstrap qui contiennent la branche xy (un arbre enraciné "contient la branche" xy s'il contient $clade(x)$, où x est l'enfant de y, et un arbre non-enraciné contient xy s'il contient le split xy). La valeur $b(xy)$ représente donc le *support* de la branche xy. Les branches ayant un faible support sont très sensibles aux séquences, et on peut difficilement leur faire confiance étant donné les erreurs pouvant s'insérer dans les données. On fait généralement confiance aux branches ayant un support au-dessus de 0.8.

3.3 Superarbres

Dans cette section, nous passons en revue les diverses méthodes de construction de superarbres, qui permettent de combiner plusieurs arbres en un seul. De façon similaire aux méthodes d'inférence phylogénétique, les algorithmes existants sont pour la plupart génériques et s'appliquent à n'importe quel type d'arbre. Dans le chapitre 7, nous spécialisons la construction de superarbre aux arbres de gènes. Quelques notions formelles supplémentaires sont nécessaires avant de continuer.

Pour $L \subseteq \mathcal{L}(T)$, la *restriction* de T à L, dénotée $T|L$, est le plus petit sous-arbre de T contenant L, auquel on contracte les noeuds de degré 2 (i.e. la contraction des branches xy telles que y n'a qu'un seul fils x). Par exemple dans la Figure 3.2, G_1, G_2 et G_3 peuvent tous être obtenus par une restriction de T. On a $T|\{a_1, b_1, c_1, d_1\} = G_1$, $T|\{a_2, a_3, c_2\} = G_2$ et $T|\{b_1, e_1, a_1, a_2\} = G_3$.

Le *triplet* $xy|z$ est l'unique arbre binaire B avec $\mathcal{L}(B) = \{x, y, z\}$ tel que $lca(x, y) \neq lca(x, z) = lca(y, z)$. On dit que $xy|z$ est un triplet de T si $T|\{x, y, z\} = xy|z$. L'ensemble de tous les triplets de T est dénoté $tr(T)$. Dans la Figure 3.2, $G_2 = a_2a_3|c_2$ est un triplet, et ce triplet appartient à $tr(T)$.

On dit qu'un arbre T_1 *contient* T_2 si $T_1|\mathcal{L}(T_2) = T_2$. Soit $\mathcal{T} = \{T_1, \ldots, T_k\}$

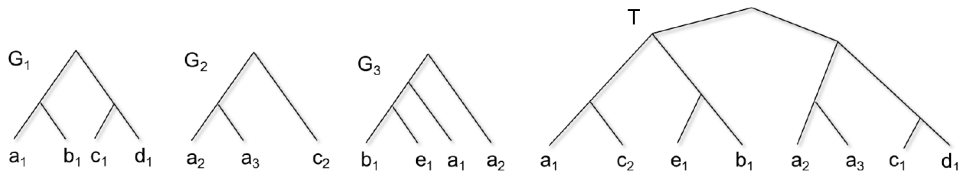

Figure 3.2: Trois arbres G_1, G_2 et G_3 consistants et un superarbre T compatible avec ces trois arbres, c'est-à-dire qui les contient tous.

un ensemble d'arbres. Si un arbre T contient chaque arbre de $\mathcal{T}$, on dit que T est *compatible* avec $\mathcal{T}$. Si un tel arbre existe, on dit que $\mathcal{T}$ est *consistant* (et inconsistant sinon). La Figure 3.2 illustre un exemple de trois arbres consistants ainsi qu'un superarbre T contenant ces trois arbres. Nous nous intéressons aux diverses façons de combiner les arbres de $\mathcal{T}$ en un seul *superarbre*. Nous résumons les principales méthodes de construction de superarbre, en mettant l'emphase sur les ensembles d'arbres consistants et sur l'algorithme Build, celui-ci étant utilisé dans le chapitre 7.

3.3.1 Méthodes de consensus

Les méthodes de consensus nécessitent que $\Gamma = \mathcal{L}(T_1) = \mathcal{L}(T_2) = \ldots = \mathcal{L}(T_k)$. Supposons que c'est le cas, et que $\mathcal{T}$ est inconsistant. On doit alors sacrifier certaines informations contenues dans les arbres de $\mathcal{T}$ afin d'en ressortir un *consensus*. Il y a évidemment plusieurs manières de trouver un tel terrain d'entente [18].

Consensus strict: l'arbre dont les clades sont exactement les clades contenus dans chaque arbre de $\mathcal{T}$. Formellement soit $C(\mathcal{T}) = \bigcap_{T_i \in \mathcal{T}} clades(T_i)$. Alors le consensus strict de $\mathcal{T}$ est l'unique arbre T tel que $clades(T) = C(\mathcal{T})$. Cet arbre existe toujours et est assez facile à retrouver: c'est l'arbre ayant comme sommets les éléments de $C(\mathcal{T})$ et $C_1, C_2 \in C(\mathcal{T})$ partagent une arête si C_2 est le plus petit clade tel que $C_1 \subseteq C_2$ (en notant que $\{x\} \in C(\mathcal{T})$ pour chaque élément de Γ et que $\Gamma \in C(\mathcal{T})$, et donc que chaque somme a un

parent unique).

Majority rule (la loi de la majorité): l'arbre qui inclut chaque clade apparaissant dans strictement plus que 50% des arbres. Pour construire l'arbre "majority rule", il suffit de lister tous ces clades et, comme pour l'arbre consensus strict, les lier par la relation d'inclusion. Il ne peut y avoir deux clades contradictoires insérés, puisque le contraire impliquerait la présence de deux clades contenus dans strictement plus de 100% des arbres.

Maximum Agreement SubTree (MAST): on cherche un plus grand sous-ensemble de feuilles $L \subseteq \Gamma$ tel que $T_1|L = T_2|L = \ldots = T_k|L$. En d'autres termes, quel est le minimum d'étiquettes à enlever pour que tous les arbres soient égaux? L'arbre ainsi obtenu est appelé le Maximum Agreement SubTree. Un consensus sur Γ peut ensuite être obtenu en ajoutant les feuilles retirées arbitrairement. Ce problème peut être résolu en temps polynomial pour deux arbres, mais est NP-complet pour trois arbres ou plus [79]. Toutefois, si tous les arbres donnés en entrée sont binaires, une solution peut être trouvée en temps polynomial [32, 53].

3.3.2 Superarbres lorsque $\mathcal{T}$ est consistant

Dans le cas où $\mathcal{T}$ est consistant, on voudra évidemment trouver un superarbre compatible avec $\mathcal{T}$. Build est un algorithme classique qui permet de vérifier si un ensemble de triplets $\mathcal{T}$ est compatible et, si oui, construire un superarbre compatible avec $\mathcal{T}$ [1]. Notons que puisqu'un arbre T_1 contient un arbre T_2 si et seulement si T_1 contient tous les triplets de $tr(T_2)$, vérifier la consistance d'un ensemble d'arbres $\mathcal{T}'$ se réduit à vérifier la consistance de $\bigcup_{T \in \mathcal{T}'} tr(T)$.

Le *graphe BUILD* pour $\mathcal{T}$ est le graphe $B(\mathcal{T}) = (V, E)$ dans lequel $V = \Gamma$ et deux éléments $x, y \in \Gamma$ partagent une arête si et seulement si il existe un triplet $xy|z \in \mathcal{T}$ pour un certain z[1]. L'idée derrière Build est la suivante: si

[1] $B(\mathcal{T})$ est appelé $\mathcal{T}$ dans le Chapite 7, notation que nous évitons ici pour éviter la

$B(\mathcal{T})$ est connexe et a plus d'un noeud, alors $\mathcal{T}$ est inconsistant. Sinon, ses composantes connexes $CC = C_1, \ldots, C_k$ forment les clades que l'on trouve sous la racine. Ensuite, pour déterminer la topologie du clade C_i, on appelle récursivement Build mais sur $\mathcal{T}|C_i$, c'est-à-dire l'ensemble des triplets $xy|z \in \mathcal{T}$ tels que $x, y, z \in C_i$. L'algorithme est détaillé ci-bas.

Algorithm 1 $BUILD(\Gamma = \{g_1, \ldots, g_n\}, \mathcal{T})$

1: **if** $|\Gamma| = 1$ **return** (g_1)
2: **if** $|\Gamma| = 2$ **return** (g_1, g_2)
3: Construire le graphe $B(\mathcal{T})$
4: **if** $B(\mathcal{T})$ est connexe **return** 'Inconsistant'
5: Soit $CC = \{C_1, \ldots, C_k\}$ les composantes connexes de $B(\mathcal{T})$
6: **for** composante connexe $C_i \in CC$ **do**
7: $T_i \leftarrow$ l'arbre retourné par $BUILD(C_i, \mathcal{T}|C_i)$
8: **if** $BUILD(C_i, \mathcal{T}|C_i)$ a échoué **return** 'Inconsistant'
9: **end for**
10: **return** L'arbre obtenu en joignant $r(T_1), \ldots, r(T_k)$ sous un parent commun

L'arbre T contenant chaque arbre de $\mathcal{T}$ retourné par Build n'est pas nécessairement binaire. Il est toutefois *minimalement résolu*, c'est à dire que toute contraction de branche de T fait en sorte qu'il n'est plus compatible avec $\mathcal{T}$. De plus, on sait que toute binarisation de T contient aussi chaque arbre de $\mathcal{T}$. Build ne produit toutefois qu'un seul arbre (ou du moins, qu'une seule classe d'arbre si on considère les binarisations), alors qu'il peut en exister beaucoup plus qu'un. L'algorithme Build a été étendu par [34] et [115] afin de produire toutes les solutions possibles. Ce travail a été continué dans [142], où les auteurs décrivent décrivent l'algorithme All-Min-Trees qui donne tous les arbres minimalement résolus, chacun pouvant être trouvé, individuellement, en temps polynomial. Toutefois, pour des arbres à n étiquettes, il peut exister $\Omega(\frac{n}{2}^{\frac{n}{2}})$ solutions minimalement résolues [87].

confusion.

3.3.3 Superarbres lorsque $\mathcal{T}$ est inconsistant

Si $\mathcal{T}$ est inconsistant, la question du "meilleur" superarbre n'est pas claire et est un sujet de débats. Bien que ce cas ne va pas nous intéresser, nous effleurons ici les méthodes de construction de superarbre dans le cas où $\mathcal{T}$ est inconsistant.

Maximum de triplets : étant donné un ensemble de triplets $\mathcal{T}$, on cherche à trouver un ensemble $\mathcal{T}' \subseteq \mathcal{T}$ de cardinalité maximum tel que $\mathcal{T}'$ est consistant. Ou encore, quel est le nombre minimum de triplets à sacrifier pour pouvoir construire un arbre avec Build? Sans surprises, ces deux problème sont NP-complet, mais ne sont pas équivalents en terme d'approximabilité. Le maximum de triplets consistants peut être approximé à un facteur 3, tandis que le minimum de triplets à retirer pour avoir un ensemble consistant n'admet pas d'algorithme d'approximation avec un facteur constant [20].

MinCutSupertree [143] : Build arrête son exécution sur un ensemble de triplets $\mathcal{T}$ lorsque le graphe $B(\mathcal{T})$ connexe. L'idée du *MinCutSupertree* est de forcer Build à continuer son exécution dans de telles situations, ce qui peut s'accomplir en rendant $B(\mathcal{T})$ non-connexe par la suppression d'arêtes. À chaque arête xy, on attribue un poids $f(xy)$ obtenu par la fréquence de xy en triplets, c'est-à-dire le nombre de triplets de la forme $xy|z$ dans $\mathcal{T}$. L'algorithme MinCut peut ensuite être appliqué pour trouver le poids minimum d'arêtes (i.e. le nombre triplets) à sacrifier pour déconnecter $B(\mathcal{T})$. Bien que rapide, cette méthode a été critiquée dans [65] puisqu'elle effectue parfois des choix arbitraires, donnant lieu à des triplets ou sous-arbres qui ne sont supportés par aucun arbre de l'entrée.

Représentation matricielle avec parcimonie (MRP) [8] : on peut représenter les clades des arbres de $\mathcal{T}$ avec une matrice M. Les rangées de M sont les n étiquettes et les colonnes sont les noeuds internes des arbres de $\mathcal{T}$. On pose $M_{\ell,x} = 1$ si ℓ est une feuille descendante du noeud x, 0 si ℓ est présente dans le même arbre que x mais n'en descend pas, et ? si ℓ n'est pas

dans le même arbre que x. La matrice M peut alors être interprétée comme un ensemble de séquences caractères-états, tel que vu dans la section 3.2. On peut alors utiliser la parcimonie pour retrouver le meilleur arbre pour M. Une des critiques envers MRP est qu'il se peut que le superarbre contienne des triplets (ou des sous-arbres) en contradiction avec *chaque* arbre de $\mathcal{T}$, la parcimonie n'offrant aucune garantie quant à la compatibilité [65]. MRP est néanmoins une des méthodes les plus utilisées. Elle a notamment servi à reconstruire le superarbre des angiospermes et des mammifères.

3.4 Détection d'erreurs et correction d'arbres de gènes

Nous survolons d'abord dans cette section les principales sources d'erreurs pouvant affecter un arbre de gènes, puis nous décrivons les méthodes de correction qui ont été publiées à ce jour.

3.4.1 Sources d'erreur dans les arbres de gènes

Il existe une multitude de sources d'erreur pouvant faire en sorte qu'un arbre de gènes est erroné. Nous en énumérons ici quelques unes:

Mauvaise différentiation au niveau des séquences: les signaux phylogénétiques envoyés par les séquences peuvent parfois être trop faibles pour en déduire des arbres précis [123]. Il se peut que certains gènes aient des séquences trop similaires, ou bien que les différences entre trois gènes ou plus n'indiquent pas de topologie claire (par exemple, que faire avec trois gènes tous différents mais équidistants, i.e. chaque paire de gènes a la même distance). Face à de telles ambiguïtés, les algorithmes de construction d'arbre doivent soit effectuer un choix, soit laisser des parties de l'arbre non-résolues, donnant ainsi lieu à des polytomies.

Famille de gènes incorrecte: les méthodes de regroupement de gènes en famille ne sont pas parfaites et peuvent parfois manquer des gènes homologues lors de l'inférence d'une famille. Il se peut par exemple que certains génomes

étudiés soient partiellement séquencés et que certains gènes nous soient inconnus. De plus, certains gènes "intrus" peuvent se retrouver au sein d'une famille à laquelle ils n'appartiennent pas. Or, la présence ou l'absence d'un gène peut avoir un impact significatif sur la topologie d'un arbre inféré par les méthodes énumérées dans la section 3.2 [136].

Homoplasie: l'homoplasie se définit par la possession de traits similaires ou identiques par deux espèces, alors que ces traits n'ont pas été hérités par un ancêtre commun de ces espèces. L'homoplasie est souvent (mais pas toujours) due à une adaptation des deux espèces à des conditions environnementales similaires. On peut aussi avoir présence d'homoplasie simplement par chance. On distingue trois types d'homoplasie.

L'*évolution convergente* se produit lorsque deux espèces n'ayant pas de lien de parenté direct évoluent le même trait de façon indépendante. Un exemple classique est la capacité de voler, qui a été acquise par les oiseaux, les mouches et les chauve-souris alors que le dernier ancêtre commun de ces espèces ne volait pas.

L'*évolution parallèle* se produit lorsque deux espèces divergent de leur dernier ancêtre commun, mais de la même façon. Les deux espèces sont donc similaires, mais différentes de leur ancêtre. Les feuilles des plantes sont reconnues pour évoluer de façon parallèle, puisque lors d'un changement environnemental, les plantes ont tendance à s'y adapter d'une façon similaire [124].

Finalement, le *renversement évolutif* se produit lorsqu'un trait avancé acquis par une espèce revient à sa version primitive ou est perdu. L'espèce redevient donc similaire à celles qui n'avaient pas acquis ce trait. Par exemple, on sait que l'ancêtre de la grenouille *Gastrotheca guentheri* possédait une dentition inférieure, mais elle a été perdue il y a 230 millions d'années [168] (les raisons explicant cette perte ne sont pas claires, mais on croit qu'un changement environnemental rendait cette dentition inutile). La grenouille a d'ailleurs "regagné" cette dentition il y a 2-30 millions d'année, un rare cas

de perte et de regain de fonction.

Notons que le phénomène d'homoplasie peut aussi s'appliquer au niveau moléculaire, et donc que des positions correspondantes de deux séquences peuvent devenir similaires alors qu'elles ne l'étaient pas. Au niveau phylogénétique, le problème avec l'homoplasie, plus particulièrement avec l'évolution convergente, est que deux espèces ou gènes en apparence similaires peuvent être distants au point de vue évolutif. Puisque toutes les méthodes de reconstruction phylogénétique tentent de rassembler les gènes et espèces qui se ressemblent le plus, autant dans les modèles de parcimonie que ceux basés sur la vraisemblance, de erreurs seront nécessairement introduites en présence d'homoplasie - à moins de pouvoir la détecter, ce qui est à jour une tâche difficile à accomplir.

Sous-optimalité des algorithmes ou de l'évolution: les critères d'optimisation utilisés en inférence phylogénétique se traduisent souvent en des problèmes NP-complet, et les algorithmes courants ne peuvent souvent explorer qu'un sous-ensemble de l'espace de solutions. L'optimalité de la solution n'est donc pas garantie. De plus, même si c'était le cas, il peut parfois y avoir un nombre exponentiel de solutions possible lorsque l'on optimise un ou des critères combinatoires par parcimonie. Cette optimisation est souvent basée sur des paramètres qui doivent être spécifiés par l'utilisateur, et ce choix peut mener à des solutions différentes. Par exemple, on peut vouloir résoudre une polytomie en minimisant la réconciliation, mais en attribuant un coût différent aux duplications et aux pertes (e.g. si on suspecte que certaines espèces sont mal séquencées, des pertes en cette espèces s'expliquent peut-être par un manque d'informations, et on voudra attribuer un coût plus faible aux pertes). Chaque coût peut alors mener à un nombre exponentiel de résolutions optimales. Notons aussi que par ailleurs, l'évolution ne suit pas toujours un chemin optimal, ou ne correspond pas toujours au scénario le plus probable.

3.4.2 Correction d'arbres par résolution de polytomies

Tel qu'expliqué plus tôt, une incertitude au niveau d'un arbre peut mener à des polytomies. L'une des façons les plus populaires de résoudre de tels noeuds non-binaires est de trouver une résolution qui donne le minimum de duplications et pertes après réconciliation avec un arbre d'espèces donné S. Un premier algorithme accomplissant cette tâche a été proposé au début des années 2006 et implémenté dans le logiciel NOTUNG [47]. Il s'agit d'un algorithme de programmation dynamique sur S qui nécessite un temps $O((|S||G|\Delta^2)$ pour résoudre tout un arbre, où Δ est la taille maximum d'une polytomie. La méthode permet de trouver toutes les solutions possibles, ainsi que d'attribuer un poids différent aux pertes et aux duplications.

Ce travail a été suivi de l'algorithme de Chang et Eulenstein en 2006 [23], qui nécessite quant à lui un temps d'exécution $O(|S||G|^2)$. En 2012, nous avons développé un algorithme nécessitant un temps $O(|S| + |P|)$ par polytomie P, et donc un temps $O(|S||G|)$ pour un arbre entier. La méthode, présentée au chapitre 4, permet de retrouver toutes les solutions possibles, mais ne supporte pas les poids différents aux duplications et pertes.

Zheng et Zhang [177] ont répondu deux ans plus tard avec un algorithme avec la même complexité pour une polytomie individuelle, mais pouvant résoudre toutes les polytomies d'un arbre entier en temps $O(|G|)$. La méthode ne permet toutefois pas de trouver toutes les solutions, ni d'attribuer des poids distincts aux événements.

Nous avons récemment renchéri avec une méthode atteignant la même complexité, mais offrant l'avantage de retrouver toutes les solutions possibles [98]. L'algorithme peut aussi résoudre toutes les polytomies d'un arbre entier en temps $O(|S||G|)$ lorsque des poids différents sont attribués aux duplications et pertes. Il permet même d'affecter à chaque espèce individuelle ses propres poids, mais ce pour un temps d'exécution de $O(|S||G|^2)$.

Dans [96], nous avons étudié une autre façon de résoudre les polytomies. L'objectif est de trouver une résolution qui minimise le nombre de *duplica-*

tions non-apparentes. L'article présente un algorithme d'approximation qui donne, dans le pire cas, une solution contenant deux fois plus de duplications non-apparentes que nécessaire. La complexité du problème demeure ouverte.

3.4.3 Correction d'arbre par exploration du voisinage des arbres

Une façon plus heuristique de corriger un arbre de gènes G est d'explorer son *voisinage*, c'est-à-dire l'ensemble des arbres que l'on peut obtenir en effectuant une opération sur G. Typiquement, les voisins de G sont les $O(n)$ arbres possibles résultant d'une opération NNI sur G. On peut alors choisir un voisin G' de G, soit de façon aléatoire ou bien en considérant le "meilleur" voisin, puis répéter ce processus. À la fin de l'exploration, on retient le meilleur arbre qui a été rencontré. En général, l'avantage de ces méthodes est qu'elles permettent d'explorer un grand nombre de candidats potentiels intéressants (en supposant que l'arbre initial était lui-même un bon candidat). Le désavantage est qu'il n'y a pas vraiment de garantie quant à l'optimalité de la solution retournée et au temps d'exécution, à moins que celui-ci soit forcé à un maximum.

L'idée de réarrangements locaux a d'abord été implémentée dans la première version de Notung [28]. Le voisinage de G considéré est obtenu en effectuant des NNI sur les branches ayant un faible support statistique, et le meilleur arbre est celui qui minimise duplications et pertes (en fait Notung retient tous les arbres optimaux s'il y en a plusieurs). Le processus s'arrête après une seule étape, mais peut être répété par l'utilisateur si désiré.

L'algorithme de Górecki et Eulenstein, publié en 2012 [70], permet d'effectuer essentiellement la même tâche mais sur un arbre non-enraciné, en plus de proposer une racine. C'est-à-dire, pour un k donné et un arbre G non-enraciné, la méthode trouve un arbre G' à une distance NNI au plus k tel que G' donne un minimum de duplications et pertes après enracinement (i.e. lorsqu'on prend l'enracinement qui minimise la réconciliation).

La méthode TreeFix, également parue en 2012 [169], ajoute la vraisem-

blance statistique à l'exploration du voisinage des arbres. Il existe une multitude de tests basés sur la vraisemblance (telle que décrite dans la section 3.2) permettant de vérifier qu'un arbre est aussi "statistiquement valable" qu'un autre (e.g. le test SH [147] ou le test KH [146]). TreeFix explore le voisinage NNI ou SPR de G et ne conserve que les voisins qui sont au moins aussi bien supportés statistiquement que G. Ces arbres sont retenus avec une certaine probabilité établie selon leur coût de réconciliation, et le processus est répété avec le meilleur arbre conservé. Cette méthode a l'avantage de pouvoir passer par des arbres sous-optimaux pendant son exploration, permettant parfois d'éviter d'être prise dans un minimum local.

3.4.4 Autres méthodes de correction

Dans [43], Doroftei et El-Mabrouk proposent de corriger un arbre en y éliminant les "gènes intrus". Ceci se traduit par la recherche du minimum de feuilles à enlever sur G pour qu'il ne contienne aucune duplication non-apparente. L'article étudie quelques cas spécifique et propose une heuristique générale, mais le problème a été démontré NP-complet (et même APX-difficile) quelques temps après [42].

Finalement dans [154], Swenson et El-Mabrouk font état d'un "biais fonctionnel" pouvant être introduit par les méthodes classiques de contruction d'arbre de gènes. En bref, les gènes orthologues ayant une fonction similaire (appelés *isoorthologues* dans l'article) ont tendance à être regroupés dans l'arbre, produisant ainsi des sous-arbres n'ayant que des orthologues. La vraie histoire de gènes peut toutefois prôner d'autres types de scénarios tout aussi valables. Les auteurs proposent une méthode de correction d'un arbre de gènes en supposant qu'il contient un biais fonctionnel.

CHAPITRE 4

AN OPTIMAL RECONCILIATION ALGORITHM FOR GENE TREES WITH POLYTOMIES

Manuel Lafond[1] , Krister M.Swenson[1], Nadia El-Mabrouk[1]

Le modèle évolutif des gènes décrit dans les chapitres précédents suppose qu'un gène ancestral g a toujours deux enfants: les copies transmises aux deux espèces descendantes dans le cas d'une spéciation, et les deux copies identiques du gène dans le cas d'une duplication. Ce modèle implique donc qu'un arbre de gènes doit toujours être binaire. Toutefois, dans de rares cas, cette supposition peut s'avérer fausse. Par exemple, on sait qu'une sous-espèce de lézard, les Cordylidae, a subi des radiations au cours de son histoire, ce qui a produit des mutations très rapides au sein de son génome, donnant lieu à des gènes et espèces ayant plus de deux descendants [151]. On dit que ces polytomies sont de type "*dur*", puisqu'elles représentent la réalité et ne devraient pas être altérées. Nous n'allons pas explorer ces cas dans cette thèse, et on suppose qu'ils ne se présentent jamais.

Même avec cette simplification, il se peut qu'un arbre de gènes (ou bien d'espèces) contienne des polytomies, c'est-à-dire des noeuds non-binaires. L'une des raisons principales est que l'on ne sait pas comment résoudre certaines parties de l'arbre - on préfère alors les laisser non-résolues en ne déterminant pas de topologie exacte. De telles incertitudes sont souvent causées par des signaux trop faibles ou même contradictoires au niveau des séquences. On peut penser aux méthodes d'inférence bayésienne décrites dans la section 3.2, qui infèrent plusieurs arbres desquels on veut extraire un consensus. Mais si plusieurs topologies contradictoires sont aussi bien supportées dans certaines parties de l'arbre, il n'y a parfois pas de compromis

[1]DIRO, Université de Montréal, Canada

possible. Le résultat est un arbre non-binaire. Aussi, les tests de bootstrapping nous donnent une valeur de support statistique sur les branches d'un arbre. Les branches ayant un faible support (e.g. < 0.8) peuvent être remises en question, et une façon de le refléter est de contracter la branche. L'effet est la création d'une polytomie dans l'arbre. En constraste avec les polytomies dures, ces polytomies sont appelées "*douces*", puisqu'elles ont été créées artificiellement et ne demandent qu'à être résolues.

L'article présenté dans ce chapitre propose une méthode permettant de résoudre une polytomie sur un arbre de gènes en minimisant duplications et pertes en temps $O(|S|)$, où S est l'arbre d'espèces. Étant donné que chaque polytomie d'un arbre de gène peut être résolue indépendamment, le temps nécessaire pour binariser un arbre G est $O(|G||S|)$. Pour une polytomie P, l'algorithme calcule une table de programmation dynamique M qui est, conceptuellement, de taille $O(|P||S|)$ La linéarité de l'agorithme exploite le fait que les rangées de M ne sont pas aléatoires: elles peuvent être représentées par ce qu'on appelle une *fonction de coupe* définie par seulement trois paramètres.

La meilleure complexité connue pour résoudre une seule polytomie était de $O(|S||P|^2)$ avant la parution de l'article, comparé à la complexité $O(|S|)$ présentée ici. Le titre suggère que le nouveau temps obtenu est imbattable, alors que Zheng et Zhang ont proposé une solution nécessitant un temps $O(|G| + |S|)$ [177] En fait, l'optimalité décrite dans l'article présenté ici réfère au temps requis pour obtenir un arbre *reconcilié* R, i.e. qui inclut les noeuds de perte. Selon le Théorème 2.7, R peut avoir une taille $O(|G||S|)$, ce qui correspond à la complexité atteinte ici. La taille de la sortie donne une borne inférieure au temps de l'algorithme, d'où l'optimalité. En revanche, la méthode n'est pas optimale si seule la binarisation de G, et non sa réconciliation, nous importe.

Notons aussi que l'article utilise une définition d'arbre réconcilié différente de celle donnée dans le chapitre 2. Ces définitions sont toutefois équivalentes.

4.1 Abstract

Reconciliation is a method widely used to infer the evolutionary relationship between the members of a gene family. It consists of comparing a gene tree with a species tree, and interpreting the incongruence between the two trees as evidence of duplication and loss. In the case of binary rooted trees, linear-time algorithms have been developed for the duplication, loss, and mutation (duplication + loss) costs. However, a strict prerequisite to reconciliation is to have a gene tree free from error, as few misplaced edges may lead to a completely different result in terms of the number and position of inferred duplications and losses. How should the weak edges be handled? One reasonable answer is to transform the binary gene tree into a non-binary tree by removing each weak edge and collapsing its two incident vertices into one. The created polytomies are "apparent" as they do not reflect a true simultaneous divergence of many copies from a common ancestor, but rather a lack of resolution. In this paper, we consider the problem of reconciling a non-binary rooted gene tree G with a binary rooted species tree S, were polytomies of G are assumed to be apparent. We give a linear-time algorithm that infers a reconciliation of minimum mutation cost between a binary refinement of a polytomy and S, improving on the best known result, which is cubic. This implies a straightforward generalization to a gene tree G with nodes of arbitrary degree, that runs in time $O(|S||G|)$, which is shown to be an optimal algorithm.

4.2 Introduction

The evolutionary history of a gene family is determined by a combination of microevolutionary events at the sequence level, and macroevolutionary events (duplications, losses, horizontal gene transfer) affecting the number and distribution of genes among genomes [47]. While sequence similarity can be considered as a footprint of microevolution and used to construct a

gene tree G for the gene family, macroevolution is harder to predict as it is not explicitely reflected by the gene tree. Having a clear picture of the speciation, duplication and loss mechanisms that have shaped a gene family is however crucial to the study of gene function. Indeed, following a duplication, the most common occurrence is for only one of the two gene copies to maintain the parental function, while the other becomes non-functional (pseudogenization) or acquires a new function (neofunctionalization) [173].

Reconciliation, first introduced by Goodman in 1979 [67] — and since widely studied and implemented in comparative genomics software [52] — is a method that compares the gene tree G with a phylogeny S of the considered species (species tree), and interprets the incongruence between the two trees as evidence describing evolution of the gene family through duplication and loss. A reconciliation $R(G, S)$ is a tree obtained from G by inserting "lost" branches so that the obtained tree is in agreement with the phylogeny S. As there can be several reconciliations for a given tree pair, a natural approach is then to select one, or a subset, that optimize some probabilistic [2, 5] or combinatorial [107] criterion such as the number of duplications (duplication cost), losses (loss cost) or both combined (mutation cost). Reconciliation of binary rooted trees is a well-studied problem, and linear-time algorithms based on the so called lowest common ancestor (LCA) mapping have been developed for the duplication, loss and mutation costs [28, 174, 178]. Generalizations of reconciliation accounting for horizontal gene tranfers have also been considered [44]. In particular, minimizing the number of duplications, losses and transfers has been shown to be computationally hard [73], but feasible in polynomial time if the input species tree is dated [44].

The fundamental hypothesis behind reconciliation is that the gene tree reflects the true phylogeny of the gene family. Therefore, a strict prerequisite is to have both gene tree and species tree free from error [43, 72]. Unfortunately gene trees are not always well-supported, and frequently many equally-supported trees are obtained as the output of a phylogenetic method.

Typically bootstrap values are used as a measure of confidence in each edge of a phylogeny. How should the weak edges of a gene tree be handled? One strategy adopted in [28] is to explore the space of gene trees obtained from the original tree G by performing Nearest Neighbor Interchanges around weakly-supported edges. Another reasonable answer is to transform the binary gene tree into a non-binary tree by removing each weak edge and collapsing its two incident vertices into one. A polytomy (node with more than two children) in a gene tree is called *true* (or *hard*) if it reflects a true simultaneous divergence of its children from a common ancestor, and it is called *apparent* (or *soft*) otherwise [148]. Implicitly, polytomies of a gene tree obtained by the method of collapsing short or poorly supported internal branches are apparent polytomies, reflecting a lack of resolution.

In this paper, we consider the problem of reconciling a non-binary rooted gene tree G with a binary rooted species tree S, where polytomies of G are assumed to be apparent. More precisely, we seek out a reconciliation of minimum mutation cost between a binary refinement of G and S. Chang and Eulenstein were the first to consider this problem [23]. They showed that each polytomy P can be treated independently in $O(|S| \times |P|^2)$ time, implying an $O(|S| \times |G|^2)$ algorithm for the entire tree. In a recent paper [175], a linear-time algorithm is developed for reconciling a non-binary gene tree G with a binary species tree S, but for the duplication cost. The output is a reconciliation with optimal loss cost over all the reconciliations with the optimal duplication cost, which does not necessarily minimize the mutation cost. Here, we describe an algorithm that infers the minimum mutation cost reconciliation between P and S in $O(max(|P|, |S|))$ time, implying an $O(|G| \times |S|)$ algorithm over the entire gene tree. This algorithm is optimal, since there exists a family of instances leading to a most parsimonious reconciliation of size $\Omega(|G| \times |S|)$.

4.3 Preliminary notation

In this paper, all the trees are considered rooted (we ommit to mention it each time). Given a tree T, we denote by T_x the subtree of T rooted at x, and by $L(T_x)$ (or simply $L(x)$ if unambiguous) the set of leaves of T_x. We also denote by $root(T)$ the root of T, by $V(T)$ the set of nodes of T and by $|T|$ the number of nodes $|V(T)|$ of T. The *degree* of an internal node x in a tree T is the number of children of x. If T is binary, we denote by x_l and x_r the two children of x in T.

A *phylogeny* over a set L is a tree with internal nodes of degree 2 or more, uniquely leaf-labeled by L. A *polytomy* (or star tree) over a set of L is a phylogeny with a single internal node, which is of degree $|L|$, adjacent to each leaf of L. For example, the tree G in Figure 4.2 is a polytomy.

A *species tree* S is a phylogeny over a set of species Σ, which represents the evolutionary relationship between these species. Similarly, we can consider the evolutionary relationship between a family of genes Γ, that appear in the genomes of Σ: a *gene tree* G for Γ is a phylogeny accompanied by a function $g : \Gamma \to \Sigma$ indicating the species where each gene is found. See Figure 4.1 for an example. Given a gene tree G, we denote by $\mathcal{S}(G_x)$ the subset of Σ corresponding to $L(G_x)$ (i.e. $\mathcal{S}(G_x) = \{g(l) \mid l \in L(G_x)\}$).

In this paper, we assume a binary species tree S and a non-binary gene tree G. As stated in the introduction, the polytomies of G are considered apparent (i.e. reflecting non resolved parts of the tree). The goal is then to find a "binary refinement" of G. For any internal node x of G with children $\{x_1, x_2, \ldots, x_n\}$, any rooted binary tree on the set of leaves $\{G_{x_1}, G_{x_2}, \ldots, G_{x_n}\}$ is a *refinement* of the polytomy G_x. The following definition generalizes this fact.

Definition 4.1 (binary refinement)**.** *A* binary refinement $B(G)$ *of a gene tree* G *is defined as follows.*

- *If* r *is a leaf then* $B(G) = G$*;*

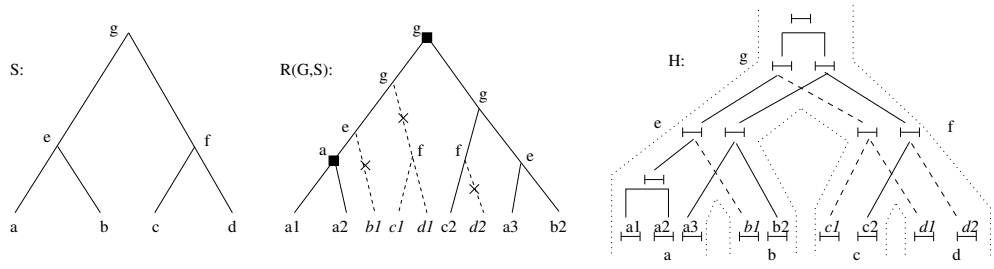

Figure 4.1: S is a species tree over $\Sigma = \{a, b, c, d\}$; $R(G, S)$ is a reconciliation between S and the gene tree G represented by plain lines. Here $\Gamma = \{a_1, a_2, a_3, b_2, c_2\}$, and for each $x_i \in \Gamma$, $g(x_i) = x$. Internal nodes of $R(G, S)$ are labeled according to the LCA mapping. Artificial genes $\{b_1, c_1, d_1, d_2\}$ are added to illustrate lost branches. Duplication nodes are indicated by bold squares, and loss leaves are represented by crosses. This reconciliation has cost 5: 2 duplications and 3 losses; H illustrates the history that has led to the gene family Γ. H is the same tree as $R(G, H)$, but represented differently (embedded in the species tree).

- *Otherwise, $B(G)$ is a rooted binary tree on the set $\{B(G_1), B(G_2) \ldots, B(G_n)\}$, where G_i is the tree rooted at the ith child of root(G) (for some ordering of the children), and $B(G_i)$ is a binary refinement of G_i.*

4.3.1 Histories and reconciliation

We study the evolution of a family of genes Γ taken from genomes Σ through duplication and loss. Conceptually, a *duplication/loss/speciation history* (or simply *history*) is a tree H reflecting the evolution from a single ancestral gene to a set of genes through duplication, loss, and speciation events. Given a binary gene tree G for the gene family and a species tree S for Σ, a reconciliation is a history obtained from G, in "agreement" with the phylogeny S. In this section we formally define history and reconciliation, as well as presenting tools for working with them. All these concepts are illustrated in Figure 4.1.

The most popular method for finding a parsimonious reconciliation is based on the "LCA mapping". The *LCA mapping* between G and S, denoted by $\mu()$, maps every node x of G to the *lowest common ancestor* of $\mathcal{S}(G_x)$ in S, which is the common ancestor of $\mathcal{S}(G_x)$ in S that is farthest from the root. We call $\mu(x)$ the *label* of x. A node x of G is considered a *duplication* with respect to S if and only if $\mu(x_\ell) = \mu(x)$ and/or $\mu(x_r) = \mu(x)$. Any node of G that is not a duplication node, is a *speciation* node.

Take a binary tree T, labeled by the LCA mapping, where there exists exactly one leaf labeled by each gene in Γ, and a function $g : \Gamma \mapsto \Sigma$ indicating the species where each gene is found. A *duplication-free restriction* $D(T)$ of a tree T is obtained by removing either T_{x_ℓ} or T_{x_r} for each duplication node x, along with x, and if x is not the root, joining the parent of x and the remaining child by a new edge. Each duplication-free restriction $D(T)$ can be considered to be a copy of a species tree S, in which case each loss leaf u corresponds subtree S_u of the species tree that is missing in $D(T)$.

A duplication-free restriction $D(T)$ *agrees* with a species tree S iff relabeling each leaf l of $D(T)$ by $g(l)$, and replacing each loss leaf u in $D(T)$ with the subtree S_u, results in a tree isomorphic to S.

Definition 4.2 (consistent)**.** *Take a species tree S and a rooted binary tree T where there exists exactly one leaf labeled by each gene in Γ, and all other leaves are labeled as losses. T is said to be* consistent *with S iff every duplication-free restriction of T agrees with S.*

Definition 4.3 (history)**.** *A* history *H is a rooted binary tree uniquely leaf-labeled by a gene set Γ, and function $g : \Gamma \mapsto \Sigma$ (indicating the species where each gene is found) with the following properties:*

1. *Any leaf not labeled by a member of Γ is a loss.*
2. *Each internal node is a duplication or speciation node.*
3. *There exists a species tree S consistent with H.*

Definition 4.4 (reconciliation)**.** *A* reconciliation $R(G, S)$ *between a binary gene tree* G *and a species tree* S *is a history that can be obtained from* G *by inserting loss leaves and labeling internal nodes as speciations or duplications so that it is consistent with* S*.*

The parsimony criteria used to choose among the large set of possible reconciliations are the number of duplications (*duplication cost*), the number of losses (*loss cost*) or both combined (*mutation cost*). The LCA mapping induces a reconciliation $R(G, S)$ between G and S, where an internal node x of G leads to a duplication node in R if and only if x is a duplication node of G with respect to S. Moreover, $R(G, S)$ is a reconciliation that minimizes the duplication, loss, and mutation costs [26, 69].

In the rest of this paper, the *cost* of a reconciliation refers to its mutation cost.

4.3.2 Problem statement

Given a binary species tree S and a non-binary gene tree G, we seek out a full resolution of G leading to a reconciliation of minimum mutation cost. We formally define the notion of a resolution of G as being a reconciled refinement of G.

Definition 4.5 (Resolution)**.** *A tree* $R(G, S)$ *is a* resolution *of* G *with respect to* S *if and only if* $R(G, S)$ *is a reconciliation between a binary refinement* $B(G)$ *of* G*, and* S*.*

We are now ready to state our optimization problem.

Minimum Resolution:
Input: A binary species tree S and a non-binary gene tree G.
Output: A *Minimum Resolution* of G with respect to S (or simply a *Minimum Resolution of* G if there is no ambiguity on S), e.g. a resolution of G with respect to S of minimum mutation cost.

We first show that each polytomy of G can be resolved independently.

Theorem 4.6. *Let $\{G_{x_i},$ for $1 \leq i \leq p\}$ be the set of subtrees of G rooted at the p children $\{x_i,$ for $1 \leq i \leq p\}$ of the root of G. Let $R_{min}(G_{x_i}, S)$ be a minimum resolution of G_{x_i} w.r.t. S. Let G' be the tree obtained from G by replacing each G_{x_i} by $R_{min}(G_{x_i}, S)$. Then a minimum resolution of G' is a minimum resolution of G.*

Proof. This statement was proved by Chang and Eulenstein in [24], which led them to a dynamic programming algorithm with running-time complexity $O(|S| \times |G|^2)$, for the MINIMUM RESOLUTION problem. The reader interested in this proof might refer to Chang's MSc. Thesis written in 2006. □

It follows from Theorem 4.6 that a minimum resolution of G can be obtained by a depth-first procedure that solves each polytomy G_x iteratively, for each internal node x of G. At each step, whether the children of the polytomy G_x are internal nodes or leaves of G, they are treated as leaves of the polytomy and we refer to each leaf l by its label $\mu(l)$.

In the next section, we consider G as a polytomy whose leaves are labeled (not uniquely) by nodes of S. Furthermore, as the subtrees S_x of S such that $V(S_x) \setminus \{x\}$ has an empty intersection with $\mathcal{S}(G)$, will never be considered in the resolution of G, we can ignore them. We say that S is a *species tree linked to the polytomy* G if and only if any internal node of S has a descendant included in $\mathcal{S}(G)$ and the root of S is the lowest common ancestor of $\mathcal{S}(G)$. For example, in Figure 4.2, S is a species tree linked to the polytomy G.

4.4 Method

In this section, we consider G to be a polytomy whose leaves are labeled (not uniquely) by nodes of a species tree S. Notice that a leaf labeled x actually represents a whole subtree of the considered gene tree, which has already been resolved, and thus is consistent with $S_{\mu(x)}$. We assume that

S is a species tree linked to G. We describe an approach for computing a minimum resolution $R(G, S)$ of G based on the observation that for any node x in $R(G, S)$, all nodes on a path from x to a leaf in $R(G, S)$ will map to a node that is on a path from $\mu(x)$ to a leaf of S. Thus, we decompose the computation of a minimum resolution of G according to a depth-first traversal of the nodes of S; for each node s of S we consider the cost of having k maximal subtrees of $R(G, S)$ whose roots map to s. For example, Figures 4.2c and 4.2d represent two such partial resolutions where there are three maximal subtrees whose roots map to e. Given, for all k, the minimum cost of a so-called "k-partial resolution" corresponding to a node s, we show how to compute the cost of a partial resolution corresponding to the parent of s. Clearly a solution of the Minimum Resolution problem is a minimum 1-partial resolution of G at the root of S.

4.4.1 Partial resolutions

Let s be a node of S. The restriction of G by node s, denoted $G_{/s}$, is the tree obtained from G by removing the set of leaves $\mathcal{L}_s$ whose labels are not in S_s.

Definition 4.7 (partial resolution)**.** *Let s be a node of S. A partial resolution $P(G, S, s)$ of G at s is a polytomy on a set $\mathcal{F}_s \cup \mathcal{L}_s$, where $\mathcal{F}_s$ is obtained from a resolution $R(G_{/s}, S)$ as follows: $\mathcal{F}_s$ is a forest of subtrees of $G_{/s}$, rooted at nodes labeled s, partitioning the set of leaves of $G_{/s}$ (i.e. each leaf of $G_{/s}$ is in a unique tree of the forest).*

The cost of a partial resolution $P(G, S, s)$ on a set $\mathcal{F}_s \cup \mathcal{L}_s$ is the sum of the cost of all the trees (reconciliations) of $\mathcal{F}_s$. See Figure 4.2 for an example.

Definition 4.8 (k-partial resolution)**.** *Let s be a node of S. A* k-partial resolution *$P^k(G, S, s)$ of G at s is a partial resolution of G at s on a set $\mathcal{F}_s \cup \mathcal{L}_s$ with exactly k trees in $\mathcal{F}_s$.*

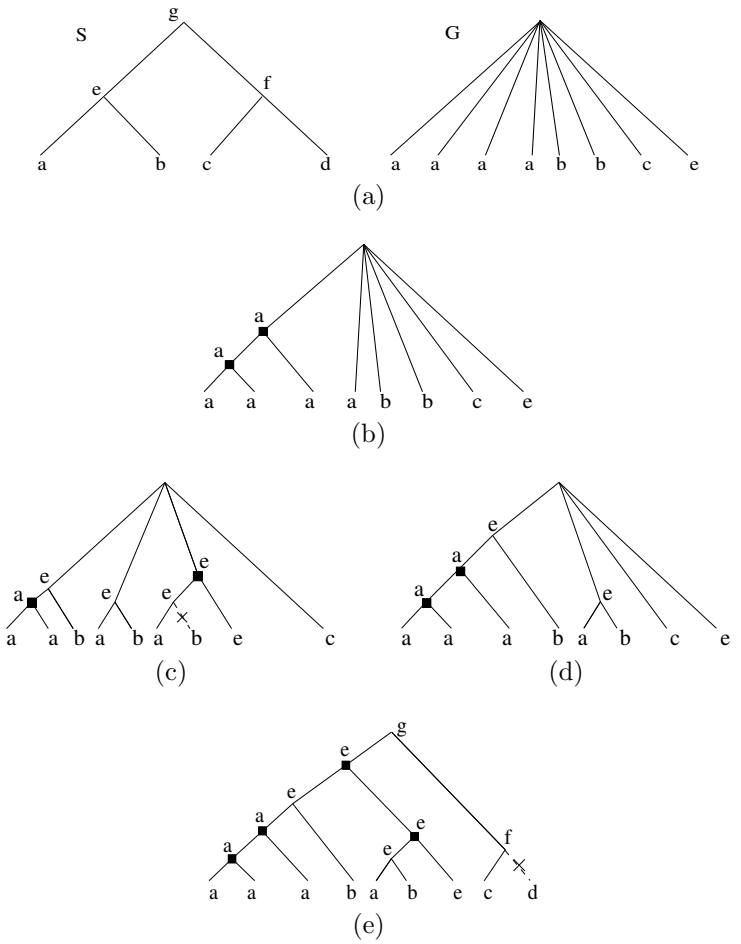

Figure 4.2: (a) A species tree S and a polytomy G; (b) A 2-partial resolution of G at a of cost 2 (2 duplications); (c) A 3-partial resolution of G at e of cost 3 (2 duplications and one loss); (d) A 3-partial resolution of G at e of cost 2 (2 duplications); (e) A full resolution of G with minimum cost (4 duplications, 1 loss).

For example, the tree G in Figure 4.2 is itself a 4-partial resolution of G at a, whereas the tree (b) is a 2-partial resolution of G at a, and (c) and (d) are two different 3-partial resolutions of G at e.

Notation 4.9. *For any integer $k \geq 1$, we denote by $M_{s,k}$ the minimum cost of a k-partial resolution of G at s. We also denote by M_s the vector $(M_{s,k})_{k\geq 1}$.*

A solution for the Minimum Resolution problem is a resolution of G with cost $M_{root(S),1}$. In this section, we describe an algorithm that computes $M_{root(S),1}$ based on the costs $M_{s,k}$ of all partial resolutions of G over all k and s. Before giving a recursive formulation of $M_{s,k}$, we need to introduce a subset of k-partial resolutions, leading to an intermediate cost $C_{s,k}$ for internal nodes s, which can be computed directly from k-partial resolutions corresponding to the children of s.

Definition 4.10 (k-speciation resolution)**.** *Let s be an internal node of S. A k-partial resolution $P^k(G, S, s)$ of G at node s is a* k-speciation resolution *of G at s if and only if each node of $P^k(G, S, s)$ labeled s is a speciation or a leaf.*

A k-speciation resolution at s contains no duplication node nor loss leaf labeled s. In Figure 4.2, the tree G is a 4-speciation resolution of G at node a, while the tree (d) is a 3-speciation resolution at e. Neither the 2-partial resolution of G at a (b) nor the 3-partial resolutions of G at e (c) is a speciation resolution, as in the first case the left-most child of the root labeled a is a duplication node, while in the second case the right-most child of the root labeled e is a duplication node.

Notation 4.11. *For any node s of S, we denote by $nb(s)$ the number of leaves of G labeled s.*

Note that there is no k-speciation resolution for $k \leq nb(s)$. Indeed, as G has $nb(s)$ leaves labeled s, any speciation resolution of G has at least $nb(s)$

speciation nodes labeles s, and thus $k \geq nb(s)$. Moreover, as S is a species tree linked to G, at least one descendant of the internal node s of S should be a leaf of G, and thus any partial resolution at s needs to have at least one additional speciation node labeled s.

Notation 4.12. *For any internal node s of S and any integer $k > nb(s)$, we denote by $C_{s,k}$ the cost of a minimum k-speciation resolution of G at s. For technical reasons, we set $C_{s,k} = \infty$ for $1 \leq k \leq nb(s)$. We denote by C_s the vector $(C_{s,k})_{k \geq 1}$.*

4.4.2 A recursive formulation

There is an infinite range of values of k for which $M_{s,k}$ and $C_{s,k}$ correspond to valid resolutions. However, the following remark is easy to validate and implies that, for some input, we need only consider a fixed-size table of values.

Remark 4.13. *There exists a value $n \in \mathbb{N}$ such that $M_{s,k} < M_{s,n}$, for any node s of S and any integer $0 < k < n$.*

The intuition behind this remark is that when k is large enough a k-partial resolution would contain too many losses, so could never be part of an optimal solution.

The following lemma exhibits a relationship between two entries of vector M_s.

Lemma 4.14. *For any node s of S and any integers $k, i \geq 1$, we have $M_{s,k} \leq M_{s,i} + |k - i|$.*

Proof. Let $P^i(G, S, s)$ be an i-partial resolution at s of cost $M_{s,i}$. If $i < k$, inserting $k - i$ losses of s at the root of $P^i(G, S, s)$ gives us a k-partial resolution of cost $M_{s,i} + k - i$. If $i > k$, joining $i - k + 1$ subtrees rooted at s in $P^i(G, S, s)$ (by duplication nodes) gives us a k-partial resolution of cost $M_{s,i} + i - k$. Both cases imply that $M_{s,k} \leq M_{s,i} + |k - i|$. □

We use Lemma 4.14 to prove the main recurrence defining $C_{s,k}$ and $M_{s,k}$.

Theorem 4.15. *Let s be a node of S and $1 \leq k \leq n$.*

1. *If s is a leaf of S, then $M_{s,k} = |k - nb(s)|$;*
2. *Otherwise, let s_l and s_r be the two children of s in S. Then,*
   (a) $C_{s,k} = M_{s_\ell,k-nb(s)} + M_{s_r,k-nb(s)}$ *if* $k > nb(s)$ *(∞ otherwise);*
   (b) $M_{s,k} = \min\left(C_{s,k}, \min_{1\leq i\leq n}(M_{s,i} + |k-i|)\right)$

Proof. Let $P^k(G,S,s)$ be a minimum k-partial resolution of G at node s of cost $M_{s,k}$. Suppose $P^k(G,S,s)$ is defined on the set of nodes $\mathcal{F}_s \cup \mathcal{L}_s$.

1. Suppose s is a leaf of S. If $k = nb(s)$, then each tree in $\mathcal{F}_s$ is a single node labeled s, with reconciliation cost 0, and thus $M_{s,k} = 0 = |k - nb(s)|$. If $k < nb(s)$ (respectively $k > nb(s)$), then at least $nb(s) - k$ duplication nodes (respec. $k - nb(s)$ losses) should be present in the trees of $\mathcal{F}_s$. As the trees of $\mathcal{F}_s$ are part of an optimal k-partial resolution, the number of duplications (losses) should be exactly $nb(s) - k$ ($k - nb(s)$), and thus $M_{s,k} = |k - nb(s)|$.

2. Otherwise, s is an internal node of S.
2(a) Let $P^k(G,S,s)$ be a k-speciation resolution of G at node s of minimum cost $C_{s,k}$. Since none of the trees in $\mathcal{F}_g$ are rooted at a duplication node nor labeled as a loss, there must be exactly $k - nb(s)$ trees in $\mathcal{F}_s$ that are rooted at speciation nodes labeled by s. Any such node must have one child labeled s_ℓ and one child labeled s_r. Since we are going through each node of S in a depth-first manner, we assume that the values of M_{s_ℓ} and M_{s_r} have been computed. The result follows from the fact that $M_{s_\ell,k-nb(s)}$ (resp. $M_{s_r,k-nb(s)}$) gives the optimal configuration yielding $k - nb(s)$ trees rooted at nodes labeled s_ℓ (resp. s_r).

2(b) If $P^k(G, S, s)$ is a k-speciation resolution of G at s, then clearly $M_{s,k} = C_{s,k}$. Otherwise, let k' be the number of trees rooted at duplication nodes of $\mathcal{F}_s$. For positive k', if each of the two children of those duplication nodes were taken as the roots of two new trees, then we would have a forest of $i' = k + k'$ trees. This gives us $M_{s,k} \geq M_{s,i'} + k' = M_{s,i'} + |k - i'|$. If $k' = 0$ (and $P^k(G, S, s)$ is not a k-speciation resolution), then $\mathcal{F}_s$ must have, say k'', trees corresponding to losses. Consider the $i'' = k - k''$ trees of $\mathcal{F}_s$ that are not losses. This gives us $M_{s,k} \geq M_{s,i''} + k'' = M_{s,i''} + |k - i''|$ as well. Therefore $M_{s,k} \geq \min_i(M_{s,i} + |k - i|)$. On the other hand, Lemma 4.14 gives us $M_{s,k} \leq \min_i(M_{s,i} + |k - i|)$. □

4.4.3 A dynamic programming approach

The recurrence 2.b in Theorem 4.15 induces a circular argument for computing the entries of M_s, as for two different constants k and i, $M_{s,k}$ may be computed from $M_{s,i}$, which in turn may be computed from $M_{s,k}$. In other words, the recurrences of Theorem 4.15 cannot be used directly in a dynamic programming algorithm for the computation of $C_{s,k}$ and $M_{s,k}$. The rest of this section focuses on reformulating recurrence 2.b. We start by giving two important properties relating M_s to C_s.

Lemma 4.16. *For an internal node s of S, there exists at least one k such that $M_{s,k} = C_{s,k}$.*

Proof. Let $P^k(G, S, s)$ be a k-partial resolution of G at node s of cost $M_{s,k}$, defined on the set $\mathcal{F}_s \cup \mathcal{L}_g$. Assume that $M_{s,k} \neq C_{s,k}$ for all k. This implies $M_{s,k} < C_{s,k}$ for all k. Consider the subforest $\mathcal{F}'_s$ consisting of the j maximal subtrees of $\mathcal{F}_s$ that are rooted at speciation nodes; $\mathcal{F}'_s$ defines a j-speciation resolution with cost $C' < M_{s,k}$. Since $C_{s,j} \leq C'$, we have $C_{s,j} < M_{s,k}$. But $M_{s,j} < C_{s,j}$, so in general, for any k there exists a j such that $M_{s,j} < M_{s,k}$, a contradiction since the minimum value in M_s must occur in within a finite range (by Remark 4.13). □

Theorem 4.15 shows that for an internal node s of S, $M_{s,k}$ can be computed from $C_{s,k}$, or from some other value in M_s. However, we have not characterized how to easily discern which case will be used, and we have no information about which i gives $M_{s,i} = C_{s,i}$. The following lemma addresses this matter by narrowing the possibilities.

Lemma 4.17. *For some internal node s of S and integer $k \geq 1$, if $M_{s,k} \neq C_{s,k}$ then there exists an i such that $M_{s,i} = C_{s,i}$ and $M_{s,k} = M_{s,i} + |k - i|$.*

Proof. By the recurrence 2(b) of Theorem 4.15, if $M_{s,k} \neq C_{s,k}$, then we should have $M_{s,k} = M_{s,i} + |k - i|$ for some i. If $M_{s,i} = C_{s,i}$, then the lemma is verified. Otherwise, $M_{s,i} = M_{s,h} + |i - h|$ for some h. This gives $M_{s,k} = M_{s,h} + |k - i| + |i - h| \geq M_{s,h} + |k - h|$. By lemma 4.16 we know that there must be some value α for which $M_{s,k_\alpha} = C_{s,k_\alpha}$, so in general, for some integers k_0 and α we have

$$\begin{aligned} M_{s,k_0} &= M_{s,k_\alpha} + |k_\alpha - k_{\alpha-1}| + |k_{\alpha-1} - k_{\alpha-2}| + \cdots + |k_2 - k_1| + |k_1 - k_0| \\ &= M_{s,k_\alpha} + \sum_{i=1}^{\alpha} |k_i - k_{i-1}| \\ &\geq M_{s,k_\alpha} + |k_\alpha - k_0| = C_{g,k_\alpha} + |k_\alpha - k_0|. \end{aligned}$$

Lemma 4.14 gives the complementary bound $M_{s,k_0} \leq C_{s,k_\alpha} + |k_\alpha - k_0|$, so equality holds. □

Lemma 4.17 allows us to rewrite the recurrence 2(b) of Theorem 4.15 as follows:

$$M_{s,k} = \min_{nb(s)<i\leq n} C_{s,i} + |k - i| \qquad (Eq.1)$$

With this new formulation of recurrence 2(b), Theorem 4.15 leads to a cubic-time dynamic programming algorithm for the computation of the cost of a solution of the Minimum Resolution problem. Indeed, let the height of a

node s of S be the maximum number of nodes in a path from s to a leaf of S. Consider an ordering $s_1, s_2 \cdots s_p$ of the nodes of S by increasing height, where $p = |S|$. In other words, leaves are listed before nodes of height 1, etc. In particular $s_p = root(S)$. Consider the tables M and C of $|S|$ lines, where each line i of M and C corresponds respectively to the vectors M_{s_i} and C_{s_i}. The table C is defined only for lines $i > L(S)$. We first compute the $L(S)$ first lines of M in $O(n)$ steps using recurrence (1) of Theorem 4.15. Then, for each line i, we successively compute C_{s_i} and M_{s_i} for increasing values of i, by using the recurrences (2) a. and b. of Theorem 4.15. Each line representing C_{s_i} is computed in time $O(n)$, while each line representing M_{s_i} is computed in $O(n^2)$ steps, leading to an $O(n^2|S|)$ algorithm for filling the two tables. The final result (cost of a solution of the Minimum Resolution problem) is just $M_{s_p,1}$. An example is given in Figure 4.3.

4.4.4 A linear-time approach

We show in this section that the recurrence (2)b of Theorem 4.15 can further be simplified in a way leading to a constant time update for each M_s vector according to the M_{s_l} and M_{s_r} vectors. This implies a linear time algorithm for the computation of $M_{root(S),k}$.

We first show that $M_{s,k} = C_{s,k}$ when $C_{s,k}$ is the minimum value among the entries of C_s.

Lemma 4.18. *For k such that $C_{s,k} = \min_{nb(s)<i\leq n} C_{s,i}$, we have $M_{s,k} = C_{s,k}$.*

Proof. If $M_{s,k} \neq C_{s,k}$, then by Lemma 4.17, there must exist an i such that $M_{s,k} = M_{s,i} + |k - i|$ and $M_{s,i} = C_{s,i}$. This implies that $C_{s,i} < C_{s,k}$, contradicting the minimality of $C_{s,k}$. □

The key observation allowing a constant-time computation of any entry in C_s and M_s is that these vectors can be seen as two functions with a cup shape. The following definition formally introduces the notion of a "cup function".

Definition 4.19 (cup function)**.** *A* cup function *is a convex piecewise linear function $m()$ which, for a minimum value $\gamma_m \in \mathbb{Z}$ and two* breakpoints *$m_1, m_2 \in \mathbb{N}$, is strictly decreasing linearly for $x < m_1$, equal to γ_m when $m_1 \leq x \leq m_2$, and strictly increasing linearly when $x > m_2$. It can be written as*

$$m(x) = \begin{cases} \gamma_m + m_1 - x + \mathcal{P}(x) & \text{if } x < m_1 \\ \gamma_m & \text{if } m_1 \leq x \leq m_2 \\ \gamma_m + x - m_2 + \mathcal{Q}(x) & \text{if } x > m_2 \end{cases}$$

where $\mathcal{P} : \mathbb{N} \to \mathbb{Z}$ is non-increasing and $\mathcal{Q} : \mathbb{N} \to \mathbb{Z}$ is non-decreasing.

We say the function $m(x)$ is a simple cup function *iff $\mathcal{P}(x) = \mathcal{Q}(x) = 0$ for all x. Roughly speaking, a simple cup function viewed from left to right has a slope of -1, a plateau of minimum values, and a slope of 1.*

Assume for now that M_s can be associated with a simple cup function $m()$ such that $M_{s,k} = m(k)$ for $1 \leq k \leq n$. Recall that the values of C_s are obtained by adding the values of M_{s_ℓ} and M_{s_r} (recurrence 2(a) of Theorem 4.15). We show that C_s can be associated with a cup function by proving that the addition of two simple cup functions yields a cup function.

Lemma 4.20. *If $\ell()$ and $r()$ are two simple cup functions, then the function $m()$ defined by $m(k) = \ell(k) + r(k)$, is a cup function.*

Furthermore, if ℓ_1, ℓ_2 and r_1, r_2 respectively denote the breakpoints of $\ell()$ and $r()$, and γ_ℓ, γ_r respectively denote the minimum values of $\ell()$ and $r()$, then the breakpoints m_1, m_2 and minimum value γ_m of $m()$ are computed according to the following table:

Condition	γ_m	m_1	m_2
If $\ell_1 < r_1, \ell_2 < r_1$	$\gamma_\ell + \gamma_r + r_1 - \ell_2$	ℓ_2	r_1
If $\ell_1 < r_1, r_1 \leq \ell_2 \leq r_2$	$\gamma_\ell + \gamma_r$	r_1	ℓ_2
If $\ell_1 < r_1, \ell_2 > r_2$	$\gamma_\ell + \gamma_r$	r_1	r_2
If $r_1 \leq \ell_1 \leq r_2, r_1 \leq \ell_2 \leq r_2$	$\gamma_\ell + \gamma_r$	ℓ_1	ℓ_2
If $r_1 \leq \ell_1 \leq r_2, \ell_2 > r_2$	$\gamma_\ell + \gamma_r$	ℓ_1	r_2
If $\ell_1 > r_2, \ell_2 > r_2$	$\gamma_\ell + \gamma_r + \ell_1 - r_2$	r_2	ℓ_1

Proof. The complete proof of this lemma is given in Appendix 4.5. Moreover, a more general version of this lemma has already been proven by Csuros in [36], where it is shown that the sum of an arbitrary number of cup functions yields a cup function □

The following theorem states that M_s and C_s can be associated with two cup functions with the same breakpoints m_1, m_2 and minimum value γ_m. Moreover, M_s is associated with a simple cup function. For example, each line of the table of Figure 4.3 can be rewritten as a cup function, with the breakpoint and minimum values indicated in vectors m_1, m_2 and γ_m.

Theorem 4.21. *For any node s of S, there exists a simple cup function $m()$ with breakpoints $m_1, m_2 \geq 1$ and minimum value γ_m, such that $M_{s,k} = m(k)$ for $1 \leq k \leq n$.*

Furthermore, if s is an internal node of S, there exists a cup function $c()$ with the same breakpoints $m_1, m_2 > nb(s)$ and same minimum value γ_m, such that $C_{s,k} = c(k)$ for $nb(s) < k \leq n$.

Proof. We prove the theorem by induction over the nodes visited in a postorder traversal of S.

Base Case: If s is a leaf and $nb(s) > 0$, let $m(k) = |k - nb(s)|$. It is clear that $m(k)$ is a simple cup function with breakpoints $m_1 = m_2 = nb(s)$ and

$\gamma_m = 0$. If $nb(s) = 0$, let $m(k)$ be a simple cup function with breakpoints $m_1 = m_2 = 1$ and minimum value $\gamma_m = 1$. We have $m(k) = |k - nb(s)|$ for $k \geq 1$. Then from Theorem 4.15.1, both cases give $M_{s,k} = m(k)$ for $1 \leq k \leq n$.

Induction Step: If s is an internal node, then from the inductive hypothesis, there exist two simple cup functions $\ell(), r()$ such that $M_{s_\ell,k} = \ell(k)$ and $M_{s_r,k} = r(k)$ for $1 \leq k \leq n$. Let $f(k) = \ell(k - nb(s)) + r(k - nb(s))$. By Lemma 4.20, $f()$ is a cup function. Let f_1, f_2 be the breakpoints of $f()$ and γ_f its minimum value. From part *(2a)* of Theorem 4.15 we have $C_{s,k} = M_{s_\ell,k-nb(s)} + M_{s_r,k-nb(s)}$, implying that $C_{s,k} = f(k)$ for $nb(s) < k \leq n$. However, it is possible that $f_1 \leq nb(s)$ or $f_2 \leq nb(s)$, making the theorem statement invalid.

Let $c()$ be another cup function with breakpoints $m_1 = \max(f_1, nb(s) + 1), m_2 = \max(f_2, nb(s)+1)$ and minimum value $\gamma_m = f(m_2) = \min_{nb(s)<k\leq n} C_{s,k}$. It can be verified that $c(k) = f(k)$ when $nb(s) < k \leq n$ and thus, $C_{s,k} = c(k)$ for $nb(s) < k \leq n$. From this, we have

$$C_{g,k} = \begin{cases} \infty & \text{if } k \leq nb(s) \\ \gamma_m + m_1 - k + \mathcal{P}(k) & \text{if } nb(s) < k < m_1 \\ \gamma_m & \text{if } m_1 \leq k \leq m_2 \\ \gamma_m + k - m_2 + \mathcal{Q}(k) & \text{if } k > m_2 \end{cases}$$

for $1 \leq k \leq n$.

By Lemma 4.18, we know that $M_{s,k} = C_{s,k} = \gamma_m$ for $m_1 \leq k \leq m_2$.

If $k < m_1$, we show that $M_{s,k} = \gamma_m + m_1 - k$. From the equation (Eq.1), we have $M_{s,k} = \min_j(C_{s,j} + |k - j|)$. If $j > m_1$, then by the definition of $C_{s,j}$ given above, we have $C_{s,j} \geq \gamma_m$ and thus $C_{s,j}+j-k \geq \gamma_m+m_1-k$. If $j < m_1$, then $C_{s,j}+|k-j| \geq \gamma_m+m_1-j+|k-j| \geq \gamma_m+m_1-k$ since we can reformulate this inequality as $|k - j| \geq j - k$. Therefore, $C_{s,j} + |k - j|$ has the minimum

value when $j = m_1$ and it follows that $M_{s,k} = C_{s,m_1} + m_1 - k = \gamma_m + m_1 - k$ when $k < m_1$.

If $k > m_2$, we show that $M_{s,k} = \gamma_m + k - m_2$. From the (Eq.1), we have $M_{s,k} = \min_j(C_{s,j} + |k - j|)$. If $j < m_2$, we have $C_{s,j} \geq \gamma_m$ and thus $C_{s,j} + k - j \geq \gamma_m + k - m_2$. If $j > m_2$, then $C_{s,j} + |k - j| \geq \gamma_m + j - m_2 + |k - j| \geq \gamma_m + k - m_2$ since we can reformulate this inequality as $|k - j| \geq k - j$. Therefore, $C_{s,j} + |k - j|$ is minimum when $j = m_2$ and it follows that $M_{s,k} = C_{s,m_2} + k - m_2 = \gamma_m + k - m_2$ when $k > m_2$.

The three cases verify that M_s can be associated with a cup function. □

Denote by $m_{1,s}, m_{2,s}$ and γ_s the breakpoints and minimum value of the cup function associated with M_s (and thus from Theorem 4.21 also with C_s). Theorem 4.15.(1) allows us to compute $m_{1,s}$, $m_{2,s}$ and γ_s for any leaf s of S. Finally, Theorem 4.15.(2a) and Lemma 4.20 allow us to compute, in constant time, the breakpoints $m_{1,s}$, $m_{2,s}$ and minimum value γ_s associated with C_s and M_s, given those associated with M_{s_l} and M_{s_r}.

Stated differently, let $s_1, s_2, \cdots s_p$ be the ordering of the nodes of S defined at the end of Section 4.4.3, and consider the three vectors $m_1 = (m_{1,s_i})_{1 \leq i \leq s}$, $m_2 = (m_{2,s_i})_{1 \leq i \leq s}$ and $\gamma = (\gamma_{s_i})_{1 \leq i \leq s}$. Then each entry of each of these vectors can be computed in constant-time. Theorem 4.21 ensures that these vectors allow us to completely define the simple cup function M_s. This leads to an $O(max(|G|, |S|))$ algorithm for computing any value $M_{s,k}$, and in particular the cost $M_{s_p,1}$ of a solution of the minimum Resolution problem.

In Algorithm CupValues(s), we detail the steps required to compute $m_{1,s}, m_{2,s}$ and γ_s for a given node s. Lines 1 to 6 follow from the Theorem 4.21's base case proof. Line 9 follows from Theorem 4.15.(2a) and Lemma 4.20 (discussion above). Line 10 is a correction needed because the table in Lemma 4.20 gives the result for the addition of two simple cup functions for the same value of k, e.g. $c(k) = \ell(k) + r(k)$. Since $C_{s,k} = M_{s_\ell, k-nb(s)} + M_{s_r, k-nb(s)}$, we need to shift the obtained breakpoints by adding $nb(s)$ to them. Lines 11 and 12 ensure that $m_{1,s}, m_{2,s} > nb(s)$ and

that γ_s is the minimum entry in C_s, as stated in Theorem 4.21.

Algorithm 2 $CupValues(s)$

if s is a leaf **then**
  if $nb(s) > 0$ **then**
    $m_{1,s} := nb(s); m_{2,s} := nb(s); \gamma_s := 0;$
  else
    $m_{1,s} := 1; m_{2,s} := 1;\ \gamma_s := 1;$
  end if
else
  Let s_ℓ, s_r be the two children of s:
  Compute $m_{1,s}, m_{2,s}$ and γ_s using the children values $m_{1,s_\ell}, m_{2,s_\ell}, m_{1,s_r}, m_{2,s_r}, \gamma_\ell, \gamma_r$ and the table given in Lemma 4.20;
  $m_{1,s} := m_{1,s} + nb(s);\ m_{2,s} := m_{2,s} + nb(s);$
  If $m_{1,s} \leq nb(s)$ then $m_{1,s} := nb(s) + 1;$
  If $m_{2,s} \leq nb(s)$ then $m_{2,s} := nb(s) + 1$ and $\gamma_s := M_{s_\ell,1} + M_{s_r,1};$
end if

4.4.5 Constructing an optimal resolution

Starting with $s = root(S)$ and $k = 1$, we recursively compute the number of losses and duplications required for each node s of S in an optimal reconciliation, based on partial resolutions at s_l and s_r, for s_l and s_r being the children of s. The algorithm presented in this section is based on the following result, which is a corollary of Theorem 4.21.

Corollary 4.22. *Let s be a node of S with children s_ℓ and s_r, and $P^k(G, S, s)$ be a minimum k-partial resolution at s defined on the set $\mathcal{F}_g \cup \mathcal{L}_g$ for $1 \leq k \leq n$.*

1. *If $M_{s,k} = M_{s_\ell,k-nb(s)} + M_{s_r,k-nb(s)}$, then the k roots of $\mathcal{F}_s$ are all speciation nodes. Otherwise,*

2. *either $M_{s,k} = \gamma_s + k - m_{2,s}$, in which case $P^k(G, S, s)$ has $k - m_{2,s}$ loss leaves labeled s,*

	1	2	3	4	5	6	m_1	m_2	γ_m
M_a	3	**2**	1	0	1	2	4	4	0
M_b	1	**0**	1	2	3	4	2	2	0
M_c	**0**	1	2	3	4	5	1	1	0
M_d	**1**	2	3	4	5	6	1	1	1
C_e	∞	4	2	2	2	4	3	5	2
M_e	**4**	3	2	2	2	3	3	5	2
C_f	1	3	5	7	9	11	1	1	1
M_f	**1**	2	3	4	5	6	1	1	1
C_g	5	5	5	6	7	9	1	3	5
M_g	**5**	5	5	6	7	8	1	3	5

Figure 4.3: An illustration of the algorithms for the gene tree G and species tree S of Figure 4.2(a). The cost of a most parsimonious resolution of G is $M_{g,1} = 5$. The gray cells are those considered by Algorithm DupLoss for computing the Dup and $Loss$ vectors, the values in bold being the first ones evaluated by the algorithm on a given row. The obtained values are $Dup(e) = 2, Dup(a) = 2, Loss(d) = 1$ and $Dup(s) = Loss(s) = 0$ for any other node s. The corresponding resolution is given in Figure 4.2(e).

3. *or $M_{s,k} = \gamma_s + m_{1,s} - k$, in which case $P^k(G, S, s)$ has $m_{1,s} - k$ duplications labeled s.*

Proof. In the first case, $M_{s,k} = C_{s,k}$, indicating that the optimal k-partial resolution at s is a k-speciation resolution. case, taking the $m_{2,s}$-speciation resolution at s of cost γ_s and adding $k - m_{2,s}$ loss leaves labeled s yields a k-partial resolution at s with minimum score $\gamma_s + k - m_{2,s}$. In the third case, taking the $m_{1,s}$-speciation resolution at s of cost γ_s and creating $m_{1,s} - k$ duplications labeled s yields a k-partial resolution at s with minimum score $\gamma_s + m_{1,s} - k$. □

Algorithm DupLoss(s, k) computes the values $Dup(s)$ and $Loss(s)$, being respectively the number of duplications labeled s and the number of loss leaves labeled s in a minimum resolution of G. Starting with a call to Algorithm DupLoss$(root(S), 1)$, the output is a pair $(Dup(s), Loss(s))$ for each

node s of S. From these values, it is easy to reconstruct the corresponding solution $R(G, S)$ of the Minimum Resolution problem.

Algorithm 3 $DupLoss(s, k)$

if s is a leaf and $k \geq nb(s)$ **then**
  $Dup(s) := 0; Loss(s) := k - nb(s)$;
else if s is a leaf and $k < nb(s)$ **then**
  $Dup(s) := nb(s) - k; Loss(s) := 0$;
else if $k - nb(s) > 0$ and $M_{s,k} = M_{s_\ell,k-nb(s)} + M_{s_r,k-nb(s)}$ **then**
  $Dup(s) := 0; Loss(s) := 0$;
  $DupLoss(s_\ell, k - nb(s))$; $DupLoss(s_r, k - nb(s))$;
else if $k < m_{1,s}$ **then**
  $Dup(s) := m_{1,s} - k; Loss(s) := 0$;
  $DupLoss(s_\ell, m_{1,s} - nb(s))$; $DupLoss(s_r, m_{1,s} - nb(s))$;
else if $k > m_{2,s}$ **then**
  $Dup(s) := 0; Loss(s) := k - m_{2,s}$;
  $DupLoss(s_\ell, m_{2,s} - nb(s))$; $DupLoss(s_r, m_{2,s} - nb(s))$;
end if

Once the vectors Dup and $Loss$ have been computed, an optimal resolution of G can be constructed easily, knowing $nb(g)$ for each node, and knowing how many of these nodes are joined under duplication or speciation and how many are inserted as losses.

4.5 Discussion

We have developed an algorithm for constructing the most parsimonious reconciliation, in term of number of duplications + losses, of a polytomy G with a binary species tree S, running in time $O(|S|)$. It naturally leads to an $O(|G| \times |S|)$ algorithm for the reconciliation of a gene tree with an arbitrary number of polytomies. Indeed, it is sufficient to traverse the tree in a depth-first manner, and resolve each polytomy at a time. Interestingly, we can find an example of trees G and S leading to a reconciliation of size $|G| \times |S|$. Indeed, let $\Sigma = \{1, 2, \cdots s\}$, and consider the species tree S over

Σ to be a caterpilar tree $(1, 2, \cdots s)$ with leaves ordered from 1 to s, where $s = |S|$. Consider the gene tree G to be the caterpilar tree $((l_1, r_1), \cdots (l_g, r_g))$ composed of g cherries (l_1, r_s), where the l leaves are labeled 1, and the r leaves are labeled s. We have $g = |G|$. Then clearly a most parsimonious reconciliation of G and S is one with $s-2$ leaves inserted in each cherry of G, which leads to a tree of size $|G| \times |S|$. Therefore, the algorithm is optimal for our considered Minimum Resolution problem. It is likely however that finding the mutation cost of an optimal reconciliation, without displaying the actual reconciliation, can be done in linear-time.

Appendix

Proof of lemma 4.20

Proof. Let

$$\ell(k) = \begin{cases} \gamma_\ell + \ell_1 - k & \text{if } k < \ell_1 \\ \gamma_\ell & \text{if } \ell_1 \leq k \leq \ell_2 \\ \gamma_\ell + k - \ell_2 & \text{if } k > \ell_2 \end{cases}$$

$$r(k) = \begin{cases} \gamma_r + r_1 - k & \text{if } k < r_1 \\ \gamma_r & \text{if } r_1 \leq k \leq r_2 \\ \gamma_r + k - r_2 & \text{if } k > r_2 \end{cases}$$

and $m(k) = \ell(k) + r(k)$. The addition of $\ell(k)$ and $r(k)$ yields a function with nine possible cases.

$$m(k) = \ell(k)+r(k) = \begin{cases} \gamma_\ell + \ell_1 - k + \gamma_r + r_1 - k & \text{if } k < \ell_1, k < r_1 \\ \gamma_\ell + \ell_1 - k + \gamma_r & \text{if } k < \ell_1, r_1 \leq k \leq r_2 \\ \gamma_\ell + \gamma_r + r_1 - k & \text{if } \ell_1 \leq k \leq \ell_2, k < r_1 \\ \gamma_\ell + \gamma_r & \text{if } \ell_1 \leq k \leq \ell_2, r_1 \leq k \leq r_2 \\ \gamma_\ell - \ell_2 + \gamma_r + r_1 & \text{if } k > \ell_2, k < r_1 \\ \gamma_\ell + \ell_1 + \gamma_r - r_2 & \text{if } k < \ell_1, k > r_2 \\ \gamma_\ell + \gamma_r + k - r_2 & \text{if } \ell_1 \leq k \leq \ell_2, k > r_2 \\ \gamma_\ell + k - \ell_2 + \gamma_r & \text{if } k > \ell_2, r_1 \leq k \leq r_2 \\ \gamma_\ell + k - \ell_2 + \gamma_r + k - r_2 & \text{if } k > \ell_2, k > r_2 \end{cases}$$

These nine cases are not all compatible with each other depending on ℓ_1, ℓ_2, r_1, r_2. In fact, there are only six possible cases of how these breakpoints cross eachother. If $\ell_1 < r_1$, either $\ell_2 < r_1, r_1 \leq \ell_2 \leq r_2$ or $\ell_2 > r_2$. If $r_1 \leq \ell_1 \leq r_2$, either $r_1 \leq \ell_2 \leq r_2$. If $\ell_1 > r_2$, then $\ell_2 > r_2$. Let $\alpha = \gamma_\ell + \gamma_r$. Then, $m(k)$ can be rewritten as :

If $\ell_1 < r_1, \ell_2 < r_1,$ then

$$m(k) = \begin{cases} \alpha - 2k + \ell_1 + r_1 & \text{if } k < \ell_1, k < r_1 \\ \alpha - k + r_1 & \text{if } \ell_1 \leq k \leq \ell_2, k < r_1 \\ \alpha + r_1 - \ell_2 & \text{if } k > \ell_2, k < r_1 \\ \alpha + k - \ell_2 & \text{if } k > \ell_2, r_1 \leq k \leq r_2 \\ \alpha + 2k - \ell_2 - r_2 & \text{if } k > \ell_2, k > r_2 \end{cases}$$

If $\ell_1 < r_1, r_1 \leq \ell_2 \leq r_2$, then

$$m(k) = \begin{cases} \alpha - 2k + \ell_1 + r_1 & \text{if } k < \ell_1, k < r_1 \\ \alpha - k + r_1 & \text{if } \ell_1 \leq k \leq \ell_2, k < r_1 \\ \alpha & \text{if } \ell_1 \leq k \leq \ell_2, r_1 \leq k \leq r_2 \\ \alpha + k - \ell_2 & \text{if } k > \ell_2, r_1 \leq k \leq r_2 \\ \alpha + 2k - \ell_2 - r_2 & \text{if } k > \ell_2, k > r_2 \end{cases}$$

If $\ell_1 < r_1, \ell_2 > r_2$, then

$$m(k) = \begin{cases} \alpha - 2k + \ell_1 + r_1 & \text{if } k < \ell_1, k < r_1 \\ \alpha - k + r_1 & \text{if } \ell_1 \leq k \leq \ell_2, k < r_1 \\ \alpha & \text{if } \ell_1 \leq k \leq \ell_2, r_1 \leq k \leq r_2 \\ \alpha + k - r_2 & \text{if } \ell_1 \leq k \leq \ell_2, k > r_2 \\ \alpha + 2k - \ell_2 - r_2 & \text{if } k > \ell_2, k > r_2 \end{cases}$$

If $r_1 \leq \ell_1 \leq r_2, r_1 \leq \ell_2 \leq r_2$, then

$$m(k) = \begin{cases} \alpha - 2k + \ell_1 + r_1 & \text{if } k < \ell_1, k < r_1 \\ \alpha - k + \ell_1 & \text{if } k < \ell_1, r_1 \leq k \leq r_2 \\ \alpha & \text{if } \ell_1 \leq k \leq \ell_2, r_1 \leq k \leq r_2 \\ \alpha + k - \ell_2 & \text{if } k > \ell_2, r_1 \leq k \leq r_2 \\ \alpha + 2k - \ell_2 - r_2 & \text{if } k > \ell_2, k > r_2 \end{cases}$$

If $r_1 \leq \ell_1 \leq r_2, \ell_2 > r_2$, then

$$m(k) = \begin{cases} \alpha - 2k + \ell_1 + r_1 & \text{if } k < \ell_1, k < r_1 \\ \alpha - k + \ell_1 & \text{if } k < \ell_1, r_1 \leq k \leq r_2 \\ \alpha & \text{if } \ell_1 \leq k \leq \ell_2, r_1 \leq k \leq r_2 \\ \alpha + k - r_2 & \text{if } \ell_1 \leq k \leq \ell_2, k > r_2 \\ \alpha + 2k - \ell_2 - r_2 & \text{if } k > \ell_2, k > r_2 \end{cases}$$

If $\ell_1 > r_2, \ell_2 > r_2$, then

$$m(k) = \begin{cases} \alpha - 2k + \ell_1 + r_1 & \text{if } k < \ell_1, k < r_1 \\ \alpha - k + \ell_1 & \text{if } k < \ell_1, r_1 \leq k \leq r_2 \\ \alpha - r_2 + \ell_1 & \text{if } k < \ell_1, k > r_2 \\ \alpha + k - r_2 & \text{if } \ell_1 \leq k \leq \ell_2, k > r_2 \\ \alpha + 2k - \ell_2 - r_2 & \text{if } k > \ell_2, k > r_2 \end{cases}$$

Using the breakpoints and minimum values provided by the table in the theorem statement, it can be verified that each of these cases is a cup function. □

4.6 Contributions

Manuel Lafond a conçu l'algorithme. Manuel Lafond, Nadia El-Mabrouk et Krister M. Swenson ont rédigé les preuves de bon fonctionnement de l'algorithme. Nadia El-Mabrouk et Krister M. Swenson ont fait la mise en contexte scientifique du problème et mis en place le contexte formel. Nadia El-Mabrouk a supervisé le déroulement du projet. Tous les auteurs ont participé à la rédaction de l'article.

CHAPITRE 5

GENE TREE CORRECTION GUIDED BY ORTHOLOGY

Manuel Lafond[1] , Magali Semeria[2] , Krister M. Swenson[1 3]
Eric Tannier[2 4] , Nadia El-Mabrouk[1]

Dans ce chapitre, nous décrivons deux algorithmes permettant de corriger un arbre afin qu'il contienne un ensemble donné de relations d'orthologie. Rappelons que deux gènes sont orthologues s'ils descendent d'un gène ancestral ayant subi une spéciation, et paralogues s'il a subi une duplication. Un arbre de gènes G étiqueté par ses événements de spéciation et duplication peut servir à distinguer les orthologues des paralogues, puisque pour deux gènes g_1 et g_2 donnés, il suffit de connaître l'étiquette de $lca_G(g_1, g_2)$. Une façon commune d'inférer l'ensemble des relations d'orthologie/paralogie d'une famille de gènes consiste d'ailleurs à inférer l'arbre de gènes pour la famille, le réconcilier avec l'arbre d'espèces et identifier la relation entre chaque paire de gènes. Cette méthode suppose bien sûr que l'arbre de gènes est correct, et que la parcimonie s'applique au niveau des spéciations et duplications [95]. Notons que dans [145], on propose une inférence d'orthologues et paralogues probabiliste basée sur l'inférence Bayésienne d'une distribution sur les réconciliations.

Dans les deux chapitres suivants, nous nous intéressons plutôt à la direction inverse, c'est-à-dire à obtenir de l'information sur l'arbre de gènes à partir de relations d'orthologie/paralogie. L'idée est que si l'on a un arbre de gènes G à valider et que l'on a accès aux (véritables) relations entre ses gènes, on peut vérifer qu'elles correspondent bel et bien à ce que G dépeint.

[1]DIRO, Université de Montréal, Canada
[2]Laboratoire de Biométrie et Biologie Évolutive, Université Lyon I, France
[3]McGill Center for Bioinformatics, McGill University, Canada
[4]INRIA Grenoble Rhône Alpes, France

Sinon, on sait que G a besoin d'être corrigé.

Il existe d'ailleurs des méthodes d'inférence de relations qui ne nécessitent pas d'arbre. La plus commune est basée sur la similarité de séquences et s'apparente à la méthode de regroupement de gènes en familles homologues décrite dans la section 3.1. Étant donné un ensemble de génomes, on effectue une recherche BLAST "all-against-all" pour retrouver les paires de gènes similaires. Toutefois, on veut ici filtrer les orthologues en ne conservant que les *meilleurs résultats réciproques*, ou MRR (appelés BBH en anglais, pour Bidirectional Best Hits [121, 160]). Pour un gène g dans une espèce s, le meilleur résultat dans une espèce s' est le gène g' de s' donnant le meilleur score BLAST avec g. Si pour g', on a aussi que g est le meilleur résultat dans s, alors g et g' forment un MRR. Le graphe des MRR a un sommet pour chaque gène, et a une arête entre g et g' si et seulement si ils forment un MRR. On peut ensuite utiliser un algorithme de clustering comme MCL (e.g. OrthoMCL [106], proteinortho [105]) ou une méthode ad-hoc (e.g. eggNOG [88]) pour séparer les gènes en groupes d'orthologues. Notons que le logiciel InParanoid [10] permet d'identifier les *paralogues récents* (souvent appelés "in-paralogs").

Ces méthodes n'infèrent donc pas, ou très peu, de relations de paralogie, et omettent souvent beaucoup de relations d'orthologie - par exemple lorque deux gènes orthologues ont un ancêtre commun assez éloigné pour qu'ils aient eu le temps de diverger. C'est pourquoi dans ce chapitre, on suppose que l'on ne connaît que certaines relations d'orthologie. Nous verrons que ces informations, même si elles sont partielles, peuvent nous permettre de détecter et corriger des erreurs sur un arbre de gènes.

5.1 Abstract

Background. Reconciled gene trees yield orthology and paralogy relationships between genes. This information may however contradict other information on orthology and paralogy provided by other footprints of evolution, such as conserved synteny.

Results. We explore a way to include external information on orthology in the process of gene tree construction. Given an initial gene tree and a set of orthology constraints on pairs of genes or on clades, we give polynomial-time algorithms for producing a modified gene tree satisfying the set of constraints, that is as close as possible to the original one according to the Robinson-Foulds distance. We assess the validity of the modifications we propose by computing the likelihood ratio between initial and modified trees according to sequence alignments on Ensembl trees, showing that often the two trees are statistically equivalent.

Availability. Software and data available upon request to the corresponding author.

5.2 Introduction

A gene tree represents the evolutionary relationships between a set of homologous genes. Gene trees are useful to unveil the molecular evolutionary events that have shaped today's genomes. They are traditionally constructed from sequence alignments [55], while recent methods also use the information from species phylogenies through reconciliation [2, 11, 16, 68, 131, 157, 161]. But constructing good gene trees is still challenging: for example, while they yield orthology and paralogy relationships between genes, often alternative or additional information, such as conserved synteny, is used to provide or confirm orthology [91].

The orthology information suggested by gene tree reconciliation may be contradictory with that suggested by an external source, such as conserved synteny [27, 102]. We explore a way to reconcile them by performing slight modifications to a given gene tree in order to fit external information on orthology.

We propose two kinds of gene tree modification, which consist in computing a gene tree as close as possible to the initial one, satisfying two kinds of constraints. One kind is a set of pairs of genes that should be orthologous but are seen as paralogous in the initial tree. This occurs when orthologs are computed with synteny for example [102]. The other kind is a set of clades that should be rooted by speciation nodes but are rooted by duplication nodes in the initial tree. This occurs when dubious duplications are detected because of the absence of extant support for a duplication, or because of ancestral synteny information [27]. We give polynomial-time algorithms for both problems under the Robinson-Foulds distance, thus proposing several ways to improve gene trees according to external information.

There are very few gene tree reconstruction methods including synteny information [167], whereas integrating this information could be valuable [17]. The modifications we propose could be included in a local search framework as other kinds of modifications based on duplications and losses [25, 70, 113, 117]. We assess the validity of the modifications we propose by computing the likelihood ratio between initial and modified trees according to sequence alignments on Ensembl trees [61], showing that often the two trees are statistically equivalent.

5.3 Different gene tree corrections

5.3.1 Phylogenies

A *phylogeny* is a rooted binary tree which represents the evolutionary relationships between the nodes. Internal nodes are extinct ancestors, leaves

are extant elements and edges represent direct descents between parents and children. Given a node x of a phylogeny T, we call an *ancestor* of x any node on the path from the root (inclusively) of T to the parent of x. For a leaf-subset X of T, $\mathrm{lca}_T(X)$, the *lowest common ancestor* of X, denotes the farthest node from the root which is an ancestor of all elements of X. We use the notation $l(x)$, and call the *clade* of x, the set of leaves which are descendant from an internal node x. We also denote by $l(T)$ the set of leaves, and by $V(T)$ the set of nodes of T.

We define two kinds of phylogenies: species trees and gene trees. Species are identified with *genomes*. For our purpose, genomes are simply sets of genes. Therefore, each gene g, extant or ancestral, belongs to a species $s(g)$. We then have one species tree S, where nodes are identified with species, and many gene trees, where nodes are identified with genes. The set of genes in a gene tree is called a *gene family*.

A *reconciliation* between a gene tree G and a species tree S consists in assigning to each gene g of G (both extant and ancestral) the species $s(g)$ corresponding to the lowest common ancestor in S of the set $\{s(l),\ \text{for all } l \in l(g)\}$. Every internal node g of G is labeled by an *event* $E(g)$, verifying $E(g) =$ speciation if $s(g)$ is different from $s(g_\ell)$ and $s(g_r)$ where g_ℓ and g_r are the two children of g, and $E(g) =$ duplication otherwise.

The reconciliation of G and S gives all informations about the gene family history. In particular it defines the gene content of an ancestral species at the time of speciation. A reconciliation also implies the orthology and paralogy relationships between genes: two genes g and g' of T are said to be *orthologous* if $E(\mathrm{lca}_T(g, g')) =$ speciation; g and g' are *paralogous* if $E(\mathrm{lca}_T(g, g')) =$ duplication. For example, Figure 5.1.(1) shows a gene tree reconciled with a species tree. In this gene tree $a1$ and $b1$ are paralogous as their lowest common ancestor is d which is a duplication node, while $a2$ and $b2$ are orthologous. The number of dots inside big circles represents the number of genes in the corresponding genome (each big circle represents a

species).

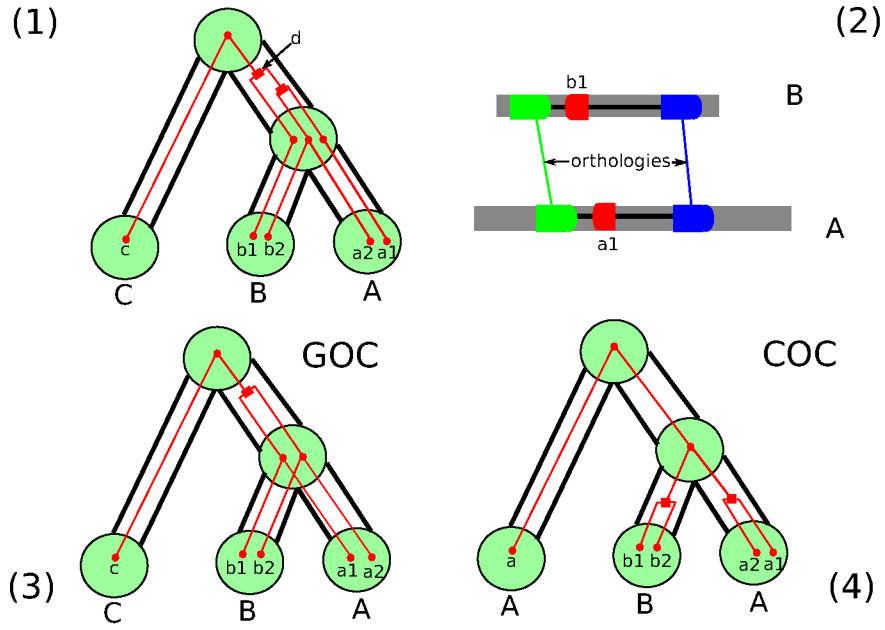

Figure 5.1: (1) A gene tree (the "initial tree") for the gene family $\{c, b1, b2, a1, a2\}$ is shown with small red nodes and single thin red edges. It is reconciled with the phylogeny of the three species A, B and C shown with large green nodes and hollow edges represented by a pair of parallel black lines. Duplication nodes of the reconciled gene tree are squared, while speciation nodes and leaves are dots. (2) The two neighbors of $b1$ on genome B and of $a1$ on genome A are inferred to be orthologous according to their lowest common ancestor in their respective gene trees (not shown). This is an argument for infering orthology between $b1$ and $a1$, which is in contradiction with the information provided by the initial tree: their lowest common ancestor is a duplication, and thus they are inferred to be paralogous. (3) A solution to the GOC problem, that is a gene tree of minimum RF distance with the initial tree verifying the constraint of $b1$ and $a1$ being orthologous. (4) A solution to the COC problem, that is a reconciled tree in which the clade $\{b1, b2, a1, a2\}$ of d in the initial tree is rather rooted by a speciation node in the corrected tree. This is an example where the optimal solutions to the two problems differ.

5.3.2 The Robinson-Fould (RF) distance

The RF distance $RF(G, G')$ between two phylogenies G and G' is the cardinality of the symmetric difference between the clade-sets of the two trees. In other words, denote by $c(G, G')$ the number of clades that are in G but not in G'. Then $RF(G, G') = c(G, G') + c(G', G)$.

In this paper, since we only compare rooted binary trees sharing the same leaf-sets, they always have the same number of internal nodes, and hence the same number of clades. Therefore $c(G, G') = c(G', G)$, and $RF(G, G') = 2c(G, G')$.

5.3.3 Two correction problems

Suppose that in addition to a species tree and a set of reconciled gene trees, we are given additional information of two kinds:

- Pairs of genes that we know are orthologous;
- Duplication nodes of some gene trees that we suspect to be false.

Constraints of orthology on pairs of genes may for example be generated from synteny analysis [91, 102]. Some pairs may contradict the information given by the gene tree. Let P be a set of pairs (g_1, g_2) of orthologous extant genes (verifying $s(g_1) \neq s(g_2)$). A gene tree G is said to *satisfy* a set P if, for any pair $(g_1, g_2) \in P$, $\text{lca}_G(g_1, g_2)$ is a speciation node.

Problem 5.1. ***Gene Orthology Correction [GOC] Problem***
Input*: A gene tree G reconciled with a species tree S, and a set P of gene pairs that are required to be orthologous;*
Output*: A corrected gene tree G_P satisfying P, such that $RF(G, G_P)$ is minimum among all possible solutions.*

An example is given in Figure 5.1: (1) is the initial tree, and (2) depicts two syntenic regions of size 3 surrounding genes $b1$ and $a1$. In general (if we

neglect the effect of gene conversion) genes in two syntenic regions should be either all pairwise orthologous or all pairwise paralogous [102]. Consequently, if the two neighbors of $b1$ on genome B and of $a1$ on genome C are inferred to be orthologous (according to their lowest common ancestor in their respective gene trees), then an orthology constraint should be imposed on the pair $(b1, a1)$. Figure 5.1. This principle is usually considered as one of the most efficient method to detect orthologies [91]. (3) is a corrected tree.

On the other hand, duplication nodes of a gene tree can be considered dubious for different reasons. For example, in Ensembl [164], "dubious" is a label assigned to the non-apparent duplication nodes [26, 43] pointing to an incongruence between the gene tree and the species tree. Alternatively, inferred ancestral synteny may also point to dubious duplication nodes [27]. Formally, clades corresponding to some duplication nodes may erroneously be considered as sets of paralogous genes, and should rather be considered as orthologous. A gene tree G is said to *satisfy* a set C of its clades if $E(\text{lca}_G(c)) = \text{speciation}$ for all $c \in C$.

Problem 5.2. ***Clade Orthology Correction [COC] Problem***
Input*: A gene tree G reconciled with a species tree S, and a set C of clades of G assigned to duplication nodes;*
Output*: A corrected tree G_C satisfying C, such that $RF(G, G_C)$ is minimum among all possible solutions.*

The two problems are different, as exemplified by Figure 5.1, where (3) is an optimal solution to GOC while (4) is an optimal solution to COC, the latter more distant to the initial tree.

In the next two sections, we use S for the species tree name, G for the reconciled gene tree, and we give efficient solutions to these two problems.

5.4 The Gene Orthology Correction Problem

Notice that for any instance of the GOC problem, a corrected tree satisfying P always exists. Indeed, for any extant species x of S, one can make a tree whose leaf-set is all the extant genes g of G for which $s(g) = x$. Doing this for every species yields a forest whose roots can be reconnected by matching the topology of S, ensuring that any pair of genes not in the same species are orthologous. However, the obtained tree can be very far from the original.

Let P be a set of gene pairs (which are leaves of G) required to be orthologous. Notice that if $(a, b) \in P$, then we also have $(b, a) \in P$. For any pair $(a, b) \in P$, if $\text{lca}_G(a, b)$ is a duplication in G, then (a, b) is a pair of *false paralogs*. The set $P_f \subseteq P$ denotes the set of all false paralogous pairs of P.

Given two distinct leaves a and b of G, we set $r_{a,b} = \text{lca}_G(a, b)$, $s_{a,b} = \text{lca}_S(s(a), s(b))$, and define $h_{a,b}$ as the highest node (closest to the root) on the path from a to $r_{a,b}$ such that $s(h_{a,b})$ is a descendant of $s_{a,b}$. Notice that $h_{a,b}$ can be a itself, but not $r_{a,b}$.

For instance on Figure 5.2.(1), a_1, c_2 are false paralogs with $r_{a_1,c_2} = e_1$ and $s_{a_1,c_2} = E$. From this, one can deduce that $h_{a_1,c_2} = d_2$ and $h_{c_2,a_1} = c_2$. We show below that, for any pair (a, b) of false paralogs, $h_{a,b}$ is the highest node on the path from a to $r_{a,b}$ over which we can move b to make $\text{lca}_G(a, b)$ a speciation node. The reason for moving b as high as possible is to preserve as many clades as possible, allowing a minimum RF distance between the initial and corrected tree.

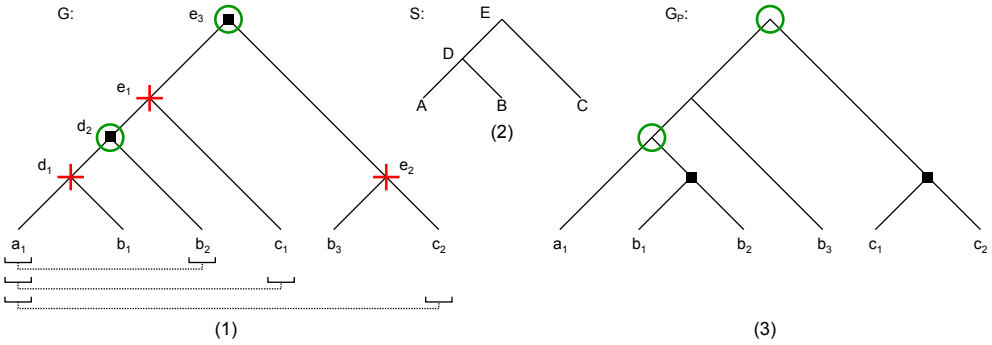

Figure 5.2: (1) A gene tree G reconciled with species tree S. Duplication nodes are denoted by a black square. The leaves and internal nodes of G are labeled with the letter of their corresponding species. Brackets denote the required orthologs given by the input set $P = \{(a_1, b_2), (a_1, c_1), (a_1, c_2)\}$. The non-preservable nodes (nodes of H) are depicted by red crosses, while preservable nodes are circled in green. (2) The species tree associated with G. (3) The tree G_P, a solution to the GOC problem, which preserves every possible clade.

Lemma 5.3. *Let (a, b) be a pair of false paralogs in G, and let G' be a tree in which a and b are orthologous. If x is an ancestor of $h_{a,b}$ and a descendant of $r_{a,b}$, then the clade of x is not in G'.*

Proof. Suppose otherwise that there is some $x' \in V(G')$ with the same clade as x (and hence $s(x) = s(x')$). Let $r'_{a,b} = \text{lca}_{G'}(a, b)$, which should be a speciation. Since b was not in the clade of x, it cannot be in the clade of x' either, implying that $r'_{a,b}$ is an ancestor of x'. Also, since $s(x') = s(x)$ and x is above $h_{a,b}$ in G, we have that $s(x')$ is $s_{a,b}$ or one of its ancestors (otherwise we would have picked x to be $h_{a,b}$). But $r'_{a,b}$ has x' in one of its subtrees, and b in the other, implying that $r'_{a,b}$ is a duplication: contradiction. □

We now have a way to identify a set of clades that cannot be in G_P. For any $(a, b) \in P_f$, denote by $H_{a,b}$ the set of ancestors of $h_{a,b}$ that are descendants of $r_{a,b}$. If G_P satisfies the set P_f, G_P cannot contain any clade from the set $H = \bigcup_{(a,b)\in P_f} H_{a,b}$. It follows that a minimum of $|H|$ clades of

G are missing in G_P. We claim that a solution G_P to the GOC problem is obtained by modifying exactly $c(G, G_P) = |H|$ clades.

Theorem 5.4. *Let G_P be a solution to the GOC problem. Then $RF(G, G_P) = 2|H|$.*

In what follows, we give a constructive proof of Theorem 5.4 by describing an algorithm for solving the GOC problem.

5.4.1 An algorithm for the GOC problem

Call $V(G) \setminus H$ the set of *preservable nodes* of G (those that we hope to preserve). For example in Figure 5.2.(1), $H = H_{a_1,c_2} \cup H_{c_2,a_1} \cup H_{a_1,c_1} \cup H_{c_1,a_1} \cup H_{a_1,b_2} \cup H_{b_2,a_1} = \{e_1\} \cup \{e_2\} \cup \emptyset \cup \emptyset \cup \{d_1\} \cup \emptyset = \{e_1, e_2, d_1\}$. The nodes of H are represented by red crosses, while the preservable nodes are circled in green. Notice that the root r of G is preservable, since any solution G_P to the GOC problem should share the same leaf-set as G. Consider the set $\mathcal{G}$ of subtrees of G rooted on the *highest preservable descendants* of r, i.e. preservable nodes for which r is the unique preservable ancestor. Observe that since any leaf of G is preservable, we have $\bigcup_{G_x \in \mathcal{G}} l(G_x) = l(G)$. If, for some $(g_1, g_2) \in P$, g_1 and g_2 are scattered across two subtrees of $\mathcal{G}$, we call these subtrees *required orthologous subtrees* (or simply *required orthologs* when the context is clear as to whether we are comparing genes or subtrees). For example in the tree G of Figure 5.2.(1), $\mathcal{G}$ is the set of subtrees rooted at d_2, c_1, b_3 and c_2 (the last four restricted to a single leaf), and the subtrees rooted at d_2 and c_1 are required orthologs, as well as those rooted at d_2 and c_2. However, connecting two subtrees under a speciation might not always be feasible. A definition of *possible orthologs* follows.

Definition 5.5 (Possible orthologs)**.** *Two subtrees $G_1, G_2 \in \mathcal{G}$ rooted at x_1, x_2 respectively are* possible orthologs *if and only if $s(x_1)$ and $s(x_2)$ are unrelated, i.e. neither is an ancestor of the other in S.*

The following lemma ensures that the roots of required orthologous subtrees can actually be joined under a common parent which is a speciation.

Lemma 5.6. *Let $G_1, G_2 \in \mathcal{G}$ be required orthologs. Then G_1 and G_2 are possible orthologs.*

Proof. Let x_1, x_2 be the roots of G_1, G_2 respectively, and let $(g_1, g_2) \in P$ such that $g_1 \in l(G_1)$ and $g_2 \in l(G_2)$. Let s_ℓ, s_r be the left and right children of s_{g_1,g_2}, and denote by S_ℓ and S_r the subtrees of S rooted at s_ℓ and s_r respectively. Suppose without loss of generality that $s(g_1)$ is in $l(S_\ell)$ and $s(g_2)$ is in $l(S_r)$. Since x_1 is preservable and on the path between g_1 and r_{g_1,g_2}, we have $x_1 \notin H_{g_1,g_2}$ and thus $s(x_1) \in V(S_\ell)$. Similarly, $s(x_2) \in V(S_r)$. Therefore $s(x_1)$ and $s(x_2)$ are unrelated and possible orthologs. □

The problem, formally defined in the sequel as the *maximum orthology tree*, consists in joining all trees of $\mathcal{G}$ into a single tree G' in a way ensuring that each pair of possible orthologs is joined under a speciation. More precisely, for some possible orthologs $G_1, G_2 \in \mathcal{G}$ rooted at nodes x_1, x_2, we get that $\text{lca}_{G'}(x_1, x_2)$ is a speciation, with G_1, G_2 being unchanged.

We begin by giving an overview of the whole algorithm.

Algorithm Outline:

1. Compute the set $H = \bigcup_{(a,b)\in P_f} H_{a,b}$ of internal nodes of G corresponding to clades that cannot be in G_P;

2. Compute the set $\mathcal{G}$ of subtrees rooted at the highest preservable descendants of the root of G. If $\mathcal{G}$ is empty, return G and terminate;

3. Construct a tree G' by joining all trees of $\mathcal{G}$ in a way ensuring that possible orthologs are joined under speciation. We call G' the *maximum orthology tree* for $\mathcal{G}$;

4. For every tree $G_x \in \mathcal{G}$, construct $G_{x,P}$ by recursively repeating Steps 2 to 4 with G being G_x, and replace the G_x subtree of G' by $G_{x,P}$.

The tree obtained corresponds to the corrected tree G_P we want. Running this algorithm on the G tree of Figure 5.2 yields the corrected tree G_P. This algorithm terminates, since we eventually reach all the leaves of G, which correspond to terminal cases in the recursion. Implementing step 1 is straightforward, while step 2 can be done by performing a depth-first search from the root, in which upon visiting a preservable node, we add it to $\mathcal{G}$ and continue the search without visiting its children. Step 3 is the purpose of the next section, so assume for now that it can be performed correctly as stated. This algorithm can be implemented to run in $O(|P| \times |V(G)|)$ steps in the worst case, the main bottleneck being the computation of H. The algorithm correctness follows from the two lemmas below.

Lemma 5.7. *Any preservable node x of G is* preserved *in G_P, meaning that the clade of G rooted at x is a clade of G_P.*

Proof. Let x be a preservable node of G and G_x be the subtree rooted at x. It is not hard to see that eventually, steps 2-4 will be run on G_x and return a tree $G_{x,P}$, which will itself be a subtree of the final corrected tree G_P. As the algorithm only moves and reconnects subtrees of G_x, we have that $l(G_x) = l(G_{x,P})$. Since $G_{x,P}$ is a subtree of G_P, it follows that the clade of x is preserved in G_P. □

Lemma 5.8. *Let $(g_1, g_2) \in P$. Then g_1 and g_2 are orthologs in G_P.*

Proof. Denote by G_v the subtree rooted at v, for some $v \in V(G)$. Let x be a preservable node and $G_{x,P}$ be the subtree produced after running steps 2-4 on G_x. Let D be the set of highest preservable descendants of x. We say that a gene pair (g_1, g_2) is *contained* in G_x if $g_1, g_2 \in l(G_x)$. We use induction on the height of the tree to show that all gene pairs in P that are contained

in G_x are orthologous in $G_{x,P}$ (which proves the lemma since x can be the root). This is trivially true for leaves as they are preservable and contain no gene pairs. We thus suppose by induction that for any $d \in D$, gene pairs in P that are contained in G_d are orthologous in $G_{d,P}$. Let $(g_1, g_2) \in P$ such that (g_1, g_2) is contained in G_x, but there is no $d \in D$ such that G_d contains (g_1, g_2). What is left to prove is that g_1 and g_2 are orthologous in $G_{x,P}$.

We first observe that g_1, g_2 belong to two different subtrees G_{d_1}, G_{d_2}, where $d_1, d_2 \in D$. Otherwise $G_{d_1} = G_{d_2}$, implying that (g_1, g_2) is contained in G_{d_1} and we are done. Therefore, G_{d_1}, G_{d_2} are required orthologs, and hence possible orthologs. Since we may assume that G_{d_1} and G_{d_2} are joined under a speciation in $G_{x,P}$, we get that $\text{lca}_{G_{x,P}}(g_1, g_2)$ is a speciation. The result follows from observing that $G_{x,P}$ is a subtree of G_P. □

5.4.2 Maximum orthology tree

We now describe a solution to the maximum orthology tree problem. Formally, given a set of k possible orthologous subtrees of G rooted on a set of nodes $X = \{x_1, \ldots, x_k\}$, the problem is to construct a tree F with $l(F) = X$, such that for each pair $x_i, x_j \in X$ that correspond to roots of possible orthologs, x_i and x_j are orthologous in F.

Roughly speaking, the algorithm proceeds as follows: start with F_0 being a copy of S. Iterate over i from 1 to k, at each step constructing F_i by grafting x_i on F_{i-1} right above the node $v \in V(F_0)$ such that $s(v) = s(x_i)$. Proceeding this way, we show in Lemma 5.9 that nodes of $V(F_0)$ are ensured to remain speciation nodes all over the procedure, and in lemma 5.10 that the lowest common ancestor of two possible orthologs belongs to $V(F_0)$, leading to corollary 5.11 stating that possible orthologs are in fact orthologous in the output tree. Finally remove the leaves artificially introduced by F_0 and *standardize* the tree, which means

- remove all nodes with no descendant labeled with extant genes;

- contract non-root degree 2 nodes, then contract the root if it is of degree one.

Starting with F_0 being a copy of S is a step that might be omitted, but the set of nodes $V(F_0)$ serves as a skeleton around which we graft our x_i's, making it both easily implementable and provable. Figure 5.3 shows how the algorithm proceeds on the set of highest preservable descendants of the root of the tree G in Figure 5.2.(1).

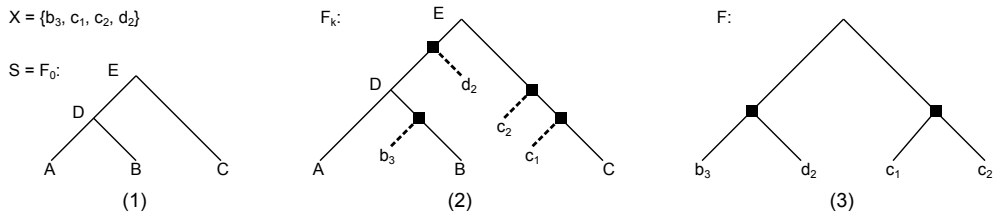

Figure 5.3: An instance of the *max orthology problem*, with X being the highest preservable descendants of the root of G in figure 5.2. (1) The starting tree F_0, which is a copy of S. (2) The F_k tree, which depicts the tree obtained after grafting every node of X. (3) The final tree F, obtained by removing the leaves initially in F_0 and standardizing.

Algorithm 4 $findMaxOrthology(S, X = \{x_1, \cdots, x_k\})$

$F_0 \leftarrow$ A copy of S
$V_0 \leftarrow V(F_0)$
$L \leftarrow l(F_0)$
for $i = 1 \rightarrow k$ **do**
  Find the unique node $v \in V_0$ such that $s(v) = s(x_i)$
  $F_i \leftarrow$ a copy of F_{i-1} on which we graft x_i on the edge linking v to its parent node
    (or if v is the root of F_{i-1}, create a new root with children v and x_i)
end for
$F \leftarrow F_k$ on which we remove L and stardardize

Lemma 5.9. *If $r \in V(F_0) \cap V(F)$, then r is a speciation.*

Proof. Since F_0 is a copy of S, all nodes of $V(F_0)$ are initially speciation nodes. We show that each grafting operation does not change the event corresponding to these nodes. Say that at iteration i, we graft x_i on the edge linking v to its parent node p. We first observe that the only nodes that can be transformed from speciation in F_{i-1} to duplication in F_i are on the path from p to the root of F_{i-1}. Suppose without loss of generality that v is the left child of p in F_{i-1}, and let w be the newly created node between p and v in F_i. Thus w has children x_i and v, and since $s(x_i) = s(v)$, we get that $s(w) = s(v)$. It follows that if p was a speciation in F_{i-1}, it remains a speciation in F_i. Moreover, this implies that $s(p)$ is left unchanged in F_i, implying in turn that any ancestor of p cannot change from speciation to duplication. Therefore, no grafting operation can affect speciation of any vertex in $V(F_{i-1})$. Finally, we note that removing leaves or deleting degree two nodes in F also cannot affect speciation nodes. □

Lemma 5.10. *Let $x_i, x_j \in X$ be the roots of possible orthologous subtrees. Then,* $\text{lca}_F(x_i, x_j) \in V(F_0)$.

Proof. First recall that if x_i, x_j are the roots of possible ortholog subtrees, then there is some $s \in V(S)$ such that $s(x_i)$ and $s(x_j)$ are in the left and right subtrees of s, respectively. Now, let r be the unique node in $V(F_0)$ such that $s(r) = s$, and let $v_i, v_j \in V(F_0)$ such that $s(v_i) = s(x_i)$ and $s(v_j) = s(x_j)$. It is clear that in F_0, $\text{lca}(v_i, v_j) = r$. This also holds for any F_i by observing that grafting nodes cannot change the lca relationship. Since x_i is grafted on some edge between v_i and r, and x_j between v_j and r, it follows that $\text{lca}(x_i, x_j) = r \in V(F_0)$. □

Corollary 5.11. *Let $x_i, x_j \in X$ be the roots of possible orthologs. Then they are orthologous in F.*

5.5 The Clade Orthology Correction Problem

We prove several results characterizing the solutions to the COC problem. Let C be a set of clades that has to be satisfied. For a clade $c \in C$, we denote by $s(c)$ the value of $s(r(c))$ where $r(c)$ is the root of c, and by $E(c)$ the value of $E(r(c))$ that we call *the label of* c.

First, unlike in the GOC problem, a solution to the COC problem does not always exist. Indeed, it is possible that no gene tree has all clades in C labeled by speciations. We give a necessary and sufficient condition for the existence of a solution. The following lemma is obvious from the definition of reconciliation, and will be used in several proofs.

Lemma 5.12. *For a reconciled gene tree G, if a node x is an ancestor of a node y and $s(x) = s(y)$ then $E(x) =$ duplication.*

Theorem 5.13. *There is a solution to the COC problem if and only if for every clade $c \in C$, $s(c)$ is not a leaf of S, and if for every pair $c_1, c_2 \in C$, either c_1 and c_2 are disjoint sets of leaves, or $s(c_1) \neq s(c_2)$.*

The necessity of these conditions directly follow from Lemma 5.12, since $s(c_1), s(c_2)$ and the ancestry relationship between c_1 and c_2 remain unchanged in a solution. Their sufficiency will be constructively demonstrated in the sequel. Suppose that the conditions are satisfied. We give a way of finding all optimal solutions according to the RF distance, followed by two ways of finding an optimal one optimizing other criteria in addition.

Given a duplication node x of G, *pushing x by multifurcation* means applying the following procedure:

- Let $s = s(x)$, and A and B be the two children of s in S.
- Let T^A be the set of maximal subtrees of the subtree of G rooted at x, such that all their leaves l verify that $s(l)$ is a descendant of A (including A itself). Let $G^A[x]$ be the multifurcated tree obtained by joining all roots of trees in T^A under a common root.

- Let symmetrically T^B be the set of maximal subtrees of the subtree of G rooted at x, such that all their leaves l verify that $s(l)$ is a descendant of B (including B itself). Let $G^B[x]$ be the multifurcated tree obtained by joining all roots of trees in T^B under a common root.

- Let G' be obtained from G by replacing the clade rooted at x by a new subtree, obtained by joining $G^A[x]$ and $G^B[x]$ under a common root.

This rearrangement is described in [113] and applied to dubious duplications as a preprocessing step for ancestral genome reconstruction.

A *binary resolution* G_b of a multifurcated tree G is a binary tree in which all the clades of G are in G_b.

Theorem 5.14. *If there is a solution to the COC problem, then a binary gene tree is an optimal solution if and only if it is a binary resolution of the multifurcated tree obtained by pushing the roots of the elements of C by multifurcation (in any order).*

Proof. It is clear that a binary resolution is a solution, provided that the conditions for the existence of a solution are satisfied. Indeed any clade is preserved through pushing a duplication node, so this operation can be done for all clades in C independently. This proves the converse part of Theorem 5.13.

Then it is an optimal solution because by Lemma 5.12, no clade x which is a descendant of the pushed clade c such that $s(c) = s(x)$ may be conserved if we want c to be a speciation node. And by construction all clades such that $s(c) \neq s(x)$ are preserved by this operation. □

Binary resolutions which minimize the number of duplications and losses are studied by [101] and may be applied to provide *bona fide* phylogenies. We describe an alternative maximizing the number of common triplets. A *triplet* in a tree G is a set of three leaves $((a, b), c)$ of G, such that the LCA of the three is strictly more ancient than the LCA of the first two.

Given a species tree S, a reconciled gene tree G and one of its duplication nodes x, *pushing x by tree duplication* means applying the following procedure, illustrated in Figure 5.4:

- Let $s = s(x)$, and A and B be the two children of s in S.
- Let $G^A[x]$ be a tree obtained from the subtree of G rooted at x, by deleting all leaves l with $s(l)$ being a descendant of A, and standardizing it, which as in the previous sections, means
  - removing all nodes with no descendant labeled with extant genes;
  - contracting non-root degree 2 nodes, then contracting the root if it is of degree one.
- Let symmetrically $G^B[x]$ be a tree obtained from the subtree of G rooted at x, by deleting all leaves l with $s(l)$ being a descendant of B, and standardizing it.
- Let G' be obtained from G by replacing the clade rooted at x by a new subtree, obtained by joining $G^A[x]$ and $G^B[x]$ under a common root.

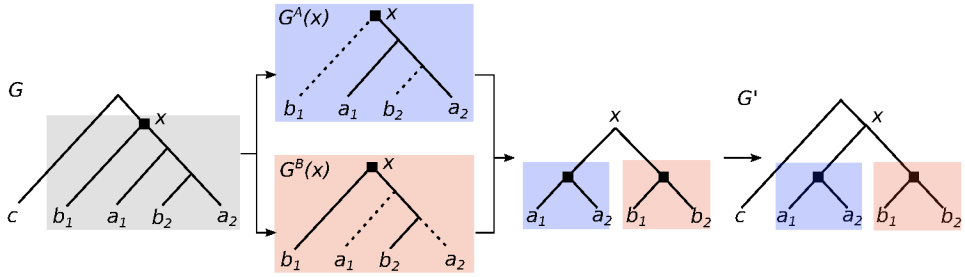

Figure 5.4: An instance of the COC problem. In the gene tree G, where s(a_1)=s(a_2)=A, s(b_1)=s(b_2)=B and s(c)=C, extract and copy the subtree rooted at x to get the subtrees $G^A(x)$ and $G^B(x)$. Remove b_1 and b_2 from $G^A(x)$ and a_1 and a_2 from $G^B(x)$. Join $G^A(x)$ and $G^B(x)$ under a common root and replace G(x) by the new subtree in the gene tree G'.

Note that if a clade y is disjoint from x or assigned to a different species, then pushing x by tree duplication does not affect the subtree rooted at y. In consequence, pushing several clades by tree duplications in any order gives a unique solution if the clades satisfy the properties of Lemma 5.13.

Theorem 5.15. *If there is a solution to the Clade Orthology Correction problem, the gene tree obtained by successively pushing the roots of the elements of C by tree duplication (in any order) is an optimal solution. Among all optimal solutions, it maximizes the number of common triplets with G.*

Proof. As already noticed pushing a duplication by multifurcation preserves all clades assigned to species which are different from the species assigned to the pushed node. So it is an optimal solution.

Now we have to prove that none of the triplets that are in G but not in G' can be preserved in any other optimal solution. For this we characterize the triplets that can be preserved. For a triplet $((a, b), c)$ of G, let $T_{((a,b),c)}$ be the rooted phylogeny with three leaves and two internal nodes containing the triplet. If the leaves a, b, c are in the pushed clade x, then the triplet can be preserved only if in the reconciliation of $T_{((a,b),c)}$, the lowest internal node is not mapped to $s(x)$. Otherwise by Lemma 5.12, the root node of the triplet cannot be a speciation.

Let $((a, b), c)$ be a triplet such that in the reconciliation of $T_{((a,b),c)}$, the lowest internal node is not mapped to $s(x)$. This triplet is entirely included in $G^1[x]$ or $G^2[x]$. So it is preserved. In consequence all triplets possibly preserved are indeed preserved by the operation, showing the optimality of the procedure reguarding the number of common triplets. □

Now if there is no solution to the Clade Orthology problem, we advice to push duplication nodes in C starting from the highest ones, without having formalized why we find this solution adequate.

5.6 Fish gene trees

Using synteny as evidence of orthology, we wanted to test the ability of our algorithm designed for the GOC problem to correct gene trees. To this end, we considered the four fish genomes *Gasterosteus aculeatus* (Stickleback), *Oryzias latipes* (Medaka), *Tetraodon nigroviridis*, and *Danio rerio* (Zebrafish) with human and mouse as outgroups. We used the *Ensembl Genome Browser* to collect all available gene trees, and filtered each tree to preserve only genes from the taxa of interest. We then reconciled the trees with the known species trees, and identified duplication and speciation nodes. Following our methodology in [102], a region surrounding a gene is defined as the substring containing the gene and both its left and right adjacencies, and two regions are considered syntenic if they contain homologous genes in the same order. We observed in [102] that more than 22% of the 6241 collected gene trees contain at least one false paralogy, that is a pair of genes required from synteny to be orthologous, but the LCA of the corresponding leaves being a duplication rather than a speciation node.

For 1000 of the trees containing at least one false paralogy, we applied the correction procedure previously described, and retrieved the gene family alignment from Ensembl. With PhyML [71], we computed the likelihood of the initial and corrected tree, given the alignment. These two likelihood values were compared with Consel [147]. For only 17.7% of the trees, the correction was rejected by the AU test. In other words, the correction algorithm is valid for a vast majority (82.3%) of the tested trees. Moreover, the likelihood of the corrected tree is higher than the original for 44.4% of the trees. Interestingly, 14.8% of the original Ensembl gene trees were rejected when compared to the corrected trees.

The correction of the gene tree for the *ZNF800* gene family, which is related to transcriptional regulation, is given as an example in figure 5.5. The corrected tree was highly favored by the AU Test, giving it a statistical sup-

port advantage with a p-value below 0.001. Furthermore, the non-apparent duplication of G, located at the root of the (m_1, t_1, s_1) subtree, was eliminated, resulting in one less duplication in G_P.

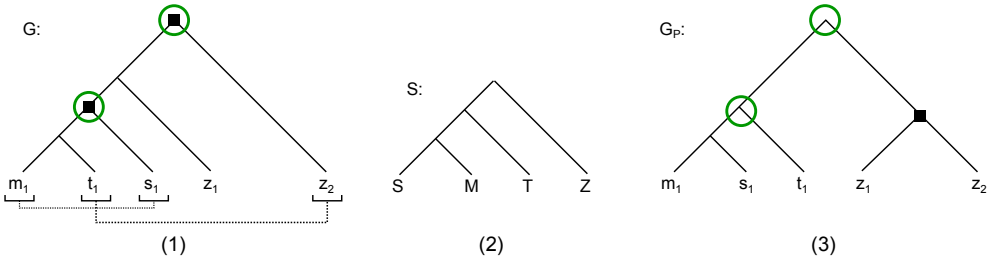

Figure 5.5: The tree for the ZNF800 gene family before and after correction, restricted to the species Stickleback (S), Medaka (M), Tetraodon (T) and Zebrafish (Z). (1) The original gene tree G given by Ensembl, using the same notation as in figure 5.2 for duplications, preservable nodes and required orthologs. Gene region analysis gave us the required orthologs $P = \{(m_1, s_1), (t_1, d_2)\}$. (2) The species tree associated with the four species. (3) The gene tree given by our correction algorithm.

5.7 Conclusion

We give two efficient algorithms for two new gene tree rearrangement problems, related to the correction of a gene tree according to some external information on orthology. The rearrangements are modifications that are as small as possible, given some distance criterion (namely the RF distance), but can be more significant according to other distances such as the usual NNI (nearest neighbor interchange) distance. We show that for fish genomes, the rearrangements we define can be efficient to explore statistically equivalent gene trees when sequence alignement is used to compute likelihood. As corrected trees satisfy synteny contraints, we can be confident enough that they describe the gene family evolution better.

Many algorithmic and theoretical problems remain open. For example, is there a similar way for handling paralogy constraints? What about having

both orthology and paralogy constraints? It can be shown that there exist sets of constraints with both types that cannot be satisfied. What are the conditions for a set of orthology/paralogy constraints to be satisfiable?

These algorithms may be used in a global framework to construct large gene tree sets which are arguably better than those found in standard databases. The implementation of such a framework is an on-going work.

Competing interests

None

Author's contributions

ML, MS, KS, ET, NE modeled the problem, devised the algorithms and wrote the paper. ML implemented the software.

Declarations

This work is funded by the Natural Sciences and Engineering Reserach Council of Canada (NSERC), Fonds de Recherche Nature et technologies of Quebec, Agence Nationale pour la Recherche and Ancestrome project ANR-10-BINF-01-01.

5.7.1 Contributions

Manuel Lafond, Nadia El-Mabrouk et Krister M. Swenson ont modélisé le problème du GOC. Manuel Lafond et Nadia El-Mabrouk ont conçu l'algorithme du problème GOC. Manuel Lafond a rédigé les preuves de bon fonctionnement de l'algorithme et a implémenté le logiciel. Magali Semeria et Eric Tannier ont modélisé le problème du COC, ont conçu l'algorithme du problème COC et ont rédigé les preuves sous-jacentes. Tous les auteurs ont participé à la rédaction de l'article.

CHAPITRE 6

ORTHOLOGY AND PARALOGY CONSTRAINTS: SATISFIABILITY AND CONSISTENCY

Manuel Lafond[1] , Nadia El-Mabrouk[1]

Dans le chapitre précédent, nous supposions n'avoir accès qu'à des relations d'orthologie. Nous nous intéressons maintenant au cas où certaines relations de paralogie peuvent aussi être connues. Bien que la plupart des méthodes d'inférence de relations s'attardent surtout à l'orthologie, il est parfois possible de déduire que deux gènes sont paralogues (outre le cas trivial dans lequel les deux gènes sont dans la même espèce). Par exemple, les méthodes d'inférence basées sur la similarité de séquence peuvent être paramétrée afin de prédire plus ou moins d'orthologues. Si deux gènes ne sont jamais considérés comme orthologues, et ce même avec les paramètres les plus permissifs, alors peut-être peut-on les considérer comme paralogues. Par ailleurs, certaines méthodes utilisent la synténie pour inférer des relations de paralogie [91, 103], l'idée étant que si deux gènes homologues g et g' sont dans un bloc synténique n'ayant que des paralogues, alors g et g' devraient aussi être paralogues (notons que c'est aussi vrai pour les orthologues).

On peut donc étendre la question du chapitre 5 à la suivante: étant donné un ensemble de relations d'orthologie **et** paralogie, peut-on corriger un arbre de gènes G de façon minimale afin qu'il contienne ces relations? Cette question est toutefois plus complexe, puisque contrairement au cas où seules des orthologies sont connues, il se peut qu'il n'existe pas d'arbre contenant les relations données. Alors, avant de pouvoir corriger un arbre en se basant sur des relations, il est nécessaire de vérifier la validité de ces relations.

[1]DIRO, Université de Montréal, Canada

Le but de cet article est de détecter dans quel cas il existe un arbre de gènes reflétant un ensemble de relations donné. Mais comme les logiciels d'inférence de relations ne peuvent vraisemblablement pas trouver toutes les relations, on suppose que certaines peuvent être inconnues. Nous étudions trois cas pour lesquels on veut vérifier l'existence d'un arbre de gènes G étiqueté:

1. *satisfaisabilité*: les spéciations de G peuvent être placées sur n'importe quel noeud de l'arbre de gènes;

2. *consistance avec* S: les spéciations de G doivent être consistantes avec un arbre d'espèces donné S;

3. *consistance*: les spéciations de G doivent être consistante avec *au moins* un arbre d'espèces (i.e. S n'est pas donné, mais il doit exister).

Si la relation entre chaque paire de gènes est connue, on décrit comment vérifier, en temps polynomial, la satisfaisabilité dans [80] et la consistance dans [82]. Les algorithmes présentés passent par l'évaluation du *graphe de relations*, dans lequel les sommets sont les gènes et une arête est ajoutée entre deux gènes s'ils sont orthologues. La propriété-clé de ces méthodes, et de celles présentées dans l'article, est la suivante: un graphe de relations est satisfaisable si et seulement si il ne contient aucun chemin induit avec 4 sommets (i.e. il n'a aucun P_4 induit). Dans le cas où certaines relations sont inconnues, nous utilisons cette caractérisation pour montrer que la satisfaisabilité et la consistance avec S peuvent être évaluées en temps polynomial, tandis que la complexité de la consistance demeure ouverte.

6.1 Abstract

Background A variety of methods based on sequence similarity, reconciliation, synteny or functional characteristics, can be used to infer orthology and paralogy relations between genes of a given gene family $\mathcal{G}$. But is a given set $\mathcal{C}$ of constraints possible, i.e., can they simultaneously co-exist in an evolutionary history for $\mathcal{G}$? While previous studies have focused on full sets of constraints, here we consider the general case where $\mathcal{C}$ does not necessarily involve a constraint for each pair of genes. The problem is subdivided in two parts: (1) Is $\mathcal{C}$ *satisfiable*, i.e. can we find an event-labeled gene tree G inducing $\mathcal{C}$? (2) Is there such a G which is *consistent*, i.e., such that all displayed triplet phylogenies are included in a species tree?

Results We show how known results on the *Graph sandwich problem* can be used to answer (1) and provide polynomial-time algorithms for satisfiability and consistency with a given species tree. We also describe a new polynomial-time algorithm for the case of consistency with an unknown species tree and full knowledge of relations, as well as a branch-and-bound algorithm in the case when unknown relations are present. We show that our algorithms can be used with appropriate combinations of parameter settings of Proteinortho, a sequence similarity-based orthology detection tool.

Availability Software is available at http://www-ens.iro.umontreal.ca/~lafonman/software.php.

6.2 Introduction

Gene families, usually constructed from sequence similarity, group *homologous* genes, i.e., genes sharing a common ancestry: starting from a single gene copy, a history of speciations, duplications and losses is assumed to be at the origin of the observed set of extant genes. Deciphering the *orthology* (divergence following a speciation) and *paralogy* (divergence following a duplication) relations between pairs of genes inside a gene family is important

and lies at the heart of many genomics studies. The reconstruction of species trees for example is usually based on the selection and alignment of orthologous gene copies. From a functional point of view, orthologs are believed to be more likely similar in function than paralogs [120].

Orthology/paralogy information is often derived from a reconciliation approach (first introduced by Goodman in 1979 [67]). A gene tree that best reflects the evolution of the sequences is first constructed for the gene family. Assuming a known phylogeny for the set of taxa, the non-agreement between the two trees is then explained by a set of duplication and loss events (other events such as horizontal gene transfer might also be inferred by reconciliation, although we will ignore them here). Reconciliation leads to a labeling of internal nodes of the gene tree as duplication/speciation nodes, yielding a full orthology/paralogy interpretation for each pair of genes (cf. e.g. TreeFam [137, 140] used for constructing the *Ensembl Compara* gene trees [61], PHOG [38], MetaPHOrs [127]). This approach assumes that an accurate gene tree can be constructed for the gene family. Although inferring phylogenies is a field with a very long history, due to various limitations constructing good gene trees is still challenging. A variety of other methods have been developed for the purpose of orthology/paralogy detection. A well-known class of algorithms is based on clustering genes according to their sequence similarity (cf. e.g. the COG database [160], OrthoMCL [106], InParanoid [10], Proteinortho [105]). Recently, we investigated another way of detecting orthology/paralogy based on conserved synteny (conservation in gene order) [100, 102]. Other initiatives, such as the Gene Ontology project [33], provide functional annotation that can be used as another source of orthology relations. In contrast to the reconciliation approach, only partial relations can be expected from such tree-free methods.

The orthology/paralogy information suggested by gene tree reconciliation may be contradictory with that suggested by an external source. As gene trees are known to be error-prone, more confidence can be given to such

homology information when it is well-supported by various genomic observations. This raises the problem of gene tree editing based on a known set $\mathcal{C}$ of pairwise orthology/paralogy constraints. But prior to any algorithmic consideration, one should be able to state whether the set $\mathcal{C}$ is possible, i.e. whether all constraints can simultaneously co-exist in an evolutionary history of the gene family. In a recent work [100], we showed that a set of orthlogy constraints is always possible and we gave a polynomial-time algorithm for correcting a gene tree in a minimal way according to the Robinson-Foulds distance.

Recent studies have considered the connection of trees and orthology from the angle of reconstructing phylogenies from orthology relations [80–82]: How much information about the gene tree, the species tree and their reconciliation is already contained in the orthology relation between genes? In other words, having a set $\mathcal{C}$ of full pairwise orthology/paralogy relations (each pair of genes is constrained), can one reconstruct the gene and species trees? Similarly to gene tree editing, the first question to be asked is whether the orthology/paralogy constraints can simultaneously co-exist in a history of the gene family. Interestingly, by making the link with symbolic ultrametrics and co-graphs, a simple characterization of *satisfiability* (symbolic ultrametric) for full paralogy/orthology relations is given in [80], where satisfiability relates to the existence of an event-labeled gene tree G (symbolic representation) leading to $\mathcal{C}$. Notice that satisfiability does not mean the possibility for orthology/paralogy relations to co-exist in a true history, as the triplet phylogenies contained in G are not necessarily *consistent* (included in a species tree). The derivation of a species tree from an event-labeled gene tree is considered in [82]. Finally, the outline of a computational framework for the construction of a least resolved species tree S from a set of orthology/paralogy relations, involving the extraction of maximum satisfiable relations and maximum consistent triple set is given in [81].

Here, we consider the general case for $\mathcal{C}$: in contrast with [80, 82], we do

not require $\mathcal{C}$ to be full, i.e., to involve a constraint for each pair of genes. We introduce the notations and problems in Section 6.3, and show in Section 6.4 how the *Graph sandwich problem* solves the problem of satisfiability. In Section 6.5 we adapt this algorithm to the problem of consistency with a given species tree. We then study in Section 6.6 the problem of finding a gene tree that is consistent with *some* species tree. Finally in Section 6.7, we show how our methodology can be used with ProteinOrtho to extract a set of robust orthology/paralogy relationships.

6.3 Notations and problem statement

In the rest of this paper, we consider a set Σ of species and a gene family $\mathcal{G}$ where each gene x belongs to a species $s(x)$ of Σ. We generalize the notation s to subsets of genes: if $U \subset \mathcal{G}$, $s(U) = \{s(x) : x \in U\}$.

As we only consider rooted trees, we will sometimes omit the word "rooted". Let T be a tree. We denote by $r(T)$ its root and by $L(T)$ its set of leaves. For any internal node x of T, we denote by T_x the subtree of T rooted at x. We say that a node y is an *ancestor* of x if the two nodes belong to the same path from a leaf to the root of T, and y is closer to the root. Two nodes x and y are *unrelated* if $x \neq y$ and none is the ancestor of the other. For a set of leaves $U \subset L(T)$, we denote by $lca_T(U)$ the *least common ancestral node* of U in T, i.e. the common ancestral node of the elements of U which is the farthest from the root.

Let L' be a subset of $L(T)$. The *restriction* $T|_{L'}$ of T to L' is the tree with leaf set L' obtained from $T_{lca_T(L')}$ by removing all leaves that are not in L', and all internal nodes of degree 2, except the root. Let T' be a tree such that $L(T') = L' \subset L(T)$. We say that T *displays* T' iff $T|_{L'}$ is label-isomorphic to T'.

A triplet is a binary tree on a set L with $|L| = 3$. For $L = \{x, y, z\}$, we denote by $xy|z$ the unique triplet t on L with root $r(t)$ for which $lca_t(x, y) \neq r(t)$ holds. We denote by $tr(T) = \{T|_L : L \in \binom{L(T)}{3}$ and $T|_L$ is binary$\}$ the

set of all rooted triplets of a tree T.

Evolution of species and genes: A *species tree* S for Σ is a rooted tree whose leaves are in bijection with Σ, representing the evolutionary relationships between the species: an internal node is an ancestral species at the moment of a speciation event, and its children are the descendants. Although species trees are generally binary, we do not make this assumption here. Genes of $\mathcal{G}$ undergo speciation when the species to which they belong do. Within a species, genes can be duplicated or lost. A *history* H for $\mathcal{G}$ is a tree representing the evolution of the gene family through speciations and duplications: each leaf of H is labeled by a gene of $\mathcal{G}$, and each internal node refers to an ancestral gene at the moment of an event (speciation or duplication). Therefore each internal node of H can be labeled as a speciation ($Spec$) or duplication (Dup) event.

As H is a history "embedded" in the species tree S of Σ, it must reflect a speciation history *consistent* with S: any speciation node of H should reflect a clustering of species in agreement with S. To formally define consistency, let first introduce a more general set of labeled trees. We call a *DS-tree* for $\mathcal{G}$ a pair (G, ℓ), where G is a tree with $L(G) = \mathcal{G}$, and ℓ is a function $\ell : V(G) \backslash L(G) \to \{Dup, Spec\}$ labeling each internal node of G as a duplication or a speciation node. For simplicity, we often refer to G as the DS-tree for $\mathcal{G}$ without explicitly stating ℓ, and assume the internal nodes of G are labeled Dup or $Spec$. For some $X \subseteq L(G)$, we implicitly assume that the internal nodes of $G|_X$ share the same label as in G.

Definition 6.1. *Let G be a DS-tree for $\mathcal{G}$ and S be a species tree for Σ. We say that G is* consistent *with S if and only if, for any speciation node x of G and any two children y, z of x, $lca_S(s(L_y))$ and $lca_S(s(L_z))$ are unrelated in S, where L_y and L_z are the leaf-sets of G_y and G_z respectively.*

Now a history H for $\mathcal{G}$ is simply a DS-tree for $\mathcal{G}$ consistent with the

species tree S for Σ. Denote

$$tr_S(G) = \{s(x)s(y)|s(z) : xy|z \in tr(G) \text{ is rooted at a speciation and } s(x) \neq s(y)\}$$

The triplets of $tr_S(G)$ are called *speciation triplets*. The following theorem, which is a reformulation of Theorem 6 in [82], relates consistency to speciation triplets.

Theorem 6.2. *Let G be a DS-tree for $\mathcal{G}$ and S be a species tree for Σ. Then G is consistent with S if and only if S displays all triplets of $tr_S(G)$.*

From the Fitch [60] terminology, two leaves x and y of a history are *orthologous* if $lca_H(x, y)$ is a speciation node, and *paralogous* otherwise. We extend this terminology to a more general DS-tree.

Definition 6.3. *Let G be a DS-tree for $\mathcal{G}$. Then two genes x, y of $\mathcal{G}$ are* orthologous *with respect to (w.r.t.) G if $lca_G(x, y)$ is a speciation node, and* paralogous *w.r.t. G if $lca_G(x, y)$ is a duplication node.*

Therefore a DS-tree *induces* a set of orthology/paralogy relationships between genes.

Constraint graph: A *constraint* is simply an unordered pair of genes $\{x, y\} \in \binom{\mathcal{G}}{2}$. A set of orthology/paralogy constraints on $\mathcal{G}$ (or simply a constraint set) is a pair $C = (C_O, C_P)$ of subsets $C_O, C_P \subset \binom{\mathcal{G}}{2}$ such that $C_O \cap C_P = \emptyset$. C_O represents the *orthology constraints* and C_P the *paralogy constraints*. We say that C is a *full constraint set* if $C_O \cup C_P = \binom{\mathcal{G}}{2}$. For example, a history H for $\mathcal{G}$ induces a full constraint set.

We represent a constraint set $C = (C_O, C_P)$ by an edge-bicoloured undirected graph $R = (V, E, U)$, called a *constraint graph*, with vertex set $V = \mathcal{G}$, and two edge sets $E = C_O$ and $U = \binom{\mathcal{G}}{2} \setminus (C_O \cup C_P)$. Said differently, two genes are linked by an edge of E if they are constrained by orthology, unlinked if they are constrained by paralogy, and linked by a "special" edge of

U if the relation between the two genes is unknown. We refer to the edges of E as the *orthology edges*, to those in U as the *unknown edges* and we refer to the unlinked pairs of genes as the *paralogy non-edges*. An example of a partial constraint graph is given in Figure 6.1.

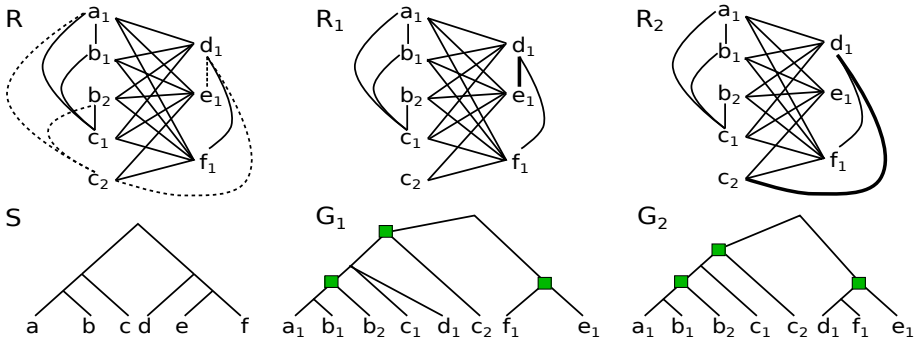

Figure 6.1: Constraint graph, satisfiability and consistency. A constraint graph $R = (V, E, U)$ with E and U depicted by solid and dotted edges, along with with two satisfiable realizations $R_1 = R(\{d_1e_1\})$ and $R_2 = R(\{d_1c_2\})$. The gene names correspond to their respective species in the species tree S. The tree G_1 is a DS-tree that satisfies R_1 while G_2 is a DS-tree that satisfies R_2. Duplication nodes in G_1 and G_2 are indicated by a green square. G_1 is not consistent with S because for instance, G_1 has the speciation triplet $s(a_1)s(d_1)|s(e_1) = ad|e$ while S has the $de|a$ triplet. The tree G_2 is consistent with S.

If C is a full constraint set then $U = \emptyset$. R is called a *full constraint graph* in this case. The *complement* of R is the graph $\overline{R}$ obtained by the alternative choice of linking paralogs instead of orthologs. Formally, $\overline{R} = (V, \overline{E} \setminus U, U)$, where $\overline{E}$ is the complement of E defined by $e \in \overline{E}$ iff $e \notin E$. Notice that R and $\overline{R}$ share the same set U of unknown edges. We denote by $R[X]$ the graph R *induced* by $X \subseteq V$, i.e. $R[X] = (X, E(X), U(X))$ where $E(X)$ (resp. $U(X)$) are the edges of E (resp. U) having both endpoints in X. Note that if $U = \emptyset$, $R[X]$ corresponds to the usual definition of the graph induced by X.

Satisfiability and consistency Given a constraint set C (or similarly a constraint graph R), is C *possible*, i.e. can we find a history for $\mathcal{G}$ inducing the orthology and paralogy constraints of C? As an orthology constraint for two genes belonging to the same genome cannot be induced by a history for $\mathcal{G}$, we assume in the rest of this paper that the set $\tilde{C}_P = \{\{x, y\} \in \binom{\mathcal{G}}{2} : s(x) = s(y)\}$ is included in C_P. A *trivial set of paralogy constraints* is a set C_P restricted to $\tilde{C}_P$.

The question whether C is possible is in two parts: (1) is C *satisfiable*, i.e. can we find a DS-tree G inducing C and (2) is there such a G which is consistent with a species tree? Formal definitions follow.

Definition 6.4. *Let $R = (V, E, U)$ be a constraint graph and G be a DS-tree with $L(G) = V$. We say that G* satisfies *R if for two genes $x, y \in L(G)$, if $xy \in E$ then they are orthologous w.r.t. G, and if $xy \in E \setminus U$ then they are paralogous w.r.t. G. We say that R is* satisfiable *if there exist a DS-tree G that satisfies R.*

If $U \neq \emptyset$, then R being satisfiable means that we can make a choice for the unknown edges as orthology edges and paralogy non-edges to obtain a full constraint graph that is satisfiable. For $F \subset U$, the *realization* of R by F corresponds to choosing F as orthology edges, and $U \setminus F$ as paralogy non-edges, leading to the full constraint graph $R(F) = (V, E \cup F, \emptyset)$. We call $R(\emptyset)$ and $R(U)$ the *empty* and *full* realizations, respectively.

As a history is a DS-tree, a set of constraints that is not satisfiable is clearly not possible, i.e. there is no history that depicts the orthology/paralogy relationships given by the constraints. Moreover, satisfiability is not sufficient to ensure the possibility of such an history, as a DS-tree is not always consistent with a species tree. Figure 6.1 shows an example of a constraint graph R along with two satisfying realizations R_1 and R_2. However, R_1 cannot be made consistent with a given species S whereas R_2 can.

Definition 6.5. *Let R be a constraint graph for $\mathcal{G}$. We say that R is* consistent with a species tree S *if and only if there is a realization of R which is satisfiable by a DS-tree G which is consistent with S. More generally, we say that R is* consistent *if and only if there is a species tree S such that R is consistent with S.*

The three following sections are respectively dedicated to the three following questions: (1) Given a constraint graph $R = (V, E, U)$, is R satisfiable? (2) Given a satisfiable constraint graph $R = (V, E, U)$, and a species tree S, is R consistent with S? (3) Given a satisfiable constraint graph $R = (V, E, U)$, is R consistent, i.e. can we find a species tree S such that R is consistent with S?

6.4 Satisfiability of a constraint graph

The problem of constraint graph satisfiability has been addressed in [80] in the restricted case of a full set of constraints. The following theorem is a reformulation of one of the main results of this paper.

Theorem 6.6 ([80])**.** *A full constraint graph R is satisfiable if and only if R is P_4-free (or equivalently, iff $\overline{R}$ is P_4-free since P_4 is self-complementary), meaning that no four vertices of R induce a path of length 4.*

Consider now the general case of a constraint graph $R = (V, E, U)$ with $U \neq \emptyset$. Then the problem is to find a realization $R(F)$ that is itself satisfiable, i.e. P_4-free. It turns out that this problem is a reformulation of the well-known *Graph sandwich problem* for P_4-free graphs. It can be stated as follows : given two graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$, with $E_1 \subseteq E_2$, does there exist a P_4-free graph $G = (V, E)$ such that $E_1 \subseteq E \subseteq E_2$. That is, G must contain every edge of G_1 and every non-edge of G_2. It is then clear that this is equivalent to finding a P_4-free realization of $R = (V, E_1, U = E_2 \setminus E_1)$. A $O(|V|^3)$ algorithm was proposed in [66] to solve this problem. In this section,

we restate under our formalism some of the useful results of this paper, and give a modified version of the proposed algorithm that outputs a DS-tree that satisfies R whenever there is one. We will make use of the following well-known characterization of P_4-free graphs.

Lemma 6.7. *A graph Γ is P_4-free if and only if, for any subset X of vertices of Γ with $|X| \geq 2$, either $\Gamma[X]$ or $\overline{\Gamma[X]}$ is disconnected.*

The next lemmata establish an important *heritability* property on satisfiable graphs: every restriction $R[X]$ of R must be satisfiable for R to be satisfiable.

Lemma 6.8. *Let G be a DS-tree that satisfies a realization $R(F)$, for some $F \subseteq U$. Let $X \subseteq V$ and let $F_X = \{ab \in F : a, b \in X\}$. Then $G|_X$ is a realization of $R[X](F_X)$.*

Proof. Let $a, b \in X$. First observe that a and b have the same orthology/paralogy relationship in $R(F)$ and $R[X](F_X)$. Let $c = lca_G(a, b)$. Now, c is also an internal node of $G|_X$, and moreover $c = lca_{G|_X}(a, b)$. As c has the same *Dup* or *Spec* label as in G, c correctly displays the relationship between a and b. Thus $G|_X$ satisfies every relationship in $R[X](F_X)$. □

The heritability property is then immediate.

Lemma 6.9. *A constraint graph $R = (V, E, U)$ is satisfiable if and only if for any $X \subseteq V$, $R[X]$ is satisfiable.*

Theorem 6.10. *[66] A constraint graph R is satisfiable if and only if at least one of the two following holds :*

1. *$R(\emptyset)$ is disconnected, and all of its connected components are satisfiable;*
2. *$\overline{R(U)}$ is disconnected, and all of its connected components are satisfiable.*

Proof. $\Leftarrow$ For 1.: Suppose $R(\emptyset)$ is disconnected. Let $\{R_1, \ldots, R_k\}$ be the connected components of $R(\emptyset)$ with $k > 1$, all being satisfiable. As each R_i is satisfiable, there is a DS-tree G_i that satisfies a realization $R_i(F_i)$ of R_i. Let $F = \cup_{1 \leq i \leq k} F_i$. Then the realization $R(F)$ of R is a full constraint graph with k full constraint components $R_i(F_i)$ in which no two components share an edge. In other words, there is a paralogy non-edge between each pair of genes from two different components. Therefore, joining the roots of $G_1, \ldots, G_k$ under a common duplication node yields a DS-tree for $R(F)$, which shows that R is satisfiable. The proof when 2. holds is the same, except that we root G at a speciation node since the components of $\overline{R(U)}$ are pairwise-complete in $R(U)$.

$\Rightarrow$ Suppose that both conditions do not hold. If $R(\emptyset)$ or $\overline{R(U)}$ has a component that is not satisfiable, then by Lemma 6.9, R is not satisfiable. So instead suppose that each of $R(\emptyset)$ and $\overline{R(U)}$ has a single connected component. Let $F \subseteq U$. The realization $R(F)$ of R must be connected as $R(\emptyset)$ is already connected and $E(R(\emptyset)) \subseteq E(R(F))$. $\overline{R(F)}$ must also be connected, as choosing all edges of U leaves $\overline{R(U)}$ connected and $E(\overline{R(U)}) \subseteq E(\overline{R(F)})$. Since both $R(F)$ and $\overline{R(F)}$ are connected, by Lemma 6.7 $R(F)$ is not P_4 free, and thus not satisfiable by Theorem 6.6. As this is true for any realization $R(F)$ of R, i.e. for any $F \subseteq U$, it follows that R is not satisfiable. $\square$

Theorem 6.10 suggests the recursive algorithm BuildDSTree that begins by finding out if one of $R(\emptyset)$ or $\overline{R(U)}$ is disconnected. If so, it creates a node of the appropriate type, gives it the identified components as children and repeats the process on each such component.

If n is the number of genes in $\mathcal{G}$, the algorithm creates a DS-tree G with at most $n - 1$ internal nodes (or stops before if R is unsatisfiable). For each such internal node v, the time taken to go through the algorithm is dominated by (at most) two depth-first searches that are performed on $L(G_v)$, and the rest of the work is handled by children nodes. So the time taken to handle v is bounded by the number of edges/non-edges in $R[L(G_v)]$,

which is $O(|L(G_v)|^2) \subseteq O(n^2)$. So in total is $O(n^3)$.

ALGORITHM BUILDDSTREE $(R = (V, E, U),\ v)$
where R is a (possibly induced) constraint graph and v is the current node of G we are creating

IF $|V| = 1$; RETURN;
$R(\emptyset) = (V, E, \emptyset)$
Find the connected components CC of $R(\emptyset)$ through
a depth-first search
IF $|CC| > 1$;
    $type \leftarrow Dup$
ELSE
    $\overline{R(U)} = (V, \overline{E}, \emptyset)$
    $type \leftarrow Spec$
    Find the connected components CC of $\overline{R(U)}$ through
    a depth-first search
    IF $|CC| = 1$; output "Unsatisfiable", and halt the recursion
END IF
$v.type \leftarrow type$
FOR $C \in CC$;
    Add child node v_C to v
    $BuildDSTree(R[C], v_C)$
END FOR
RETURN

6.5 Consistency with a given species tree

Let $R = (V, E, U)$ be a constraint graph for $\mathcal{G}$ and S be a species tree for Σ. We want to know whether the orthology/paralogy constraints represented by R can be induced by a history for $\mathcal{G}$ consistent with S. More precisely, is there a realization $R(F)$ of R that is satisfiable and such that the DS-tree satisfying $R(F)$ is consistent with S? If R is not satisfiable, then the answer is clearly no. Therefore hereafter we assume that R is satisfiable. We first show that the problem at hand still has the heritability property.

Lemma 6.11. *R is consistent with S if and only if for any $X \subseteq V$, $R[X]$ is consistent with S.*

Proof. The '$\Leftarrow$' part is trivial since we can choose $X = V$ to show that R is consistent with S. Conversely, assume R is consistent with S. Let G be a DS-tree for some realization of R such that G is consistent with S, and let $X \subseteq V$. By Lemma 6.8, $G|_X$ is a DS-tree for $R[X]$. Let $ab|c \in tr_S(G|_X)$. Since going from $G|_X$ to G only involves adding subtrees on branches of $G|_X$, it follows that $ab|c \in tr_S(G)$. Therefore, $tr_S(G|_X) \subseteq tr_S(G)$. Now, since S displays $tr_S(G)$, $G|_X$ is a realization of $R[X]$ that is consistent with S.

□

We need to introduce one last notation before stating the main theorem for characterizing consistency of a constraint graph R with a species tree S. Let $R(F)$ be a realization of R, and let $CC = \{R_1, \ldots, R_k\}$ be the connected components of $\overline{R(F)}$. Notice that the components of CC are pairwise complete in $R(F)$. A *speciation partition* $P = \{P_1, \ldots, P_{|P|}\}$ is a non-trivial partition of CC (i.e. $|P| > 1$) such that $lca_S(s(P_i))$ is unrelated to $lca_S(s(P_j))$ whenever $i \neq j$.

Theorem 6.12. *R is consistent with S if and only if at least one of the following conditions holds:*

1. *$R(\emptyset)$ is disconnected and each connected component is consistent with S;*

2. *$\overline{R(U)}$ is disconnected, its components admit a speciation partition and each component in this partition is consistent with S.*

Proof. $\Leftarrow$ For 1., Let $\{R_1, \ldots, R_k\}$ be the connected components of $R(\emptyset)$, each R_i having a DS-tree G_i consistent with S. We can then join the roots of $G_1, \ldots, G_k$ under a common duplication parent. This yields a DS-tree G that satisfies R as each pair of components of $R(\emptyset)$ are related by paralogy. Furthermore, all rooted triplets of G that were not in any G_i are rooted at $r(G)$, a *Dup* node. Therefore, $tr_S(G) = \cup_{1 \leq i \leq k} tr_S(G_i)$, which S displays.

$\Leftarrow$ 2.: Let $P = \{P_1, \ldots, P_k\}$ be a non-trivial speciation partition of the connected components of $\overline{R(U)}$. By assumption every $P_i \in P$ has a DS-tree G_i that is consistent with S, implying that S displays $\cup_{1 \leq i \leq k} tr_S(G_i)$. Here all elements of P are components of R that are pairwise-complete, and we obtain a DS-tree G for R by joining $G_1, \ldots, G_k$ under a common speciation parent. Let $T = tr_S(G) \setminus \cup_{1 \leq i \leq k} tr_S(G_i)$. Every triplet of T is rooted at $r(G)$. Thus if three genes a, b, c of $L(G)$ form a speciation triplet $s(a)s(b)|s(c) \in T$, then a and b are in some part P_i while c is in another part P_j. But by the definition of speciation partitions, $lca_S(s(P_i))$ is unrelated to $lca_S(P_j)$, implying that $s(a)s(b)|s(c) \in tr(S)$. It follows that S displays T.

$\Rightarrow$: suppose both conditions are not met, but that R is consistent with S. If $R(\emptyset)$ is disconnected but has an inconsistent component, then R is inconsistent by Lemma 6.11. So we assume $R(\emptyset)$ is connected. If $\overline{R(U)}$ is also connected, then we saw in Theorem 6.10 that R is not even satisfiable. If $\overline{R(U)}$ is disconnected and admits a speciation partition, but a member of this partition is not consistent, then again by Lemma 6.11, R is not consistent. So we assume that $R(\emptyset)$ is connected, and $\overline{R(U)}$ is disconnected but admits no speciation partition. Let G be a DS-tree for R consistent with S. Suppose $r(G)$ is a duplication node and let r_1, r_2 be two children of $r(G)$.

We have that every gene in $L(G_{r_1})$ is paralogous with every gene in $L(G_{r_2})$ and vice-versa. This implies that $L(G_{r_1})$ and $L(G_{r_2})$ are two components of $R(\emptyset)$ that share no edge, a contradiction since we assume $R(\emptyset)$ is connected. So $r(G)$ is a speciation node. Let $r_1, \ldots, r_k$ be the children of $r(G)$. The sets $L(G_{r_1}), \ldots, L(G_{r_k})$ form a partition P of the connected components of $\overline{R(U)}$. Since S displays $tr_S(G)$, it follows that for two distinct $P_i, P_j \in P$, $lca_S(s(P_i))$ and $lca_S(s(P_j))$ are unrelated. Hence P is a speciation partition, a contradiction. □

This theorem suggests a small modification to algorithm BUILDDSTREE. Connected components of $R(\emptyset)$ are handled in the same manner, but in the case of a disconnected $\overline{R(U)}$, we need to find a speciation partition P after having found its connected components CC. To accomplish this, it suffices to observe that some $C_1, C_2 \in CC$ must be in the same part of P when $lca_S(s(C_1))$ is on the path from $lca_S(s(C_2))$ to the root of S (or vice-versa). Thus for each $P_i \in P$, we can find the member of $C \in P_i$ that has $lca_S(s(C))$ the closest to the root of S, then any other component C' having $lca_S(s(C'))$ in the subtree rooted at $lca_S(s(C))$ will be in P_i. FINDSPECIATIONPARTITION uses that fact to find P through a pre-order traversal of S.

ALGORITHM FINDSPECIATIONPARTITION(CC, s, P, P_i)
where CC is the set of components to partition, $s \in V(S)$ is the current node of S in the pre-order traversal, P is the partition of CC, and P_i is the current part of P we are adding components to

```
FOR C ∈ CC such that lca_S(C) = s;
    IF P_i is not set; let P_i be a new empty set and add P_i to P
    Add C to P_i
END FOR
FOR s' ∈ children(s);
    FindSpeciationPartition(CC, s', P, P_i)
END FOR
```

Assuming constant time lca lookups, we can precompute $lca_S(s(C))$ in time $|C|$ for each $C \in CC$. If CC has a total of k nodes, by mapping each $s \in S$ to the list of $C \in CC$ with $lca_S(s(C)) = s$, the whole algorithm takes time $O(k + |S|)$. We need to call this algorithm in up to $n - 1$ calls of BUILDDSTREE. We argued that one call on a node v of G in BUILDDSTREE takes time $O(|L(G_v)|^2)$, so adding this step makes it $O(|L(G_v)|^2 + |S| + k)$. Noting that $k = |L(G_v)|$, and assuming that $|L(G_v)| \geq |S|$, this modified algorithm still runs in time $O(n^3)$, where $n = |\mathcal{G}|$.

6.6 Consistency of a satisfiable constraint graph

Now let $R = (V, E, U)$ be a constraint graph for $\mathcal{G}$ and suppose the species tree for Σ is unknown. The question is to know whether the graph R is consistent, and if so to output a species tree S such that R is consistent with S. As above, we assume that R is satisfiable. Note that unlike the two previous problems, we cannot treat each connected component of $R(\emptyset)$ or

$\overline{R(U)}$ independently, as two (or more) components might give gene histories consistent by themselves but not together.

Consider now a full constraint graph R. The results in [80, 82] suggest a polynomial-time algorithm for solving the consistency problem that consists in building a DS-tree G satisfying R, extracting all speciation triplets of G and checking their consistency with a species tree. Here we propose an alternative polynomial-time algorithm for the same problem, avoiding the first step of a DS-tree construction. We first introduce the following subset $P_3(R)$ of triplets of $\binom{V}{3}$ inducing a path of size 3 in R:

$$P_3(R) = \{s(x)s(y)|s(z) : \{x,y,z\} \in \binom{V}{3},\ zx, zy \in E \text{ and } xy \notin E \cup U\}$$

Notice that $s(x)s(y)|s(z) \in P_3(R)$ implies that any DS-tree G satisfying R has $s(x)s(y)|s(z) \in tr_S(G)$. Indeed, since $xy \notin E \cup U$, $lca_G(x,y)$ is a duplication node. And since both x and y are related to z by speciation, $lca_G(x,z) = lca_G(y,z)$ and $xy|z$ must be a speciation triplet of G.

For example, consider the vertices b_1, c_2, e_1 of the R graph in Figure 6.1, which form a path of length 3 with e_1 in the center. In the DS-tree G_1, $lca_{G_1}(b_1, c_2)$ is a duplication, and $lca_{G_1}(\{b_1, c_2, e_1\})$ is a speciation. Restricting G_1 to the three vertices yields the triplet $b_1c_2|e_1$ rooted at a speciation, and therefore, $s(b_1)s(c_2)|s(e_1) \in tr_S(G_1)$. The same holds for the $s(c_1)s(c_2)|s(e_1)$ triplet implied by the P_3 induced by c_1, c_2, e_1. Notice however that in both DS-trees, $s(b_1)s(c_1)|s(e_1)$ is a speciation triplet, though b_1, c_1, e_1 do not induce a P_3. We show that this kind of speciation triplet is implied by the other two aforementioned P_3, and that the P_3 subgraphs actually imply every mandatory speciation triplet.

Theorem 6.13. *Let $R = (V, E, U = \emptyset)$ be a satisfiable full constraint graph. Then R is consistent if and only if there exists a species tree S displaying all the triplets of $P_3(R)$.*

Proof. $\Rightarrow$: since $s(x)s(y)|s(z) \in P_3(R)$ implies that $s(x)s(y)|s(z) \in tr_S(G)$,

it follows that any species tree S consistent with R must display every triplet of $P_3(R)$.

$\Leftarrow$: we first obtain a least-resolved DS-tree G for R in terms of speciation. Let G' be a consistent DS-tree satisfying R, and let S be a species tree displaying $P_3(R)$. If G' has any speciation node v that has a speciation child w, we obtain G'' by contracting v and w (delete w and give its children to v). Since v and w are both speciations, this operation does not change the label of $lca_G(x, y)$ for any two leaves x and y and G'' still satisfies R. Moreover, $tr_S(G'') \subset tr_S(G')$, so there is no risk of breaking consistency. We obtain a DS-tree G by repeating this operation until we cannot find such a v and w.

Let $xy|z$ be a triplet of G rooted at a speciation node. We have that $lca_G(z, x) = lca_G(z, y)$ is a speciation, and that $zx, zy \in E$. If $lca_G(x, y)$ is a duplication node, then $xy \notin E$. So $\{x, y, z\}$ induces a P_3 in R, and S displays $s(x)s(y)|s(z)$. Suppose instead that $lca_G(x, y)$ is a speciation node. Because G is a least resolved DS-tree, there must be a duplication node u on the path between $lca_G(x, y)$ and $lca_G(x, z)$. This implies there is a leaf d in G_u such that x and y are related to d by duplication, but with d and z related by speciation. In R, we then have $zd \in E$, and $xd, yd \notin E$. Thus both $\{x, d, z\}$ and $\{y, d, z\}$ induce a P_3 in R with z being the middle vertex, and $s(x)s(d)|s(z), s(y)s(d)|s(z) \in P_3(R)$ are both displayed by S. This is only possible if there is a node in S that has all of $s(x), s(y), s(d)$ in one child subtree and $s(z)$ in another. Therefore, S must display $s(x)s(y)|s(z)$. Having taken care of both types of speciation triplets, we deduce that displaying $P_3(R)$ is sufficient to display $tr_S(G)$. □

Therefore the consistency problem for a full constraint graph reduces to the problem of verifying whether the set $P_3(R)$ of triples can be displayed in a species tree for Σ. This is in fact a well know problem with a solution presented in [144]: given a triplet set $\mathcal{R}$, there is a polynomial-time algorithm, called BUILD [1], that, when applied to $\mathcal{R}$ either outputs a species tree that displays $\mathcal{R}$ or recognizes that $\mathcal{R}$ is inconsistent. Therefore, in the case of a

full constraint graph, the consistency problem is resolved in polynomial time by first constructing the set $P_3(R)$, and then applying the BUILD algorithm.

Consider now the general case of a constraint graph $R = (V, E, U)$ with $U \neq \emptyset$. The branch-and-bound algorithm CHECKCONS iterates over the edges of U, tries to make them edges and non-edges but stops as soon as one decision creates a set of P_3 that is inconsistent. Since at worst, every possibility is tested, it follows that this algorithm is exact, though exponential in the worst case.

ALGORITHM CHECKCONS $(R = (V, E, U))$
Obtain a species tree S by running BUILD on $P_3(R)$
IF S is not set (i.e. BUILD failed), RETURN FALSE
IF R is not satisfiable, RETURN FALSE
IF $U = \emptyset$, return (R, S)
Let $e \in U$ and let $R_e = (V, E \cup \{e\}, U \setminus \{e\})$
$(R', S) \leftarrow CHECKCONS(R_e)$
IF (R', S) is set (i.e. CHECKCONS succeeded), RETURN (R', S)
Otherwise let $R_{\overline{e}} = (V, E, U \setminus \{e\})$
$(R', S) \leftarrow CHECKCONS(R_{\overline{e}})$
IF (R', S) is set (i.e. CHECKCONS succeeded), RETURN (R', S)
Otherwise RETURN FALSE

Possible improvements of this algorithm include removing as many edges from U as possible, and choosing an ordering of the edges that may speed up the branch-and-bound process. For instance, it may be worthwhile to first identify every induced P_4 of $R(\emptyset)$. The P_4 subgraphs that admit only one possibility for removal, i.e. the P_4 can only be removed by making a unique edge $e \in U$ an orthology edge, can be corrected before entering the

algorithm. Note that the same applies for the edges of U that must be edges of $\overline{R(U)}$. We may then prioritize the handling of the other P_4 by considering the edges that resolve them first. Similarly, it would also be possible to identify edges of U that are mandatory in E by finding the P_3 subgraphs of $R(\emptyset)$ that are not in $P_3(R)$, but that disagree with a triplet of $P_3(R)$. For instance, $R(\emptyset)$ might have a P_3 with edges xz, zy, but this P_3 is not in $P_3(R)$ because $xy \in U$. If say $s(y)s(z)|s(x)$ is in $P_3(R)$, then xy is forced in E as otherwise the contradictory $s(x)s(y)|s(z)$ triplet would be present.

6.7 Experiments

We show how the developed algorithms for checking satisfiability and consistency can be used, in combination with an orthology detection tool such as ProteinOrtho [105], to infer a robust set of orthology and paralogy constraints. Given a set of protein sequences, Proteinortho infers homologous gene families as well as orthology relationships within these families, based on various similarity scores. Proteinortho does not infer paralogy relationships. However, if we choose a set of parameters leading to a loose characterization of orthologs, then we can assume that unpredicted constraints should represent paralogy. Different combinations of parameters therefore lead to different constraint sets that can be analyzed for satisfiability and consistency.

ProteinOrtho has been run on 265 gene families of vertebrates, each representing the leaf-set of an *Ensembl* [61] gene tree. Trees were chosen randomly among the *Ensembl* gene trees containing at least 20 leaves. For each family, five different parameter settings, numbered from -2 to $+2$, were tested, 0 representing the default parameter choice of ProteinOrtho, and the smaller the number, the looser is the induced characterization of orthology. For each parameter setting i, we define the full constraint graph R^i where all gene pairs not predicted as orthologs are interpreted as paralogs. Typically, a graph R^- for a negative number (-1 or -2) contains more orthology (and thus less paralogy) constraints than R^0, while the converse is true for a graph

R^+. Combining two constraint graphs R^- and R^+ consists in keeping only orthology and paralogy edges that are common to both, and completing the graph with unknown edges.

Table 6.I summarizes the results on satisfiability and consistency with the *Ensembl* species tree S, obtained for each gene family and each parameter setting or combination. Among the 265 gene families, only 112 (42%) produced at least one satisfiable full constraint graph and only 44 (15%) produced such a graph which is also consistent with the *Ensembl* species tree. However, combining loose and strict parameter settings lead to much better results with at least 95% satisfiability and 56% consistency with S. The partial orthology/paralogy constraints obtained from combinations correspond to about half of the constraints of a full graph, as illustrated by the last column of the table.

	# satisfiable families	# consistent families	% constraints when consistent
-2	82 (30.9%)	30 (11.3%)	
-1	44 (16.6%)	13 (4.91%)	
0	26 (9.81%)	9 (3.40%)	
+1	48 (18.1%)	14 (5.28%)	
+2	55 (20.8%)	18 (6.79%)	
-2/+2	260 (98.1%)	172 (64.9%)	42.0%
-2/+1	258 (97.4%)	172 (64.9%)	44.8%
-1/+1	254 (95.8%)	149 (56.2%)	50.6%
-1/+2	255 (95.9%)	157 (59.2%)	47.5%

Tableau 6.I: The results over 265 gene families from Ensembl. The first five rows correspond to the full constraint sets obtained from ProteinOrtho for the five classes of parameters. The last four rows correspond to the partial constraint sets obtained after combining two graphs over two types of parameters. The first column is the number of families for which the settings of ProteinOrtho yielded a satisfiable graph, the second that number consistent with the Ensembl species tree. The last column shows the percentage of constraints that were not unknown when a consistent solution was found (which is 100% for the first five rows).

In order to get a rough idea of the accuracy of the obtained partial orthology/paralogy predictions for each gene family $\mathcal{G}$, we compared them with those resulting from the labeling of the *Ensembl* gene tree nodes as duplication and speciation nodes. An *orthology disagreement* refers to orthology predictions on the four combined graphs depicted in Table 6.I, that are rather inferred as paralogs from the *Ensembl* gene tree labeling. A *paralogy disagreement* refers to the reverse situation. Overall, the orthology disagreement percentage is between 15.1% and 15.9% depending on the two classes of parameters combined. For paralogy disagreement, it varies between 11% and 17%, depending on the 2 parameters combined (-2/+1 and -2/+2 were

around 11.2% while -1/+1 and -1/+2 were around 17.4%).

Notice that *Ensembl* annotates many duplication nodes as "dubious". If we ignore orthology disagreements caused by a dubious duplication node, the orthology disagreement percentage drops to an average of 5.0%, strengthening the doubts on those duplication nodes.

6.8 Conclusion

In this work we have developed methods to assess the plausibility of a partial set of orthology and paralogy relationships between pairs of homologous genes. In particular, we showed how extending algorithms for the Graph sandwich problem can solve, in time $O(|V(R)|^3)$, the problems of satisfiability and consistency with a species tree. The complexity of verifying whether it is possible to construct G such that it is consistent with *some* species tree S remains open. We have elaborated on the P_3 property of R that lead to a branch-and-bound algorithm, but it remains possible that this property could be used to create a more efficient method. While previous work consisted in verifying whether a full set of relationships was satisfiable or consistent, admitting uncertainty within these relationships makes it possible to bring the theory from [82] into practice, as current orthology (or paralogy) inference methods based on sequences cannot guarantee 100% accuracy in their predictions. We show how a confidence set of such predictions can be inferred using our methods and Proteinortho. One possible application of finding solid predictions is to compare them with the relationships present on actual gene trees, then correct these trees in case of disagreement.

Competing interests

The authors declare they have no competing interests.

Author's contributions

ML, NE devised the proofs and algorithms and wrote the paper. ML implemented the software.

6.8.1 Contributions

Manuel Lafond et Nadia El-Mabrouk ont rédigé les preuves et les algorithmes de l'article. Manuel Lafond a implémenté l'algorithme et effectué les tests sur les données réelles.

CHAPITRE 7

RECONSTRUCTING A SUPERGENETREE MINIMIZING RECONCILIATION

Manuel Lafond[1] , Aïda Ouangraoua[2] , Nadia El-Mabrouk[1]

Cet article est axé sur la construction d'arbre de gènes plutôt que sur la correction. Plus précisément, on cherche à combiner des arbres déjà construits en un seul superarbre. Nous supposons ici que l'ensemble d'arbres donné est compatible, et on cherche donc un superarbre qui les contient tous. Une des motivations est la suivante. Il existe plusieurs bases de données d'arbres de gènes (par exemple Ensembl Compara [165], Hogenom [122], Phog [38], MetaPHOrs [127], PhylomeDB [85], Panther[111]). Ainsi, à une famille de gènes peut correspondre plusieurs arbres, et la question de comment les combiner mène au problème des superarbres. Par ailleurs, les méthodes de regroupement en familles de gènes peuvent produire des familles trop élargies, faisant en sorte qu'on y retrouve des gènes distants et difficiles à aligner. Le résultat est que les méthodes d'inférence d'arbres de gènes sur cette famille seront plus sujettes à l'erreur. Une solution est de raffiner de telles familles en plus petits groupes de gènes, construire un arbre pour chacun d'entre eux, ensuite les "amalgamer" en un superarbre.

Il peut toutefois exister un nombre exponentiel de solutions, et on peut se demander comment en choisir une. Fidèle aux thème de cette thèse, la réponse passe ici par la réconciliation. C'est-à-dire, parmi tous les superarbres possibles, quel est celui qui minimise le nombre de duplications et pertes? On montre que trouver une réponse à cette question n'est pas facile; en fait, même si on se limite a minimiser seulement les duplications, le problème est NP-complet. Il est même difficile à approximer à un ratio $O(n^{1-\epsilon})$, où

[1]DIRO, Université de Montréal, Canada
[2]Département d'informatique, Université de Sherbrooke

n est le nombre d'étiquettes des arbres donnés et $0 < \epsilon < 1$. Ceci veut dire que pour tout algorithme en temps polynomial A, la solution proposée par A à ce problème contient parfois $\Omega(n^{1-\epsilon})$ fois plus de duplications que le minimum possible et ce, pour une infinité d'instances du problème (sauf si $P = NP$). Ce résultat s'applique aussi si on ne compte que les noeuds de duplications associés à la racine de S selon le lca-mapping. De plus, le problème reste NP-complet même si les arbres n'ont aucun gène en commun. Nous proposons un algorithme exact basé sur Build et All-min-trees nécessitant un temps $\Omega(n \cdot (n/2)^{n/2})$, un autre algorithme exact de programmation dynamique requérant un temps $O(4^n)$ et une heuristique basée sur le problème Max-Cut.

7.1 Abstract

Combining a set of trees on partial datasets into a single tree is a classical method for inferring large phylogenetic trees. Ideally, the combined tree should display each input partial tree, which is only possible if input trees do not contain contradictory phylogenetic information. The simplest version of the supertree problem is thus to state whether a set of trees is compatible, and if so, construct a tree displaying them all. Classically, supertree methods have been applied to the reconstruction of species trees. Here we rather consider reconstructing a super gene tree in light of a known species tree S. We define the supergenetree problem as finding, among all supertrees displaying a set of input gene trees, one supertree minimizing a reconciliation distance with S. We first show how classical exact methods to the supertree problem can be extended to the supergenetree problem. As all these methods are highly exponential, we also exhibit a natural greedy heuristic for the duplication cost, based on minimizing the set of duplications preceding the first speciation event. We then show that both the supergenetree problem and its restriction to minimizing duplications preceding the first speciation are NP-hard to approximate within a $n^{1-\epsilon}$ factor, for any $0 < \epsilon < 1$. Finally, we show that a restriction of this problem to uniquely labeled speciation gene trees, which is relevant to many biological applications, is also NP-hard. Therefore, we introduce new avenues in the field of supertrees, and set the theoretical basis for the exploration of various algorithmic aspects of the problems.

7.2 Introduction

A fundamental task in evolutionary biology is to combine a collection of rooted trees on partial, possibly overlapping, sets of data, into a single rooted tree on the full set of data. This is the goal of supertree methods, mainly designed and used for the purpose of reconstructing a species supertree from

a set of species trees (see overviews of early methods in [12, 13, 18], and more recent methods in [7, 35, 116, 129, 130, 152, 155]).

Ideally, the combined supertree should "display" each of the input tree, in the sense that by restricting of the supertree to the leaf set of an input tree, we obtain the same input tree. However, this is not always possible, as the input trees may contain conflicting phylogenetic information. Note that considering a set of input trees that are not all compatible leads to the questions of correcting input gene trees or finding a subset of compatible input trees or subtrees [141]. Here, we leave open these questions and study the more direct formulation of the supertree problem that is to consider a set of compatible input trees and find a supertree displaying them all. The BUILD algorithm by Aho *et al.* [1] can be used to test, in polynomial time, whether a collection of rooted trees is compatible, and if so, construct a compatible supertree, not necessarily fully resolved. This algorithm has been generalized in [34, 115] to output all compatible supertrees, and adapted in [142] to output all minimally resolved compatible supertrees.

Although supertree methods are classically applied to the construction of species trees, they can be used as well for the purpose of constructing gene trees. Several gene tree databases are available (see for example Ensembl Compara [165], Hogenom [122], Phog [38], MetaPHOrs [127], PhylomeDB [85], Panther[111]). For a gene family of interest, many different gene trees can therefore be available, and finding one single supertree displaying them all leads to a supertree question. On the other hand, given a gene of interest, a homology-based search tool is usually used to output all homologs in a set of genomes. The resulting gene family may be very large, involving distant gene sequences that may be hard to align, leading to weakly supported trees - or even worse, highly supported gene trees that are in fact incorrect. A standard way of reducing such errors is then to use a clustering algorithm based on sequence similarity, such as OrthoMCL [106], InParanoid [10], Proteinortho [105] or many others (see Quest for Orthologs links at

http://questfororthologs.org/), to group genes into smaller sets of orthologs or inparalogs (paralogs that arose after a given speciation). Trees obtained for such partial gene families can then be combined by using a supertree method.

Considering input trees as parts of gene trees rather than as parts of species trees does not make any difference regarding the compatibility test procedure. However, for reconstructing a compatible "super gene tree", if a species tree is known for the taxa of interest, then it can be used as an additional information to choose among all possible supertrees displaying the input partial gene trees. Indeed, a natural optimization criterion is to minimize the reconciliation cost, i.e. either the duplication or the duplication plus loss cost, induced by the output tree. We call the problem of finding a compatible supertree minimizing a reconciliation cost *the supergenetree problem*.

In this paper, we first show how the exact methods developed for the supertree problem can be adapted to the supergenetree problem. As for the original algorithms, all the extensions have also exponential worst-time complexity. We then exhibit a heuristic, which can be seen as a greedy approach classically used for the supertree problems, that consists in constructing progressively the tree from its root to its leaves. The main module of this heuristic is to infer the minimum number of duplications preceding the first speciation, which we call the *Minimum pre-Speciation Duplication* problem. We show that the supergenetree problem for the duplication cost, and even its restricted version the Minimum pre-Speciation Duplication problem, are NP-hard to approximate within a $n^{1-\epsilon}$ factor, for any $0 < \epsilon < 1$ (n being the number of genes). Moreover, these inapproximability results even hold for instances in which there is only one gene per species in the input trees. Finally we consider the supergenetree problem with restrictions on input trees that are relevant to many biological applications. Namely, we require each gene to appear in at most one tree, and genes of any tree to be related

through orthology only. This is for example the case of gene trees obtained for OrthoMCL clusters called orthogroups [106]. We show that even for this restriction, the supergenetree problem remains NP-hard for the duplication cost.

The following section introduces preliminary notations that will be required in the rest of the paper.

7.3 Preliminaries

Notations on trees. Given a set L, a *tree T for L* is a rooted tree whose leafset $\mathcal{L}(T)$ is in bijection with L. We denote by $V(T)$ the set of nodes and by $r(T)$ the root of T. Given an internal node x of T, the subtree of T rooted at x is denoted T_x. The *degree* of an internal node x of T is the number of children of x. If T is binary, we arbitrarily set one of the two children of x as the left child x_l and the other as the right child x_r. We call $(\mathcal{L}(T_{x_l}), \mathcal{L}(T_{x_r}))$ the bipartition of a node x of degree 2 (note that the term 'bipartition' is sometimes used, in the context of unrooted trees, to denote the nodes or leaves of the two components obtained after removing a given edge. To avoid confusion, note that this is not what we mean here by 'bipartition').

A node x is an *ancestor* of y if x is on the (inclusive) path between y and the root, and we then call y a descendant of x. Two nodes x and y are *separated* in T if none is an ancestor of the other. The *lowest common ancestor* (lca) of a subset L' of $\mathcal{L}(T)$, denoted $lca_T(L')$, is the ancestor common to all nodes in L' that is the most distant from the root. The restriction $T|_{L'}$ of T to L' is the tree with leafset L' obtained from the subtree of T rooted at $lca_T(L')$ by removing all leaves that are not in L', and contracting all internal nodes of degree 2, except the root. We generalize this notation to a set of trees: For a set $\mathcal{T}$ of trees on L, $\mathcal{T}|_{L'} = \{T|_{L'} \ : \ T \in \mathcal{T}\}$. Let T' be a tree such that $\mathcal{L}(T') = L' \subseteq \mathcal{L}(T)$. We say that T displays T' iff $T|_{L'}$ is the same tree as T'.

A *triplet* is a binary tree on a set L with $|L| = 3$. For $L = \{x, y, z\}$, we

denote by $xy|z$ the unique triplet t on L with root $r(t)$ for which $lca_t(x, y) \neq r(t)$ holds.

A *polytomy* (or star tree) over a set L is a tree for L with a single internal node, which is of degree $|L|$.

A *resolution* $B(T)$ of a non-binary tree T is a binary tree respecting all the ancestral relations given by T. More precisely, $B(T)$ is a binary tree such that $\mathcal{L}(B(T)) = \mathcal{L}(T)$, and for any $u, v \in V(T)$, if u is an ancestor of v in T, then $lca_{B(T)}(\mathcal{L}(T_u))$ is an ancestor of $lca_{B(T)}(\mathcal{L}(T_v))$.

Gene and species trees. Figure 7.1 is an illustration of the notations defined in this section.

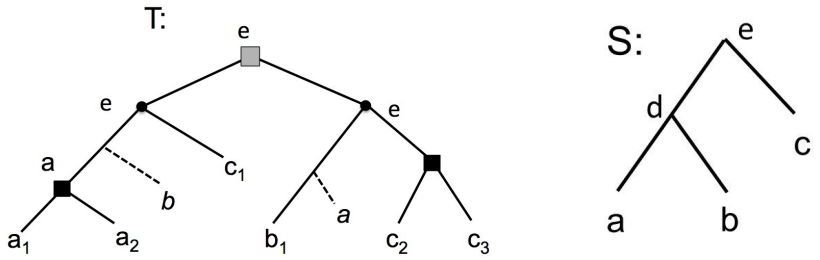

Figure 7.1: A gene tree T for the gene family $\Gamma = \{a_1, a_2, b_1, c_1, c_2, c_3\}$ and a species tree S for the set of species $\Sigma = \{a, b, c\}$ and where, for any x and any i, x_i is a gene in genome x. The label of an internal node x of T corresponds to $s(x)$. Speciation nodes are represented by circles and duplication nodes by squares. The pre-speciation duplication nodes (here only one node) are grey-colored. The dotted lines represent losses that are inferred by a most parsimonious reconciliation algorithm. The duplication cost of T is 3 and its reconciliation cost is 5.

A species tree S for a set $\Sigma = \{\sigma_1, \cdots, \sigma_t\}$ of species represents an ordered set of speciation events that have led to Σ: an internal node is an ancestral species at the moment of a speciation event, and its children are the new descendant species. Inside the species' genomes, genes undergo speciation when the species to which they belong do, but also duplications and losses (other events such as transfers can happen, but we ignore them here). A *gene*

family is a set of genes Γ accompanied with a *mapping function* $s : \Gamma \to \Sigma$ mapping each gene to its corresponding species.

Consider a gene family Γ where each gene $x \in \Gamma$ belongs to a species $s(x)$ of Σ. The evolutionary history of Γ can be represented as a *gene tree* T for Γ, which is a rooted binary tree with its leafset in bijection with Γ, where each internal node refers to an ancestral gene at the moment of an event (either speciation or duplication). The mapping function s is generalized as follows: if x is an internal node of T, then $s(x) = lca_S(\{s(x') : x' \in \mathcal{L}(T_x)\})$.

An internal node x of T is called a *speciation node* if $s(x_l)$ and $s(x_r)$ are separated in S. Otherwise, x is a *duplication node* preceding the speciation event $lca_S(s(x_l), s(x_r))$ if $lca_S(s(x_l), s(x_r))$ is an internal node of S, otherwise it is a duplication inside the extant species $lca_S(s(x_l), s(x_r))$. A duplication node x such that $s(x) = r(S)$ is called a *pre-speciation duplication* node. A gene tree T with all internal nodes being speciation nodes is called a *speciation tree*. Two genes x, y of $\mathcal{L}(\mathcal{T})$ are *orthologs in* T if their $lca_T(x, y)$ is a speciation node.

The *duplication cost* of T is the number of duplication nodes of T. It reflects the minimum number of duplications required to explain the evolution of the gene family inside the species tree S according to T. A well-known reconciliation approach [26, 28] allows to further recover, in linear time, the minimum number of losses underlined by such an evolutionary history. We refer to the minimum number of duplications and losses required to explain T with respect to S as the *reconciliation cost* of T with respect to S, or simply the reconciliation cost if there is no ambiguity on the considered trees.

Supergenetree problem statement. A set $\mathcal{G}$ of gene trees is said *consistent* if there is a tree T, called a *supergenetree* for $\mathcal{G}$ displaying each tree of $\mathcal{G}$, and *inconsistent* otherwise. A supergenetree T for $\mathcal{G}$ is said *compatible* with $\mathcal{G}$. For example, the four triplets in Figure 7.2 are consistent, and the gene tree T of Figure 7.1 is compatible with them.

However, adding the dotted tree to the set of triplets makes the gene tree set incompatible. Consistency of a set of trees can be tested in polynomial time [1]. For a consistent set of trees, the problem considered here is to find a compatible gene tree of minimum reconciliation cost with respect to a given species tree. A formal statement of the general problem follows.

Minimum SuperGeneTree Problem (MinSGT problem):
Input: A species set Σ and a binary species tree S for Σ; a gene family Γ, a set $\Gamma_{i,1\leq i\leq k}$ of subsets of Γ, and a set $\mathcal{G} = \{G_1, G_2, \cdots, G_k\}$ of consistent gene trees where, for each $1 \leq i \leq k$, G_i is a tree for Γ_i.
Output: Among all gene trees for Γ compatible with $\mathcal{G}$, one tree T of minimum reconciliation cost.

When the considered cost is the duplication cost, the problem is called the Minimum Duplication SuperGeneTree Problem (MinDUPSGT problem).

7.4 From the SuperTree to the SuperGeneTree Problem

The classical supertree problem is to state whether or not a set of partial trees are consistent, and if so construct a tree containing them all. Here, we introduce the classical methods for solving this problem, and explore natural generalizations to the supergenetree problem.

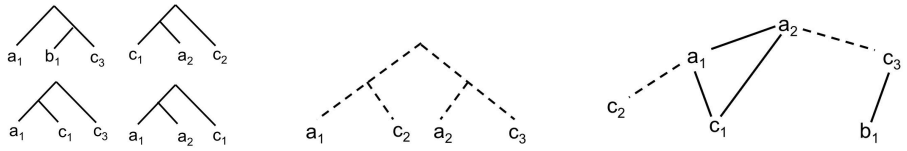

Figure 7.2: Genes trees (left and middle) and their corresponding triplet graphs (right). Plain edges of the graph correspond to the four triplet trees, while dotted edges correspond to the triplets of the four-leaves tree.

Let Γ be a set of n taxa (usually species in case of the supertree problem,

and genes in case of the supergenetree problem), $\Gamma_{i,1\leq i\leq k}$ be a set of possibly overlapping subsets of Γ, and $\mathcal{G} = \{G_1, G_2, \cdots, G_k\}$ be a set of trees where, for each $1 \leq i \leq k$, G_i is a tree for Γ_i. Let $tr(\mathcal{G})$ be the set of triplets of $\mathcal{G}$ defined as $tr(\mathcal{G}) = \{xy|z : \exists\ 1 \leq i \leq k$ such that $G_i|_{\{x,y,z\}} = xy|z\}$. Let $\mathcal{T}(\Gamma, E)$ be the triplet graph with the set of vertices Γ and the set of edges $E = \{xy :\ \exists\, z \in \Gamma$ such that $xy|z \in tr(\mathcal{G})\}$ (see Figure 7.2 for an example).

The classical BUILD algorithm [1] determines, in polynomial time, whether a set of triplets is consistent and if so constructs a tree T, possibly non-binary, compatible with them. The algorithm takes as input the graph $\mathcal{T} = \mathcal{T}(\Gamma, E)$. Let $\mathcal{C}(\mathcal{T}) = \{C_1, \cdots, C_m\}$ be the set of connected components of $\mathcal{T}$. If $\mathcal{T}$ has at least three vertices and $|\mathcal{C}(\mathcal{T})| = 1$, then $\mathcal{G}$ is inconsistent, and the algorithm terminates. For example, the set of five gene trees of Figure 7.2 is inconsistent, as the corresponding triplet graph (including dotted lines) is connected. Otherwise, if $|V(\mathcal{T})| \geq 3$, a polytomy is created over $\mathcal{C}(\mathcal{T})$, the internal node of the polytomy being the root $r(T)$ of the compatible tree T under construction and its children being m subtrees with leafsets $V(C_1), \ldots, V(C_m)$, with their topology yet to be determined (where $V(C_i) \subseteq \Gamma$ denotes the set of taxa appearing in C_i). The algorithm then recurses into each connected component, i.e. the subtree for $V(C_i)$ is determined recursively from the graph $\mathcal{T}(V(C_i), E|_{C_i})$ defined by $E|_{C_i} = \{xy :\ \exists\, z \in \Gamma$ such that $xy|z \in tr(\mathcal{G}|_{V(C_i)})\}$. If, at any step, the considered graph has a single component containing more than two vertices, then $\mathcal{G}$ is reported as an inconsistent set of trees and the algorithm terminates. Otherwise, recursion terminates when the graph has at most two vertices, eventually returning a supertree T. See Figure 7.3 for an example.

The BUILD algorithm has been generalized in an algorithm called All-Trees [115] to output all supertrees compatible with a set of triplets in case consistency holds. Instead of taking each element of $\mathcal{C}(\mathcal{T})$ as a separate leaf of $r(T)$, all possible groupings, in other words all partitions of $\mathcal{C}(\mathcal{T})$, are considered (see Figure 7.3, right, for a choice of bipartitions). For each partition

$\mathcal{P}(\mathcal{C}(\mathcal{T}))$ of $\mathcal{C}(\mathcal{T})$, a polytomy is created over $\mathcal{P}(\mathcal{C}(\mathcal{T}))$. The algorithm then iterates by considering each possible partition of each subgraph induced by each element of $\mathcal{P}(\mathcal{C}(\mathcal{T}))$. The algorithm is polynomial in the size of the output that may be exponential in the size of the input.

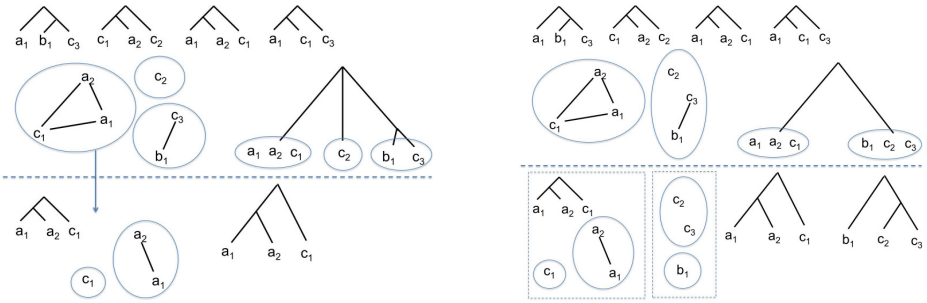

Figure 7.3: Left: Execution of the BUILD algorithm on the set of given four triplets. This example requires two iterations of the algorithm (delimited by a dotted line). At the first iteration, the triplet graph contains three components, leading to a polytomy with three leaves. The algorithm then iterates on the component $\{a_1, a_2, c_1\}$, which terminates the supertree reconstruction procedure. Notice that the gene tree of Figure 7.1, which is compatible with the four triplets, is not a resolution of this non-binary tree ; Right: A variant of the BUILD algorithm, with the triplet graph components grouped into bipartitions - in this case leading to a fully resolved tree. This tree is actually the gene tree T of Figure 7.1.

A tree T compatible with $\mathcal{G}$ such that no internal edge of T can be contracted so that the resulting tree is also compatible with $\mathcal{G}$ is called a minimally resolved supertree. Minimally resolved supertrees contain all the information about all supertrees compatible with $\mathcal{G}$ but in a "compressed" format. By exhibiting some properties on graph components, Semple shows in [142] how some partitions of the triplet graph components can be avoided without loss of generality. The new developed algorithm, named AllMinTrees [142], outputs a minimally resolved tree in polynomial time. However, it was shown in [87] that the cardinality of the solution space can be exponential in $n = |\Gamma|$, leading to an exponential time algorithm with complexity $\Omega(\frac{n}{2}^{\frac{n}{2}})$.

Notice that, in general, the trees output by all these methods are non-binary trees.

Extensions to the SuperGeneTree problem. Natural exact solutions for the supertree problem can be extended to the supergenetree problem as follows:

(1) Use AllMinTrees to output all minimally resolved supertrees, and for each one which is non-binary in general, find in linear time a resolution minimizing the reconciliation [101, 177] or duplication [176] cost. Among all optimally resolved trees, select one of minimum cost. Clearly this approach has the same complexity as the AllMinTrees algorithm, multiplied by a factor of n to resolve each tree, which is $\Omega(n \cdot \frac{n}{2}^{\frac{n}{2}})$.

(2) As we are seeking a binary tree, each created node x of the supergenetree T under construction should determine a bipartition $(\mathcal{L}(T_{x_l}), \mathcal{L}(T_{x_r}))$. Therefore, the AllTrees algorithm can be simplified by considering, instead of all partitions of $\mathcal{C}(\mathcal{T})$, only all bipartitions of the triplet graph components set. See an example in Figure 7.3, right. Notice that this simplification approach is not applicable to the AllMinTrees algorithm, as by imposing bipartitions, the minimum resolution condition cannot be guaranteed.

A branch-and-bound approach. The tree space which is explored by the two exact methods described above can be reduced by using a branch-and-bound approach. Consider for example method (1) using the AllMinTrees algorithm. At each iteration of computing one minimally resolved tree, resolve the intermediate non-binary tree obtained at this step, using for example the linear-time algorithm presented in [101]. If its reconciliation cost is greater than the cost of a full tree already obtained at a previous stage of the AllMinTrees algorithm, then stop expanding this tree as this can only increase the reconciliation cost.

A dynamic programming approach. The recursive top-down method (2) can instead be handled by a dynamic programming approach computing the minimum reconciliation cost of a tree on a subset of Γ according to the reconciliation costs of trees on smaller subsets, similarly to the wrok done in [73].

More precisely, let P be an arbitrary subset of Γ, and denote by $R(P)$ the minimum duplication cost of a tree $T|_P$ having leafset P and compatible with the set $\mathcal{G}|_P = \{G_1|_P, G_2|_P, \cdots, G_k|_P\}$. Let $\mathcal{T}(P, E|_P)$ be the BUILD graph restricted to P and $\mathcal{G}|_P$, and $\mathcal{C}(\mathcal{T}) = \{C_1, \ldots, C_m\}$ the set of its connected components. If $C \subseteq \mathcal{C}(\mathcal{T})$, by $V(C)$ we mean $\bigcup_{C_i \in C} V(C_i)$. Denote the complement of C by $\overline{C} = \mathcal{C}(\mathcal{T}) \setminus C$. Finally set $d(V(C), V(\overline{C}))$ to 0 if $s(V(C))$ and $s(V(\overline{C})$ are separated in S, in which case $(V(C), V(\overline{C}))$ is the bipartition of a speciation node, and 1 otherwise i.e. if $(V(C), V(\overline{C}))$ is the bipartition of a duplication node. Then:

$$R(P) = \min_{C \subset \mathcal{C}(\mathcal{T})} R(V(C)) + R(V(\overline{C})) + d(V(C), V(\overline{C}))$$

the value of interest being $R(\Gamma)$. First note that, assuming constant-time *lca* queries over S, $d(V(C), V(\overline{C}))$ can be computed in constant time if $s(V(C))$ and $s(V(\overline{C}))$ can be accessed in constant time, since if suffices to check that the *lca* of $s(V(C))$ and $s(V(\overline{C}))$ differs from both. To achieve this, we precompute $s(X)$ for every subset X of Γ of size $1, 2, \ldots, n$ in increasing order. Noting that if $|X| > 1$, then for any $x \in X$, $s(X) = lca_S(s(X \setminus \{x\}), s(x))$, $s(X)$ can be computed in constant time assuming that $s(X \setminus \{x\})$ was computed previously and assuming constant-time *lca* queries. As there are 2^n subsets of Γ, each computed in constant time, this preprocessing step takes time $O(2^n)$.

As for $R(\Gamma)$, we can simply ensure that each $R(P)$ is computed at most once by storing its value in a table for subsequent accesses (i.e. when $R(P)$ is needed, we use its value if it has been computed, or compute it and store

it otherwise). In this manner, each subset P takes time, not counting the recursive calls, proportional to $|P||\mathcal{G}| + |P| + 2^{|\mathcal{C}(\mathcal{T})|}$ to construct $\mathcal{T}(P, E|_P)$, find $\mathcal{C}(\mathcal{T})$, and evaluate each bipartition of $\mathcal{C}(\mathcal{T})$. We will simply use the fact that $|P||\mathcal{G}| + |P| + 2^{|\mathcal{C}(\mathcal{T})|}$ is in $O(2^n)$. As this has to be done for, at worst, each of the 2^n subsets of Γ, we get a total time $O(2^n + 2^n \cdot 2^n) = O(4^n)$. Note that this analysis probably overestimates the actual complexity of the algorithm, as we are assuming that each subset P and each component set $\mathcal{C}(\mathcal{T})$ are both always of size n. It is also worth mentioning that the $R(P)$ recurrence can easily adapted to the mutation cost (duplications + losses).

A greedy heuristic for the duplication cost. Instead of trying all partitions of the triplet graph components set at each step of the AllTrees or AllMinTrees algorithms, if the goal is to minimize the duplication cost, then a natural greedy approach would be to choose the best partition at each iteration, namely the one allowing to minimize the number of duplications preceding each speciation event. Such an approach would result in pushing duplications down the tree. It leads to the following restricted version of the supergenetree problem.

Minimum Pre-Speciation Duplication Problem (MinPreSpeDup problem):
Input: A species set Σ and a binary species tree S for Σ; a gene family Γ, a set $\Gamma_{i,1\leq i\leq k}$ of subsets of Γ, and a set $\mathcal{G} = \{G_1, G_2, \cdots, G_k\}$ of consistent gene trees where, for each $1 \leq i \leq k$, G_i is a tree on Γ_i.
Output: Among all gene trees for Γ compatible with $\mathcal{G}$, one tree T with minimum pre-speciation duplication nodes.

We will show in the following section that even this restricted version of the supergenetree problem is hard. Here, we give the intuition of a natural way of solving this problem, that reduces to repeated applications of the

Max-Cut problem. Although known to be NP-hard, efficient heuristics exist (up to a factor of 0.878 [64]), that can be used for our purpose.

For the supertree problem, the triplet graph $\mathcal{T} = \mathcal{T}(\Gamma, E)$ represents all triplets of the input trees that have to be combined. In the case of the supergenetree problem, another tree is available, the species tree S. A triplet $xy|z$ found in the input trees $\mathcal{G}$ can be reconciled with S, and if $r(xy|z)$ is a duplication, then any tree compatible with $\mathcal{G}$ must contain this duplication. Say that $r(xy|z)$ is a required duplication mapped to $r(S)$ if $s(r(xy|z)) = r(S)$ and $r(xy|z)$ is a duplication. Let us include this information in $\mathcal{T}$. More precisely, let $\mathcal{C} = \mathcal{C}(\mathcal{T})$ denote the set of connected components of $\mathcal{T}$, and let $\mathcal{T}(\mathcal{C})$ be the graph whose vertex set is $\mathcal{C}$, and $C_1, C_2 \in \mathcal{C}$ share an edge if C_1 has vertices x, y and C_2 has a vertex z such that $xy|z$ is a triplet in $\mathcal{G}$ with $r(xy|z)$ being a required duplication mapped to $r(S)$. If there are, say, d distinct such triplets, one can possibly set a weight of d to the C_1C_2 edge. See Figure 7.4 for an example.

Consider the problem of clustering the components of $\mathcal{T}(\mathcal{C})$ into two parts B_1, B_2 of a bipartition in a way minimizing the number of duplications preceding the speciation event $r(S)$. For each $C_1 \in B_1$ and $C_2 \in B_2$ such that C_1C_2 is an edge of $\mathcal{T}(\mathcal{C})$, a tree T rooted at the bipartition (B_1, B_2) contains the required duplications mapped to $r(S)$ represented by the C_1C_2 edge. If there are k such edges between B_1 and B_2 totalizing a weight of w, the single duplication at the root of T contains those w required duplications. In other words, we have "merged" w required duplications into one. It then becomes natural to find the bipartition of $\mathcal{T}(\mathcal{C})$ that merges a maximum of duplications, i.e. that contains a set of edges crossing between the two parts of maximum weight. This is the well-known Max-Cut problem. For instance in Figure 7.4, the Max-Cut has a weight of 3 and leads to the optimal tree T_1. Any other bipartition sends a required duplication to a lower level and is hence suboptimal. The T_2 tree is obtained from first taking the suboptimal $(\{a_1, b_1, d_1\}, \{c_1, e_1, f_1\})$ bipartition, which creates a duplication at the root

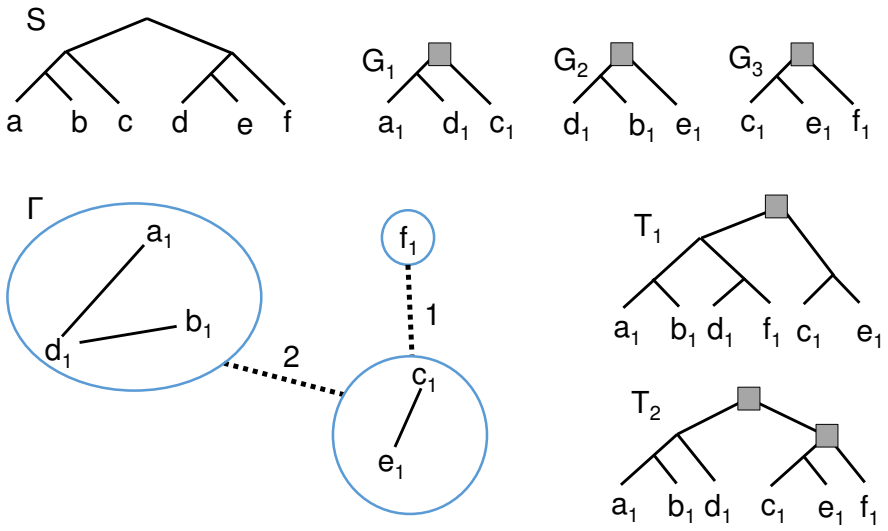

Figure 7.4: Example of how Max-Cut can be applied to the MinPreSpeDup problem. S is a species tree, $\mathcal{G} = \{G_1, G_2, G_3\}$ and $\mathcal{T}$ is the BUILD graph (solid edges). Its connected components are enclosed in circles, and the dotted edges represent required duplications mapped to $r(S)$. The edge of weight 2 is explained by the $a_1 d_1 | c_1$ and $d_1 b_1 | e_1$ triplets, whereas the edge of weight 1 is explained by the $c_1 e_1 | f_1$ triplet. A Max-Cut creates the bipartition $(\{a_1, b_1, d_1, f_1\}, \{c_1, e_1\})$, leading to the T_1 tree which merges all required duplications at its root. The tree T_2 is obtained from the suboptimal bipartition $(\{a_1, b_1, d_1\}, \{c_1, e_1, f_1\})$ and has 2 duplications.

and defers the $c_1e_1|f_1$ required duplication for later.

Note however that the components of $\mathcal{T}$ may contain required duplications themselves, which are not represented by the edges of $\mathcal{T}(\mathcal{C})$. Thus, a Max-Cut must then be applied recursively on both parts of the chosen bipartition. Therefore, this method does not benefit directly from the efficient approximation factor known for the Max-Cut problem, as the approximation error stacks with each application. In the next section, we show that, unlike Max-Cut, the MinPreSpeDup problem cannot admit a constant factor approximation (unless P = NP).

7.5 Inapproximability of the MinDupSGT and MinPreSpeDupSGT problems

Through the rest of this section, we denote by $n = |\Gamma|$ the size of the considered gene family. We show that both the MinDupSGT problem and its restriction the MinPreSpeDupSGT problem are NP-hard.

Theorem 7.1. *The* MinDupSGT *and* MinPreSpeDupSGT *problems are both NP-hard to approximate within a factor of* $n^{1-\epsilon}$ *for any constant* $0 < \epsilon < 1$. *Moreover, this result holds for both problems even when restricted to instances having at most one gene per species in* Γ.

Proof. We use a reduction from the minimum k-colorability problem. Recall that a graph $H = (V, E)$ is k-colorable if there is a partition $\{V_1, V_2, \ldots, V_k\}$ of V into independent sets (i.e. if $x, y \in V_i$ for some $1 \leq i \leq k$, then $xy \notin E$). It is now well-known [179] that the smallest k for which H is k-colorable cannot be approximated within a factor of $|V|^{1-\epsilon}$ unless $P = NP$.

Now, given a graph $H = (V, E)$, we construct a gene set Γ, a set of rooted triplet gene trees $\mathcal{G}$ and a species tree S such that H is k-colorable if and only if $\mathcal{G}$ is compatible with some gene tree T having at most $k - 1$ duplications when reconciled with S. Using the same construction, we also show that H is k-colorable if and only if $\mathcal{G}$ is compatible with some gene tree T having

at most $k-1$ pre-speciation duplications when reconciled with S. In both cases, the gene-species mapping s is bijective, proving the second part of the theorem statement.

Let $\Gamma = \{v_1, v_2 : v \in V\}$ and for each edge $vw \in E$, add the triplets $v_1v_2|w_1$, $v_1v_2|w_2$, $w_1w_2|v_1$ and $w_1w_2|v_2$ to $\mathcal{G}$. Observe that this forces any tree T that displays $\mathcal{G}$ to display the tree $((v_1, v_2), (w_1, w_2))$. Add one species to Σ for each gene of Γ so that the gene-species mapping s is bijective. As for S, first let S_1 be any binary tree with one leaf for each member of $\{s(v_1) : v \in V\}$, and in the same manner let S_2 be any binary tree with one leaf for each member of $\{s(v_2) : v \in V\}$. Obtain S by connecting the root of S_1 and the root of S_2 under a common parent $r(S)$. Thus $s(v_1)$ and $s(v_2)$ are separated by $r(S)$ for any $v \in V$. Clearly, $\mathcal{G}$ and S can be constructed in polynomial time.

Claim 1 : if H is k-colorable, then we can find a tree T compatible with $\mathcal{G}$ having at most $k-1$ duplications. Moreover each such duplication x is a pre-speciation duplication (i.e. $s(x) = r(S)$).

Let $\{V_1, V_2, \ldots, V_k\}$ be a k-coloring of H. For each $1 \leq i \leq k$, let T_i be the tree with leafset $V_i' = \{v_1, v_2 : v \in V_i\}$ that has only speciations, i.e. T_i is $S|_{s(V_i')}$ (because all genes in V_i' belong to a different species). Notice that $s(r(T_i)) = r(S)$, since $r(S)$ separates v_1 from v_2 for all $v \in V$. Obtain T by taking any binary tree on k leaves (and hence $k-1$ internal nodes), then replacing each leaf by a distinct T_i. In this manner, T has $k-1$ duplications since only the internal nodes of T that do not belong to any T_i need to be duplications. Moreover, each duplication node x has $s(x) = r(S)$. It remains to show that T is compatible with $\mathcal{G}$. It suffices to observe that all triplets of $\mathcal{G}$ are of the form $v_1v_2|w_h$ with $h \in \{1, 2\}$, and that such a triplet being in $\mathcal{G}$ implies that $vw \in E$. For such a triplet, we must then have $v \in V_i$ and $w \in V_j$ with $i \neq j$, implying $v_1, v_2 \in V_i'$ and

$w_h \in V'_j$. By the construction of T, $v_1v_2|w_h$ must be a triplet of T, as desired.

Claim 2 : if there is a tree T compatible with $\mathcal{G}$ having $k-1$ duplications, then H is k-colorable. Moreover if T has $k-1$ duplications such that each duplication x has $s(x) = r(S)$, then H is k-colorable.

Let T be a tree compatible with $\mathcal{G}$ having $k-1$ duplications. Call a node x of T *S-maximal* if x is not a duplication node mapped to $r(S)$ but every proper ancestor of x is a duplication mapped to $r(S)$. Let $X = \{x_1, x_2, \ldots, x_m\}$ be the set of S-maximal nodes of T. Note that if $y \neq r(T)$ is a duplication mapped to $r(S)$, then so is the parent of y. This implies that every leaf ℓ of T has at least one ancestor x_i in X, since x_i is the highest (i.e. closest to the root) ancestor of ℓ that is not a duplication mapped to $r(S)$ (such an x_i always exists, since ℓ is itself one such node). Moreover, x_i is unique, as no other $x_j \in X$ can be the ancestor of x_i. Therefore, $\{\mathcal{L}(T_{x_1}), \ldots, \mathcal{L}(T_{x_m})\}$ is a partition of $\mathcal{L}(T)$. We next show that $m \leq k$. Let T' be the tree obtained by removing all descendants of x_i in T, for all $1 \leq i \leq m$. Then T' is a binary tree with m leaves, and all its $m-1$ internal nodes are duplications mapped to $r(S)$. Since T has no more than $k-1$ duplications (in either cases of the claim), T' has at most $k-1$ internal nodes and therefore at most k leaves. We deduce that $m \leq k$.

Observe that if $vw \in E$, then $\alpha = lca(v_1, v_2, w_1, w_2)$ must be a duplication such that $s(\alpha) = r(S)$. Indeed, α separates $lca(v_1, v_2)$ from $lca(w_1, w_2)$ since T displays $((v_1, v_2), (w_1, w_2))$. But since $s(lca(v_1, v_2)) = s(lca(w_1, w_2)) = r(S)$ by the construction of S, $s(\alpha)$ can only be $r(S)$ as well, and so α must be a duplication.

Now, let $V_i = \{v : v_1 \text{ is a descendant of } x_i\}$ for each $1 \leq i \leq m$. Take $v, w \in V_i$ for some i. We show that $vw \notin E$, and thus that $\{V_1, \ldots, V_m\}$ forms a coloring of H with at most k colors. The argument applies whether each duplication maps to $r(S)$ or not, proving both parts of the claim. Suppose for

the sake of contradiction that $vw \in E$, but $v, w \in V_i$. In T, $lca(v_1, w_1)$ must be a descendant of x_i, since x_i is a common ancestor of v_1 and w_1 by the definition of V_i. Moreover, $lca(v_1, w_1) \neq x_i$ since $lca(v_1, w_1) = lca(v_1, v_2, w_1, w_2)$ is a duplication mapped to $r(S)$, as shown above, while x_i is not such a duplication, by its definition. Therefore, $lca(v_1, w_1)$ is a proper descendant of x_i. But $s(lca(v_1, w_1)) = r(S) = s(x_i)$ implies that x_i is a duplication mapped to $r(S)$, a contradiction. We conclude that $\{V_1, \ldots, V_m\}$ with $m \leq k$ is a proper coloring of H.

This reduction, together with the fact that the k-coloring problem is NP-hard to approximate within a $n^{1-\epsilon}$ factor, proves the Theorem. □

7.6 Independent Speciation trees

We now consider the MinDupSGT problem in the special case where the input gene trees are *independent speciation trees*, meaning: (1) each gene of Γ appears in at most one gene tree leafset, and (2) gene trees of $\mathcal{G} = \{G_1, G_2, \cdots, G_k\}$ are all speciation trees with respect to the species tree S. Our objective is to find a gene tree T compatible with $\mathcal{G}$ minimizing duplications that also maintains the orthology relationships specified by $\mathcal{G}$. In other words, we require that for every $G_i \in \mathcal{G}$, $T|_{\mathcal{L}(G_i)}$ has only speciations. We say that a gene tree T that satisfies this property *preserves the speciations* of $\mathcal{G}$. Note that if T preserves the speciations of $\mathcal{G}$, then it is necessarily compatible with $\mathcal{G}$. We call $T|_{\mathcal{L}(G_i)}$ the *copy of* G_i in T.

Minimum Speciation SuperGeneTree (MinSpecSGT problem):
Input: A species set Σ and a binary species tree S for Σ; a gene family Γ, a set $\Gamma_{i,1 \leq i \leq k}$ of disjoint subsets of Γ, and a set $\mathcal{G} = \{G_1, G_2, \cdots, G_k\}$ of consistent *independent speciation trees* such that, for each $1 \leq i \leq k$, G_i is a tree for Γ_i.
Output: Among all gene trees for Γ that preserve the speciations of $\mathcal{G}$, one tree T of minimum duplication cost.

Notice that, since no gene of Γ appears more than once in the set of input trees, $\mathcal{G}$ always admits a solution. Indeed, taking any binary tree on k leaves and replacing each leaf by a distinct G_i achieves the desired result. However, while apparently easier, we show that finding such a gene tree T minimizing the number of duplications is still hard.

Theorem 7.2. *The decision version of the MinSpecsSGT problem is NP-complete, i.e. it is NP-complete to decide if a species tree S and a set of independent speciation trees $\mathcal{G}$ admit a supertree T that preserves its speciations with at most k duplications.*

Proof. The problem is easily seen to be in NP, as it is easy to verify in polynomial time that a given gene tree T is compatible with $\mathcal{G}$, preserves its speciations and has k duplications. For NP-hardness, we turn to the decision version of the k-colorability problem. That is, for a given k, deciding if a graph $H = (V, E)$ is k-colorable is NP-hard. We create from H a species tree S and a set of independent speciation trees $\mathcal{G}$ such that H is k-colorable if and only if S and $\mathcal{G}$ admit a supertree T with at most $k - 1$ duplications.

Let $n = |V|$, and denote $V = \{v_1, \ldots, v_n\}$. To create S, start with any binary tree S' on $\binom{n}{2}$. denote this leafset $W = \{w_{i,j} : 1 \leq i < j \leq n\}$ so that there is a one-to-one correspondence between W and the unordered pairs of V. Then, add a special leaf a by joining it with the root of S' under a common parent p, and finally obtain S by adding another special leaf b by joining it with p under a common parent. Therefore, the species set is $\Sigma = \mathcal{L}(S) = W \cup \{a, b\}$.

For the construction of each gene tree $G \in \mathcal{G}$, we ease up notation by labeling each leaf g of G by $s(g)$ directly (e.g. if we say that G is of the form (a, b), we mean that G has two leaves g_a, g_b such that $s(g_a) = a$ and $s(g_b) = b$). In this manner, since all trees of $\mathcal{G}$ contain only speciations, each tree $G \in \mathcal{G}$ must be a subtree of S (or it is obtained from such a subtree by

contracting edges). Also recall that we are assuming that each gene appears in at most one gene tree of $\mathcal{G}$, and so the genes from two distinct trees must also be distinct (even if they share the same label).

In $\mathcal{G}$, we first add k trees of the form (a, b), plus one tree G_i for each vertex v_i in H. The tree G_i corresponding to $v_i \in V$ is a copy of S from which we remove every leaf except those $w_{j,k}$ for which one of $j = i$ or $k = i$ holds, and $v_j v_k \in E$ (i.e. we keep the leaves of W that correspond to an edge incident to v_i). Also contract the degree 2 nodes of G_i. Notice that if $v_i v_j \in E$ and $i < j$, then both G_i and G_j contain a gene in the $w_{i,j}$ species. Also, if $v_i v_j \notin E$, then G_i and G_j have no genes from a common species.

Claim 1 : if H is k-colorable, then S and $\mathcal{G}$ admit a supertree T having at most $k - 1$ duplications.

Let $\{V_1, \ldots, V_k\}$ be a k-partition of V into independent sets. Take any h such that $1 \leq h \leq k$. Recall that if $v_i, v_j \in V_h$, then G_i and G_j share no gene from a common species (since $v_i v_j \notin E$). Thus the trees in $\mathcal{G}_h = \{G_i : v_i \in V_h\}$ are all disjoint in terms of species. Let Σ_h be the set of species that appear in some tree of $\mathcal{G}_h$. Then, the tree $S|_{\Sigma_h}$ contains a copy of each tree in $\mathcal{G}_h$, and none of these copies overlap. Obtain T_h by joining a gene labeled a to $r(S|_{\Sigma_h})$ under a common parent p, then joining a gene labeled b to p under a new common parent. Now, T_h contains a copy of each tree in $\mathcal{G}_h$ and a copy of one of the (a, b) trees. By taking a tree with k leaves (where at worst, each $k - 1$ internal node is a duplication), and replacing each leaf by the speciation trees $T_1, \ldots, T_k$, we obtain a gene supertree T, which preserves the speciations of $\mathcal{G}$ and has at most $k - 1$ duplications.

Claim 2 : if S and $\mathcal{G}$ admit a supertree T having $k - 1$ duplications, then H is k-colorable.

We first show that if T has $k-1$ duplications, then it must have exactly k speciations mapped to $r(S)$. It cannot have more, as there would then be more than $k-1$ duplications. Suppose instead that there are $k' < k$ such speciations, and denote them $x_1, \ldots, x_{k'}$. Note that there must be at least $k'-1$ duplications in the ancestors of the x_is. Now, for $1 \leq i \leq k'$, T_{x_i} must contain a certain number of copies of a and b. Let $m_i(a)$ and $m_i(b)$ denote, respectively, the number of copies of a and b contained in T_{x_i}, noting that in total, there are k copies of each since there are k subtrees of the form (a, b) in $\mathcal{G}$. Since x_i is a speciation mapped to $r(S)$, it separates the a copies from the b copies, thus the T_{x_i} subtree must contain at least $m_i(a) - 1 + m_i(b) - 1$ duplications. Denote by $d(T)$ the number of duplications in T. It follows that $d(T) \geq k' - 1 + \sum_{i=1}^{k'}(m_i(a) + m_i(b) - 2) = k' - 1 - 2k' + \sum_{i=1}^{k'} m_i(a) + \sum_{i=1}^{k'} m_i(b) = -k' - 1 + k + k = 2k - k' - 1 > k - 1$ when $k' < k$, a contradiction.

Now, we can let $x_1, \ldots, x_k$ be the k speciation nodes of T mapped to $r(S)$. The $k-1$ duplications of T must then all be ancestors of the x_i, and they are all mapped to $r(S)$. Therefore the $T_{x_1}, \ldots, T_{x_k}$ subtrees each contain only speciations. For any $G_i \in \mathcal{G}$ corresponding to v_i, one of the T_{x_h} must contain the copy of G_i (for otherwise, the root of the copy of G_i in T would be a duplication, while it should be a speciation). Take any h such that $1 \leq h \leq k$. We claim that $V_h = \{v_i : T_{x_h} \text{ contains the copy of } G_i\}$ forms an inpedendent set. Since T_{x_h} contains only speciations, it cannot contain genes from the same species. Thus for any G_i, G_j contained in T_{x_h}, we must have $v_i v_j \notin E$, as otherwise G_i and G_j would share a gene from the same species. Therefore V_h is an independent set. Thus $\{V_1, \ldots, V_k\}$ form a k-coloring of H, and the proof is completed. □

It is interesting to note that this does not show the NP-hardness of the special case in which the input trees are only triplets. Indeed, a tree G_i created in this reduction has as many leaves as the number of neighbors of its corresponding vertex v_i. Therefore, if H is a cubic graph (ie. 3-regular), one

can generate an input with only triplets. However, deciding if a cubic graph is k-colorable can be done in linear time, and thus the triplets case cannot be shown NP-hard through this reduction. The 3-colorability problem is NP-hard on 4-regular graph though, showing the NP-hardness of the problem on input trees having at most 4 leaves.

7.7 Conclusion

We introduce the supergenetree problem which aims at constructing a supertree that displays a set of input gene trees while minimizing the reconciliation cost with respect to an input species tree. This problem is a natural formulation of the question of combining a set of gene trees obtained for subsets of a gene family into a full gene tree for the whole gene family.

The supergenetree problem is an extension of the classical supertree problem on a set of input leaf-labeled trees, where the input trees are gene trees and a species tree is used in order to evaluate the reconciliation/duplication cost of a supergenetree. We show how existing exact and greedy heuristic algorithms for the supertree problem can be used to devise approaches for solving the supergenetree problem. The resulting approaches have exponential worst-time complexity as the original supertree algorithms.

We show that the supergenetree problem for the duplication cost is NP-hard to approximate within a factor essentially better than n, and this complexity remains the same even when the problem is restricted, in a greedy approach, to finding a supertree with a minimum number of duplications before each speciation of the species tree. We also consider a restriction of the supergenetree problem relevant to many biological applications where subsets of orthologs are studied separately and then amalgamated into a single tree. Even this restriction is shown to be NP-complete. The reconciliation cost remains to be studied, although we conjecture all of the above mentioned problems are hard in this case also.

These negative complexity results are not surprising though as they ex-

tend an already large set of problems related to supertrees that are known to be NP-hard. We think that appropriate heuristics for various classes of input trees are worth to be considered in future projects. Removing the assumption that the input gene trees are compatible would also lead to new interesting problems. A promising avenue would be to consider constructive FPT algorithms that can be integrated in greedy heuristics or dynamic programming algorithms. Also other restrictions on the input gene trees can be explored, hopefully leading to polynomial problems. Constructing gene trees by amalgamating smaller trees for subsets of orthologous genes is a natural way of constructing large trees that would benefit from a thorough theoretical and algorithmic analysis.

Competing interests

The authors declare they have no competing interests.

Author's contributions

ML, AO, NE devised the proofs and algorithms and wrote the paper.

Declarations

This work is funded by the Natural Sciences and Engineering Research Council of Canada (NSERC), Fonds de Recherche Nature et Technologies of Quebec (FRQNT) and Université de Sherbrooke.

7.7.1 Contributions

Manuel Lafond a rédigé les preuves de d'inapproximabilité et de NP-complétude aux problèmes étudiés. Manuel Lafond, Aïda Ouangraoua et Nadia El-Mabrouk ont participé à l'élaboration des algorithmes et des heuristiques. Tous les auteurs ont participé à la rédaction de l'article.

CHAPITRE 8

CONCLUSION

Dans cette thèse, nous avons étudié plusieurs avenues algorithmiques permettant d'améliorer un arbre de gènes dans le modèle DLS. Plus précisément, nous avons décrit comment y détecter et corriger des erreurs potentielles, notamment en transformant les branches faiblement supportées de l'arbre en polytomie, puis en réarrangeant cette polytomie de façon à minimiser le nombre de duplications et pertes. Nous avons aussi vu que si l'on avait en main une méthode parfaite d'inférence d'orthologie, celle-ci pouvait servir à détecter et corriger des problèmes de fausses paralogies au niveau d'un arbre de gènes. Bien évidemment, il n'existe pas de méthode parfaite, et on se doit de vérifier que les relations d'orthologie et paralogie données par une méthode d'inférence sont fiables. Nous avons développé, dans cette thèse, les outils algorithmiques nécessaires permettant d'effectuer l'une des vérifications les plus fondamentales par rapport à de telles relations: il doit bel et bien exister une histoire évolutive dépeignant ces relations. Dans cette optique, la réconciliation s'avère être un critère intéressant puisqu'elle mène à des problèmes algorithmes pouvant être résolus en temps polynomial, du moins dans le modèle DLS. Pour ce qui est de combiner plusieurs arbres en un seul superarbre dans ce même modèle, nous avons vu que malheureusement, la réconciliation est d'une utilité limitée puisqu'elle mène à des formulations de problèmes NP-complets, et même difficile à approximer.

Les méthodes présentées dans cette thèse peuvent toutes être incorporées au sein d'une même étude phylogénétique. La Figure 8.1 illustre une proposition de "pipeline" de construction et correction d'arbres de gènes. Ce processus peut s'adapter à n'importe quelle méthode de reconstruction phylogénique, qu'elle produise en sortie un arbre binaire ou non-binaire, ou même plusieurs arbres. Il est à noter qu'il reste encore à développer un algorithme

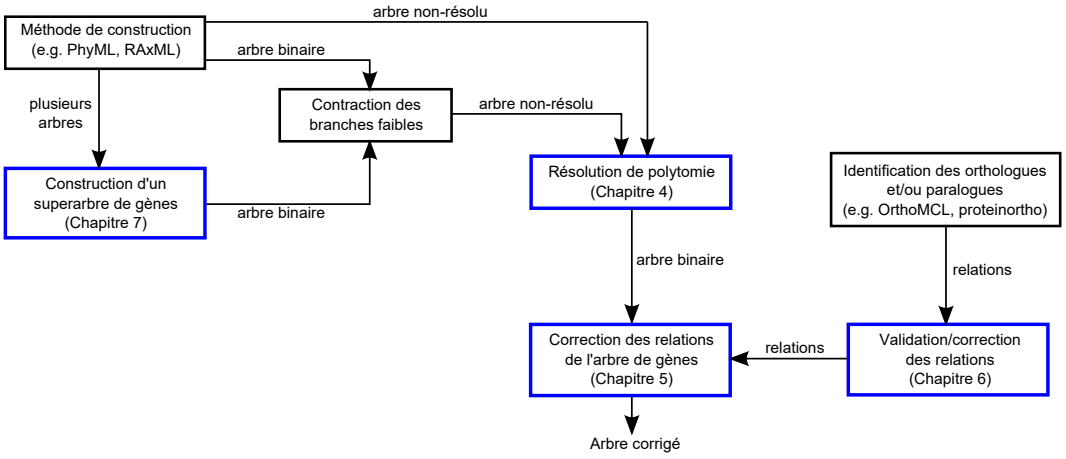

Figure 8.1: Pipeline de correction d'arbre incorporant les méthodes présentées dans cette thèse. Les méthodes de la thèse contribuant à ce pipeline sont encadrées en bleu. Le point de départ est la construction d'un arbre initial (ou de plusieurs arbres initiaux) à partir de n'importe quelle méthode.

applicable en pratique pour le problème de superarbre. Malgré le fait que le problème soit NP-complet et même difficiles à approximer, il est possible que certaines instances soient traitables, par exemple s'il n'y a pas trop d'arbres. Aussi, alors qu'il est possible de valider des relations d'orthologie et de corriger un arbre par rapport à celles-ci, on ne sait pas traiter de façon efficace des relations d'orthologie *et* paralogie. Il reste encore bien du travail à accomplir dans cette direction.

Toutes les méthodes étudiées dans cette thèse supposent que l'arbre d'espèces S est connu et exact. C'est d'ailleurs ce que font presque toutes les méthodes de correction d'arbres de gènes développées à ce jour. Les arbres d'espèces sont toutefois sensibles à la plupart des sources d'erreurs dont nous avons fait état pour les arbres de gènes. Il n'existe pourtant pas de méthode qui tienne compte de cette imprécision. Une première idée serait de permettre un degré d'incertitude dans l'arbre d'espèces, par exemple en

contractant les branches ayant un faible support. Il est possible de définir la réconciliation avec un arbre d'espèces non-binaire [153] et ainsi formuler un problème d'optimisation de résolution de polytomies analogue à celui du chapitre 4. À la connaissance de l'auteur, les seuls travaux faits à ce sujet sont dans [175], où l'on propose de résoudre à la fois l'arbre de gènes G et l'arbre d'espèces S. Trouver les meilleures résolutions est NP-complet, et les auteurs décrivent une heuristique qui résout d'abord S, puis ensuite G séparément. Des méthodes de correction d'arbres de gènes *sans* arbre d'espèces seraient aussi envisageables. Par exemple, le problème de consistance du chapitre 6 requiert seulement qu'il *existe* un arbre d'espèces consistant avec un arbre corrigé - aucune supposition n'est initialement faite sur la topologie exacte de l'arbre d'espèces. La question d'identifier des propriétés propres aux gènes qui pourraient servir à le corriger tout en étant libre d'un arbre d'espèces demeure toutefois ouverte.

Notons par ailleurs que toutes les méthodes décrites dans cette thèse sont basées sur le modèle de duplications, spéciations et pertes. On suppose donc que toutes les histoires des familles de gènes peuvent s'expliquer exclusivement par ces trois types d'événements. Les méthodes de construction d'arbres énumérées dans le chapitre 3 sont d'ailleurs basées sur la même hypothèse. Par contre, il serait intéressant de lever, ou du moins de relaxer, ces hypothèses, puisqu'on sait que d'autres événements tels que la coalescence profonde, l'hybridation ou les transferts latéraux de contenu génétique participent à l'évolution. Les erreurs au sein des arbres de gènes sont peut-être alors dues au fait que ces événements ne sont pas considérés lors de la construction. Ceci ouvre la porte à une multitude de questions. Au sens large, peut-on étendre les méthodes décrites ici à ces nouveaux types d'événements? Peut-on détecter si une erreur est due à une mauvaise construction ou à un événement évolutif non-considéré? Comment résoudre une polytomie en présence de transfert horizontaux (un début de réponse est dans [117])? Par ailleurs, la présence de transferts ou d'hybridation

donne lieu à un nouveau type de relation entre gènes, soit la *xénologie* (deux gènes sont xénologues s'il y a présence de transfert/hybridation sur le chemin menant à leur ancêtre commun). Est-ce que les questions de satisfaisabilité et consistance s'appliquent aussi lorsque certains gènes sont xénologues? À l'instar du graphe de relations défini dans le chapitre 6, peut-on parler d'un graphe d'orthologie/paralogie/xénologie? Si oui, y a-t-il des propriétés qui doivent être satisfaites par ce graphe?

Mais même en se limitant aux relations d'orthologie et paralogie, cette thèse laisse des questions sans réponses. Que faire avec un graphe de relations insatisfaisable ou inconsistant? Ce dernier nécessite bien sûr d'être corrigé. Or, nous avons démontré dans [97] que trouver un minimum d'arêtes à modifier est NP-complet. Une formulation en programmation linéaire du problème a été publiée [81], mais on ne sait pas s'il existe un algorithme d'approximation borné à ce problème, ni même s'il y a des heuristiques susceptibles d'être efficaces en pratique.

Dans un autre ordre d'idées, cette thèse présente une multitude de résultats à saveur plutôt théorique. La question de l'utilisabilité pratique de ces résultats demeure ouverte, et la suite logique des choses serait d'effectuer une étude empirique sur des arbres de gènes réels. À ce jour, certains travaux par rapport à la résolution de polytomies ont déjà été entrepris. L'algorithme présenté au chapitre 4 a été implémenté et nommé ProfileNJ[1]. La méthode permet aussi de raffiner le choix de solution: parmi toutes les résolutions qui minimisent le coût de réconciliation, on choisit celle qui optimise le critère Neighbor-Joining (d'où le nom du programme). Des résultats préliminaires sur des données simulées montrent que la méthode de reconstruire l'arbre, contracter les branches faibles et appliquer ProfileNJ, est plus rapide que d'autres méthodes de correction telles que NOTUNG et TreeFix, et retrouve le vrai arbre environ aussi souvent. Le logiciel a aussi été utilisé sur les arbres d'Ensembl afin de les corriger. On observe que malgré une légère baisse

[1]Disponible à l'adresse `https://github.com/UdeM-LBIT`

au niveau de la vraisemblance des arbres corrigés, ceux-ci peuvent servir à inférer le contenu en gènes de génomes ancestraux avec plus de précision.

Quant à la correction par orthologie et paralogie, nous avons proposé à la fin du chapitre 6 une façon d'inférer des relations: exécuter un logiciel d'inférence (e.g. OrthoMCL [106], proteinortho [105]) plusieurs fois avec une multitude de paramètres, et conserver les relations communes à toutes les inférences. On ne sait pas vraiment à quel point les relations partielles inférées par cette technique sont précises - des tests plus approfondis sont nécessaires afin de valider cette méthode. De plus, les relations pourraient être accompagnées par un poids (e.g. une probabilité) plutôt que d'être orthologie/paralogie/inconnu. Ce poids pourrait être basé sur la fréquence à laquelle la relation apparaît dans les exécutions du logiciel d'inférence. Ceci soulève plusieurs questions algorithmiques, notamment l'extension des algorithmes connus aux relations avec poids. Pour ce qui est des superarbres, bien que les résultats du chapitre 7 soient quelques peu décourageants, l'applicabilité pratique de la réconciliation à la création de superarbres n'est pas exclue. Les algorithmes exacts exponentiels sont analysés selon leur pire cas, mais on ne sait pas quelles en sont les performances sur de vrais arbres. De plus, on croit que le problème est résoluble à paramètre fixé (*fixed parameter tractable* en anglais) sur le nombre d'arbres. C'est-à-dire, s'il n'y a pas trop d'arbres, le problème de trouver un superarbre minimisant le coût de réconciliation peut se résoudre rapidement. Un algorithme accomplissant cette tâche nous permettrait alors de tester l'impact de la réconciliation sur de vrais arbres.

RÉFÉRENCES

[1] A. Aho, S. Yehoshua, T. Szymanski, and J. Ullman. Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions. *SIAM J. Comput.*, 10(3):405- 421, 1981.

[2] O. Akerborg, B. Sennblad, L. Arvestad, and J. Lagergren. Simultaneous bayesian gene tree reconstruction and reconciliation analysis. *Proceedings of the National Academy of Sciences of the United States of America*, 106(14):5714–5719, 2009. doi:10.1073/pnas.0806251106.

[3] A. M. Altenhoff, R. A. Studer, M. Robinson-Rechavi, and C. Dessimoz. Resolving the ortholog conjecture: orthologs tend to be weakly, but significantly, more similar in function than paralogs. *PLoS Comput Biol*, 8(5):e1002514, 2012.

[4] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, 1990.

[5] L. Arvestad, A. Berglund, J. Lagergren, and B. Sennblad. Gene tree reconstruction and orthology analysis based on an integrated model for duplications and sequence evolution. In *RECOMB*, pages 326-335. 2004.

[6] K. Atteson. The performance of neighbor-joining algorithms of phylogeny reconstruction. In *Computing and Combinatorics*, pages 101–110. Springer, 1997.

[7] M. Bansal, J. Burleigh, O. Eulenstein, and D. Fernández-Baca. Robinson-foulds supertrees. *Alg. Mol. Biol.*, 5(18), 2010.

[8] B. R. Baum. Combining trees as a way of combining data sets for

phylogenetic inference, and the desirability of combining gene trees. *Taxon*, pages 3–10, 1992.

[9] S. Bérard, C. Gallien, B. Boussau, G. J. Szollosi, V. Daubin, and E. Tannier. Evolution of gene neighborhoods within reconciled phylogenies. *Bioinformatics*, 28(18):i382–i388, 2012.

[10] A. Berglund, E. Sjolund, G. Ostlund, and E. Sonnhammer. InParanoid 6: eukaryotic ortholog clusters with inparalogs. *Nucleic Acids Research*, 36:D263 - D266, 2008.

[11] A. Berglund-Sonnhammer, P. Steffansson, M. Betts, and D. Liberles. Optimal gene trees from sequences and species trees using a soft interpretation of parsimony. *J. Mol. Evol.*, 63:240-250, 2006.

[12] O. Bininda-Emonds, editor. *Phylogenetic Supertrees combining information to reveal The Tree Of Life*. Computational Biology. Kluwer Academic, Dordrecht, the Netherlands, 2004.

[13] O. Bininda-Emonds, J. Gittleman, and M. Steel. The super tree of life: Procedures, problems and prospects. *Annu. Rev. Ecol. Syst.*, 33:265-289, 2002.

[14] O. R. Bininda-Emonds, M. Cardillo, K. E. Jones, R. D. MacPhee, R. M. Beck, R. Grenyer, S. A. Price, R. A. Vos, J. L. Gittleman, and A. Purvis. The delayed rise of present-day mammals. *Nature*, 446(7135):507–512, 2007.

[15] B. Boussau, G. J. Szollosi, L. Duret, M. Gouy, E. Tannier, and V. Daubin. Genome-scale coestimation of species and gene trees. *Genome research*, 23(2):323–330, 2013.

[16] B. Boussau, G. J. Szollosi, L. Duret, M. Gouy, E. Tannier, and V. Daubin. Genome-scale coestimation of species and gene trees. *Genome Research*, 23:323-330, 2013.

[17] S. Bérard, C. Gallien, B. Boussau, G. J. Szollosi, V. Daubin, and E. Tannier. Evolution of gene neighborhoods within reconciled phylogenies. *Bioinformatics*, 28(18):i382–i388, 2012. doi:10.1093/bioinformatics/bts374.

[18] D. Bryant. A classification of consensus methods for phylogenetics. *DIMACS series in Discrete Math. and Theo. Comput. Sci.*, 2003.

[19] P. Buneman. The recovery of trees from measures of dissimilarity. *Mathematics the the Archeological and Historical Sciences*, page 387–395, 1971. Edinburgh University Press.

[20] J. Byrka, S. Guillemot, and J. Jansson. New results on optimizing rooted triplets consistency. *Discrete Applied Mathematics*, 158(11):1136–1147, 2010.

[21] P. J. Campbell, E. D. Pleasance, P. J. Stephens, E. Dicks, R. Rance, I. Goodhead, G. A. Follows, A. R. Green, P. A. Futreal, and M. R. Stratton. Subclonal phylogenetic structures in cancer revealed by ultra-deep sequencing. *Proceedings of the National Academy of Sciences*, 105(35):13081–13086, 2008.

[22] R. Cann, M. Stoneking, and A. Wilson. Mitochondrial dna and human evolution. *Nature*, 325:31–36, 1987. doi:doi:10.1038/325031a0.

[23] W. Chang and O. Eulenstein. Reconciling gene trees with apparent polytomies. In D. Chen and D. T. Lee, editors, *Proceedings of the 12th Conference on Computing and Combinatorics (COCOON)*, volume 4112 of *Lecture Notes in Computer Science*, pages 235–244. 2006.

[24] W. Chang and O. Eulenstein. Reconciling gene trees with apparent polytomies, technical report. In *Department of Computer Science, Iowa State University*. 2006.

[25] R. Chaudhary, J. Burleigh, and O. Eulenstein. Efficient error correction algorithms for gene tree reconciliation based on duplication, duplication and loss, and deep coalescence. *BMC-Bioinformatics*, 13(Supp.10):S11, 2011.

[26] C. Chauve and N. El-Mabrouk. New perspectives on gene family evolution: losses in reconciliation and a link with supertrees. In *RECOMB 2009*, volume 5541 of *LNCS*, pages 46-58. Springer, 2009.

[27] C. Chauve, N. El-Mabrouk, L. Gueguen, M. Semeria, and E. Tannier. *Models and algorithms for genome evolution*, chapter Duplication, rearrangement and reconciliation: a follow-up 13 years later. Springer, 2013.

[28] K. Chen, D. Durand, and M. Farach-Colton. Notung: Dating gene duplications using gene family trees. *J. Comp. Biol.*, 7:429–447, 2000.

[29] X. Chen and J. Zhang. The ortholog conjecture is untestable by the current gene ontology but is supported by rna sequencing data. *PLoS Comput Biol*, 8(11):e1002784, 2012.

[30] F. Chiaromonte, W. Miller, and E. E. Bouhassira. Gene length and proximity to neighbors affect genome-wide expression levels. *Genome research*, 13(12):2602–2608, 2003.

[31] B. Chor and T. Tuller. Finding a maximum likelihood tree is hard. *Journal of the ACM (JACM)*, 53(5):722–744, 2006.

[32] R. Cole, M. Farach-Colton, R. Hariharan, T. Przytycka, and M. Thorup. An o (n log n) algorithm for the maximum agreement subtree problem for binary trees. *SIAM Journal on Computing*, 30(5):1385–1404, 2000.

[33] T. G. O. Consortium. Gene ontology: tool for the unification of biology. *Nat. Genet.*, 25(1):25 - 29, 2000.

[34] M. Constantinescu and D. Sankoff. An efficient algorithm for supertrees. *J. Classif.*, 12:101- 112, 1995.

[35] J. Cotton and M. Wilkinson. Majority-rule supertrees. *Syst. Biol.*, 56(3):445- 452, 2007.

[36] M. Csurös. Ancestral reconstruction by asymmetric wagner parsimony over continuous characteand squared parsimony over distributions. *Sixth RECOMB Satellite Workshop on Comparative Genomics*, pages 72–86, 2008.

[37] P. H. da Silva, S. Dantas, C. Zheng, and D. Sankoff. Graph-theoretic modelling of the domain chaining problem. In *Algorithms in Bioinformatics*, pages 296–307. Springer, 2015.

[38] R. Datta, C. Meacham, B. Samad, C. Neyer, and K. Sjolander. Berkeley phog: Phylofacts orthology group prediction web server. *Nucleic Acids Res*, 37:W84-W89, 2009.

[39] T. J. Davies, T. G. Barraclough, M. W. Chase, P. S. Soltis, D. E. Soltis, and V. Savolainen. Darwin's abominable mystery: insights from a supertree of the angiosperms. *Proceedings of the National Academy of Sciences of the United States of America*, 101(7):1904–1909, 2004.

[40] R. Dawkins. *The selfish gene*. 199. Oxford university press, 1976.

[41] W. H. Day. Optimal algorithms for comparing trees with labeled leaves. *Journal of classification*, 2(1):7–28, 1985.

[42] R. Dondi and N. El-Mabrouk. Minimum leaf removal for reconciliation: Complexity and algorithms. In *CPM*, volume 7354 of *Lecture Notes in Computer Science*, pages 399-412. Springer, 2012.

[43] A. Doroftei and N. El-Mabrouk. Removing noise from gene trees. In *WABI 2011, Algorithms in Bioinformatics*, volume 6833 of *LNCS/LNBI*, pages 76–91. 2011.

[44] J. Doyon, C. Scornavacca, and G. S. et al. An efficient algorithm for gene/species trees parsimonious reconciliation with losses, duplications and transfers. *Proc 14th Int Conf Res Comput Mol Biol (RECOMB-CG) 2011*, pages 93–108, 2011.

[45] J.-P. Doyon, S. Hamel, and C. Chauve. An efficient method for exploring the space of gene tree/species tree reconciliations in a probabilistic framework. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, 9(1):26–39, 2012.

[46] A. J. Drummond and A. Rambaut. Beast: Bayesian evolutionary analysis by sampling trees. *BMC evolutionary biology*, 7(1):214, 2007.

[47] D. Durand, B. Haldórsson, and B. Vernot. A hybrid micro-macro-evolutionary approach to gene tree reconstruction. *J. Comput. Biol.*, 13:320–335, 2006.

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

[49] J. Eisen. Phylogenomics: Improving functional predictions for uncharacterized genes by evolutionary analysis. *Genome Research*, pages 163–167, 1998. doi:doi:10.1101/gr.8.3.163.

[50] A. J. Enright, S. Van Dongen, and C. A. Ouzounis. An efficient algorithm for large-scale detection of protein families. *Nucleic acids research*, 30(7):1575–1584, 2002.

[51] D. S. *et al.* PAUP: phylogenetic analysis using parsimony. 4th ed., Sinauer Associates, 2002.

[52] G. Fang, N. Bhardwaj, R. Robilotto, and M. B. Gerstein. Getting started in gene orthology and functional analysis. *PLoS Comput Biol*, 6(3):e1000703, 03 2010. doi:10.1371/journal.pcbi.1000703.

[53] M. Farach, T. M. Przytycka, and M. Thorup. On the agreement of many trees. *Information Processing Letters*, 55(6):297–301, 1995.

[54] J. Felsenstein. Cases in which parsimony or compatibility methods will be positively misleading. *Systematic Biology*, 27(4):401–410, 1978.

[55] J. Felsenstein. Evolurionary trees from DNA sequences: a maximum likelihood approach. *J. Mol. Evol.*, 17:368-376, 1981.

[56] J. Felsenstein. Confidence limits on phylogenies: an approach using the bootstrap. *Evolution*, pages 783–791, 1985.

[57] J. Felsenstein and J. Felenstein. *Inferring phylogenies*, volume 2. Sinauer Associates Sunderland, 2004.

[58] R. D. Finn, J. Clements, and S. R. Eddy. Hmmer web server: interactive sequence similarity searching. *Nucleic acids research*, page gkr367, 2011.

[59] W. M. Fitch. Toward defining the course of evolution: minimum change for a specific tree topology. *Systematic Biology*, 20(4):406–416, 1971.

[60] W. M. Fitch. Homology. a personal view on some of the problems. *TIG*, 16(5):227- 231, 2000.

[61] P. Flicek, M. Amode, D. Barrell, K. Beal, S. Brent, D. Carvalho-Silva, P. Clapham, G. Coates, S. Fairley, S. Fitzgerald, L. Gil, L. Gordon, M. Hendrix, T. Hourlier, N. Johnson, A. Kahari, D. Keefe, S. Keenan, R. Kinsella, M. Komorowska, G. Koscielny, E. Kulesha, P. Larsson, I. Longden, W. McLaren, M. Muffato, B. Overduin, M. Pignatelli,

B. Pritchard, H. Riat, G. Ritchie, M. Ruffier, M. Schuster, D. Sobral, Y. Tang, K. Taylor, S. Trevanion, J. Vandrovcova, S. White, M. Wilson, S. Wilder, B. Aken, E. Birney, F. Cunningham, I. Dunham, R. Durbin, X. Fernández-Suarez, J. Harrow, J. Herrero, T. Hubbard, A. Parker, G. Proctor, G. Spudich, J. Vogel, A. Yates, A. Zadissa, and S. Searle. Ensembl 2012. *Nucleic Acids Res.*, 40:D84- D90, 2012.

[62] T. Gabaldón and E. V. Koonin. Functional and evolutionary implications of gene orthology. *Nature Reviews Genetics*, 14(5):360–366, 2013.

[63] M. Gerlinger, A. J. Rowan, S. Horswell, J. Larkin, D. Endesfelder, E. Gronroos, P. Martinez, N. Matthews, A. Stewart, P. Tarpey, et al. Intratumor heterogeneity and branched evolution revealed by multiregion sequencing. *New England Journal of Medicine*, 366(10):883–892, 2012.

[64] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995.

[65] P. A. Goloboff and D. Pol. Semi-strict supertrees. *Cladistics*, 18(5):514–525, 2002.

[66] M. Golumbic, H. Kaplan, and R. Shamir. Graph sandwich problems. *J. Algorithms*, 19(3):449–473, November 1995. ISSN 0196-6774. doi: 10.1006/jagm.1995.1047.

[67] M. Goodman, J. Czelusniak, G. Moore, A. Romero-Herrera, and G. Matsuda. Fitting the gene lineage into its species lineage, a parsimony strategy illustrated by cladograms constructed from globin sequences. *Systematic Zoology*, 28:132–163, 1979.

[68] P. Gorecki and O. Eulenstein. A linear-time algorithm for error-corrected reconciliation of unrooted gene trees. In *ISBRA*, volume 6674 of *LNBI*, pages 148-159. Springer-Verlag, 2011.

[69] P. Gorecki and J. Tiuryn. DLS-trees: a model of evolutionary scenarios. *Theoretical Computer Science*, 359:378–399, 2006.

[70] P. Gòrecki and O. Eulenstein. Algorithms: simultaneous error-correction and rooting for gene tree reconciliation and the gene duplication problem. *BMC Bioinformatics*, 13(Suppl 10):S14, 2012. doi: 10.1186/1471-2105-13-S10-S14.

[71] S. Guidon and O. Gascuel. A simple, fast and accurate algorithm to estimate large phylogenies by maximum likelihood. *Systematic Biology*, 52:696- 704, 2003.

[72] M. Hahn. Bias in phylogenetic tree reconciliation methods: implications for vertebrate genome evolution. *Genome Biol.*, 8(R141), 2007.

[73] M. Hallett, J. Lagergren, and A. Tofigh. Simultaneous identification of duplications and lateral transfers. In *Proceedings of the eighth annual international conference on Resaerch in computational molecular biology*, pages 347–356. ACM, 2004.

[74] L. Hartwell, L. Hood, M. Goldberg, A. Reynolds, L. Silver, and R. Veres. Genetics: from genes to genomes. 2006.

[75] M. Hasegawa, H. Kishino, and T.-a. Yano. Dating of the human-ape splitting by a molecular clock of mitochondrial dna. *Journal of molecular evolution*, 22(2):160–174, 1985.

[76] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[77] J. Hein. Reconstructing evolution of sequences subject to recombination using parsimony. *Mathematical biosciences*, 98(2):185–200, 1990.

[78] J. Hein. A heuristic method to reconstruct the history of sequences subject to recombination. *Journal of Molecular Evolution*, 36(4):396–405, 1993.

[79] J. Hein, T. Jiang, L. Wang, and K. Zhang. On the complexity of comparing evolutionary trees. *Discrete Applied Mathematics*, 71(1):153–169, 1996.

[80] M. Hellmuth, M. Hernandez-Rosales, K. Huber, V. Moulton, P. Stadler, and N. Wieseke. Orthology relations, symbolic ultrametrics, and cographs. *J. Math. Biol.*, 66(1–2):399–420, 2013.

[81] M. Hellmuth, N. Wieseke, M. Lechner, H. Lenhof, and M. Middledorf. Phylogenetics from paralogs, 2014. Unpublished manuscript.

[82] M. Hernandez-Rosales, M. Hellmuth, N. Wieseke, K. Huber, V. Moulton, and P. Stadler. From event-labeled gene trees to species trees. *BMC Bioinformatics*, 13 (Suppl. 19):56, 2012.

[83] H.Li, A. Coghlan, J. R. L. Coin, J. Heriche, L. Osmotherly, R. Li, T. L. T, Z. Zhang, L. Bolund, G. Wong, W. Zheng, P. Dehal, J. W. J, and R. Durbin. TreeFam: a curated database of phylogenetic trees of animal gene families. *Nucleic Acids Research*, 34:D572- 580, 2006.

[84] J. P. Huelsenbeck, B. Larget, R. E. Miller, and F. Ronquist. Potential applications and pitfalls of bayesian inference of phylogeny. *Systematic biology*, 51(5):673–688, 2002.

[85] J. Huerta-Cepas, S. Capella-Gutierrez, L. Pryszcz, I. Denisov, D. Kormes, M. Marcet-Houben, and T. Gabald'on. Phylomedb v3.0: an expanding repository of genome-wide collections of trees, alignments and

phylogeny-based orthology and paralogy predictions. *Nucleic Acids Res*, 39:D556-D560, 2011.

[86] D. H. Huson, R. Rupp, and C. Scornavacca. *Phylogenetic networks: concepts, algorithms and applications*. Cambridge University Press, 2010.

[87] J. Jansson, R. Lemence, and A. Lingas. The complexity of inferring a minimally resolved phylogenetic supertree. *SIAM J. on computing*, 41(1):272- 291, 2012.

[88] L. J. Jensen, P. Julien, M. Kuhn, C. von Mering, J. Muller, T. Doerks, and P. Bork. eggnog: automated construction and annotation of orthologous groups of genes. *Nucleic acids research*, 36(suppl 1):D250–D254, 2008.

[89] B. D. X. H. T. Jiang, M. Li, J. Tromp, and L. Zhang. On computing the nearest neighbor interchange distance. In *Discrete Mathematical Problems with Medical Applications: DIMACS Workshop Discrete Mathematical Problems with Medical Applications, December 8-10, 1999, DIMACS Center*, volume 55, page 125. American Mathematical Soc., 2000.

[90] T. H. Jukes and C. R. Cantor. Evolution of protein molecules. *Mammalian protein metabolism*, 3(21):132, 1969.

[91] J. Jun, I. I. Mandoiu, and C. E. Nelson. Identification of mammalian orthologs using local synteny. *BMC Genomics*, 10:630, 2009. doi: 10.1186/1471-2164-10-630.

[92] R. M. Karp. *Reducibility among combinatorial problems*. Springer, 1972.

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

[94] E. V. Koonin. Orthologs, paralogs, and evolutionary genomics 1. *Annu. Rev. Genet.*, 39:309–338, 2005.

[95] D. M. Kristensen, Y. I. Wolf, A. R. Mushegian, and E. V. Koonin. Computational methods for gene orthology inference. *Briefings in bioinformatics*, 12(5):379–391, 2011.

[96] M. Lafond, C. Chauve, R. Dondi, and N. El-Mabrouk. Polytomy refinement for the correction of dubious duplications in gene trees. *Bioinformatics*, 30(17):i519–i526, 2014.

[97] M. Lafond and N. El-Mabrouk. Orthology relation and gene tree correction: Complexity results. In *Algorithms in Bioinformatics*, pages 66–79. Springer, 2015.

[98] M. Lafond, E. Noutahi, and N. El-Mabrouk. Efficient non-binary gene tree resolution with weighted reconciliation cost. In *CPM 2016*. 2016. à paraître.

[99] M. Lafond, A. Ouangraoua, and N. El-Mabrouk. Reconstructing a supergenetree minimizing reconciliation. *BMC bioinformatics*, 16(Suppl 14):S4, 2015.

[100] M. Lafond, M. Semeria, K. Swenson, E. Tannier, and N. El-Mabrouk. Gene tree correction guided by orthology. *BMC Bioinformatics*, 14 (supp 15)(S5), 2013.

[101] M. Lafond, K. Swenson, and N. El-Mabrouk. An optimal reconciliation algorithm for gene trees with polytomies. In *Algorithms in Bioinformatics, proceedings of WABI'12*, volume 7534 of *LNCS/LNBI*, pages 106–122. 2012.

[102] M. Lafond, K. Swenson, and N. El-Mabrouk. *Models and algorithms for genome evolution*, chapter Error detection and correction of gene trees. Springer, 2013.

[103] M. Lafond, K. M. Swenson, and N. El-Mabrouk. Error detection and correction of gene trees. In *Models and Algorithms for Genome Evolution*, pages 261–285. Springer, 2013.

[104] M. Larkin and al. Clustalw and clustalx version 2. *Bioinformatics*, 23:2947- 2948, 2007.

[105] M. Lechner, S. Findeib, L. Steiner, M. Marz1, P. Stadler, and S. Prohaska. Proteinortho: detection of (co-)orthologs in large-scale analysis. *BMC Bioinformatics*, 12:124, 2011.

[106] L. Li, C. J. Stoeckert, and D. Roos. OrthoMCL: identification of ortholog groups for eukaryotic genomes. *Genome Research*, 13:2178-2189, 2003.

[107] B. Ma, M. Li, and L. Zhang. From gene trees to species trees. *SIAM J. on Comput.*, 30:729–752, 2000.

[108] J. Ma, A. Ratan, L. Zhang, W. Miller, and D. Haussler. A heuristic algorithm for reconstructing ancestral gene orders with duplications. In G. Tesler and D. Durand, editors, *Comparative Genomics*, volume 4751 of *Lecture Notes in Computer Science*, pages 122–135. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-74959-2. doi: 10.1007/978-3-540-74960-8_10.

[109] W. Maddison. Gene trees in species trees. *Systematic Biology*, 46(3):523–536, 1997. doi:10.1093/sysbio/46.3.523.

[110] O. Mahmudi, J. Sjöstrand, B. Sennblad, and J. Lagergren. Genome-wide probabilistic reconciliation analysis across vertebrates. *BMC bioinformatics*, 14(Suppl 15):S10, 2013.

[111] H. Mi, A. Muruganujan, and P. Thomas. Panther in 2013: modeling the evolution of gene function, and other gene attributes, in the context of phylogenetic trees. *Nucleic Acids Res*, 41:D377-D386, 2012.

[112] G. W. Moore, M. Goodman, and J. Barnabas. An iterative approach from the standpoint of the additive hypothesis to the dendrogram problem posed by molecular data sets. *Journal of Theoretical Biology*, 38(3):423–457, 1973.

[113] M. Muffato, A. Louis, C.-E. Poisnel, and H. R. Crollius. Genomicus: a database and a browser to study gene synteny in modern and ancestral genomes. *Bioinformatics*, 26(8):1119–1121, 2010. doi: 10.1093/bioinformatics/btq079.

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

[115] M. Ng and N. Wormald. Reconstruction of rooted trees from subtrees. *Discrete Appl. Math*, 69:19- 31, 1996.

[116] N. Nguyen, S. Mirarab, and T. Warnow. MRL and SuperFine+MRL: new supertree methods. *J. Algo. for Mol. Biol.*, 7(3), 2012.

[117] T. Nguyen, V. Ranwez, S. Pointet, A.-M. A. Chifolleau, J.-P. Doyon, and V. Berry. Reconciliation and local gene tree rearrangement can be of mutual profit. *Algorithms Mol Biol*, 8(1):12, Apr 2013. doi: 10.1186/1748-7188-8-12.

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

[119] T. H. Ogden and M. S. Rosenberg. Multiple sequence alignment accuracy and phylogenetic inference. *Systematic Biology*, 55(2):314–328, 2006.

[120] S. Ohno. *Evolution by gene duplication*. Springer, Berlin, 1970.

[121] R. Overbeek, M. Fonstein, M. D'souza, G. D. Pusch, and N. Maltsev. The use of gene clusters to infer functional coupling. *Proceedings of the National Academy of Sciences*, 96(6):2896–2901, 1999.

[122] S. Penel, A.-M. Arigon, J.-F. Dufayard, A.-S. Sertier, V. Daubin, L. Duret, M. Gouy, and G. Perrière. Databases of homologous gene families for comparative genomics. *BMC Bioinformatics*, 10 Suppl 6:S3, 2009. doi:10.1186/1471-2105-10-S6-S3.

[123] H. Philippe, H. Brinkmann, D. V. Lavrov, D. T. J. Littlewood, M. Manuel, G. Wörheide, and D. Baurain. Resolving difficult phylogenetic questions: why more sequences are not enough. *PLoS Biol*, 9(3):e1000602, 2011.

[124] P. Piazza, S. Jasinski, and M. Tsiantis. Evolution of leaf developmental mechanisms. *New Phytologist*, 167(3):693–710, 2005.

[125] D. Pisani, J. A. Cotton, and J. O. McInerney. Supertrees disentangle the chimerical origin of eukaryotic genomes. *Molecular Biology and Evolution*, 24(8):1752–1760, 2007.

[126] J. F. Poyatos and L. D. Hurst. The determinants of gene order conservation in yeasts. *Genome Biol*, 8(11):R233, 2007.

[127] L. Pryszcz, J. Huerta-Cepas, and T. Gabaldón. MetaPhOrs: orthology nd paralogy predictions from multiple phylogenetic evidence using a consistency-based confidence score. *Nucleic Acids Research*, 39:e32, 2011.

[128] O. Pulkkinen and R. Metzler. Distance matters: the impact of gene proximity in bacterial gene regulation. *Physical review letters*, 110(19):198101, 2013.

[129] V. Ranwez, V. Berry, A. Criscuolo, P. Fabre, S. Guillemot, C. Scornavacca, and E. Douzery. PhySIC: a veto supertree method with desirable properties. *Syst. Biol.*, 56(5):798- 817, 2007.

[130] V. Ranwez, A. Criscuolo, and E. Douzery. SuperTriplets: a triplet-based supertree approach to phylogenomics. *Bioinformatics*, 26(12):i115- i123, 2010.

[131] M. D. Rasmussen and M. Kellis. A bayesian approach for fast and accurate gene tree reconstruction. *Molecular Biology and Evolution*, 28(1):273–290, 2011. doi:10.1093/molbev/msq189.

[132] M. D. Rasmussen and M. Kellis. Unified modeling of gene duplication, loss, and coalescence using a locus tree. *Genome Research*, 22(4):755–765, 2012.

[133] D. Robinson and L. Foulds. Comparison of phylogenetic trees. *Mathematical Biosciences*, 53(1–2):131 – 147, 1981. ISSN 0025-5564. doi: http://dx.doi.org/10.1016/0025-5564(81)90043-2.

[134] D. F. Robinson. Comparison of labeled trees with valency three. *Journal of Combinatorial Theory, Series B*, 11(2):105–119, 1971.

[135] F. Ronquist and J. P. Huelsenbeck. Mrbayes 3: Bayesian phylogenetic inference under mixed models. *Bioinformatics*, 19(12):1572–1574, 2003.

[136] B. Roure, D. Baurain, and H. Philippe. Impact of missing data on phylogenies inferred from empirical phylogenomic data sets. *Molecular biology and evolution*, 30(1):197–214, 2013.

[137] R. Ruan, H. Li, and Z. C. *et al.*. TreeFam: 2008 update. *Nucleic Acids Research*, 36(suppl. 1):D735-D740, 2008.

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

[139] D. Sankoff and P. Rousseau. Locating the vertices of a steiner tree in an arbitrary metric space. *Mathematical Programming*, 9(1):240–246, 1975.

[140] F. Schreiber, M. Patricio, M. Muffato, M. Pignatelli, and A. Bateman. TreeFam v9: a new website, more species and orthology-on-the-fly. *Nucleic Acids Res*, 42:D922-D925, 2013.

[141] C. Scornavacca, E. Jacox, and G. J. Szollosi. Joint amalgamation of most parsimonious reconciled gene trees. *Bioinformatics*, page btu728, 2014.

[142] C. Semple. Reconstructing minimal rooted trees. *Discrete Appl. Math.*, 127(3), 2003.

[143] C. Semple and M. Steel. A supertree method for rooted trees. *Discrete Applied Mathematics*, 105(1):147–158, 2000.

[144] C. Semple and M. Steel. Phylogenetics. *Oxford Lecture Series in Mathematics and in Applications*, 24:119- 120, 2003. Oxford, UK: Oxford University Press.

[145] B. Sennblad and J. Lagergren. Probabilistic orthology analysis. *Systematic biology*, 58(4):411–424, 2009.

[146] H. Shimodaira and M. Hasegawa. Multiple comparisons of log-likelihoods with applications to phylogenetic inference. *Molecular biology and evolution*, 16:1114–1116, 1999.

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

[148] J. Slowinski. Molecular polytomies. *Molecular Phylogenetics and Evolution*, 19(1):114-120, 2001.

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

[150] R. R. Sokal and C. D. Michener. A statistical method for evaluating systematic relationships. *University of Kansas Scientific Bulletin*, 28:1409–1438, 1958.

[151] E. L. Stanley, A. M. Bauer, T. R. Jackman, W. R. Branch, and P. L. F. N. Mouton. Between a rock and a hard polytomy: rapid radiation in the rupicolous girdled lizards (squamata: Cordylidae). *Molecular phylogenetics and evolution*, 58(1):53–70, 2011.

[152] M. Steel and A. Rodrigo. Maximum likelihood supertrees. *Syst. Biol.*, 57(2):243- 250, 2008.

[153] M. Stolzer, H. L. H, M. Xu, D. Sathaye, B. Vernot, and D. Durand. Inferring duplications, losses, transfers and incomplete lineage sorting with nonbinary species trees. *Bioinformatics*, 28(18):i409-i415, 2012.

[154] K. M. Swenson and N. El-Mabrouk. Gene trees and species trees: Irreconcilable differences. *BMC Bioinformatics*, 13((Suppl 19)):S15, 2012.

[155] M. Swenson, R. Suri, C. Linder, and T. Warnow. SuperFine: fast and accurate supertree estimation. *Sys. Biol.*, 61(2):214-227, 2012.

[156] D. Swofford, G. Olsen, P. Waddell, and D. Hillis. Phylogenetic inference. *Molecular Systematics*, pages 407–513, 1996.

[157] G. J. Szöllosi, W. Rosikiewicz, B. Bousseau, E. Tannier, and V. Daubin. Efficient exploration of the space of reconciled gene trees, 2013. Submitted.

[158] G. J. Szollosi, E. Tannier, N. Lartillot, and V. Daubin. Lateral gene transfer from the dead. *Systematic biology*, page syt003, 2013.

[159] K. Tamura, D. Peterson, N. Peterson, G. Stecher, M. Nei, and S. Kumar. Mega5: molecular evolutionary genetics analysis using maximum likelihood, evolutionary distance, and maximum parsimony methods. *Molecular biology and evolution*, 28(10):2731–2739, 2011.

[160] R. Tatusov, M. Galperin, D. Natale, and E. Koonin. The COG database: a tool for genome-scale analysis of protein functions and evolution. *Nucleic Acids Research*, 28:33- 36, 2000.

[161] P. Thomas. GIGA: a simple, efficient algorithm for gene tree inference in the genomic age. *BMC Bioinformatics*, 11:312, 2010.

[162] S. Van Dongen. A cluster algorithm for graphs. *Report-Information systems*, (10):1–40, 2000.

[163] B. Vernot, M. Stolzer, A. Goldman, and D. Durand. Reconciliation with non-binary species trees. *J. Comput. Biol.*, 15:981–1006, 2008.

[164] A. Vilella, J. Severin, A. Ureta-Vidal, L. Heng, and E. Birney. EnsemblCompara GeneTrees: Complete, duplication-aware phylogenetic trees in vertebrates. *Genome Research*, 19(2):327–335, 2009. doi: 10.1101/gr.073585.107.

[165] A. Vilella, J. Severin, A. Ureta-Vidal, L. Heng, R. Durbin, and E. Birney. EnsemblCompara gene trees: Complete, duplication-aware phylogenetic trees in vertebrates. *Genome Research*, 19:327-335, 2009.

[166] L. Wang and T. Jiang. On the complexity of multiple sequence alignment. *Journal of computational biology*, 1(4):337–348, 1994.

[167] I. Wapinski, A. Pfeffer, N. Friedman, and A. Regev. Automatic genome-wide reconstruction of phylogenetic gene trees. *Bioinformatics*, 23(13):i549-i558, 2007.

[168] J. J. Wiens. Re-evolution of lost mandibular teeth in frogs after more than 200 million years, and re-evaluating dollo's law. *Evolution*, 65(5):1283–1296, 2011.

[169] Y.-C. Wu, M. Rasmussen, M. Bansal, and M. Kellis. Treefix: Statistically informed gene tree error correction using species trees. *Syst. Biol.*, 62(1):110-120, 2012.

[170] Y.-C. Wu, M. D. Rasmussen, M. S. Bansal, and M. Kellis. Most parsimonious reconciliation in the presence of gene duplication, loss, and deep coalescence using labeled coalescent trees. *Genome research*, 24(3):475–486, 2014.

[171] Z. Yang. Statistical properties of the maximum likelihood method of phylogenetic estimation and comparison with distance matrix methods. *Systematic biology*, 43(3):329–342, 1994.

[172] T. Zerjal, Y. Xue, G. Bertorelle, R. S. Wells, W. Bao, S. Zhu, R. Qamar, Q. Ayub, A. Mohyuddin, S. Fu, et al. The genetic legacy of the mongols. *The American Journal of Human Genetics*, 72(3):717–721, 2003.

[173] J. Zhang. Evolution by gene duplication: an update. *TRENDS in Ecology and Evolution*, 18(6):292- 298, 2003.

[174] L. Zhang. On Mirkin-Muchnik-Smith conjecture for comparing molecular phylogenies. *Journal of Computational Biology*, 4:177–188., 1997.

[175] Y. Zheng, T. Wu, and L. Zhang. Reconciliation of gene and species trees with polytomies. *arXiv preprint arXiv:1201.3995*, 2012.

[176] Y. Zheng, T. Wu, and L. Zhang. A linear-time algorithm for reconciliation of non-binary gene tree and binary species tree. In *Combinatorial Optimization and Applications*, volume 8287 of *LNCS*, pages 190-201. 2013.

[177] Y. Zheng and L. Zhang. Reconciliation with non-binary gene trees revisited. In *Research in Computational Molecular Biology*, pages 418–432. Springer, 2014.

[178] C. M. Zmasek and S. R. Eddy. A simple algorithm to infer gene duplication and speciation events on a gene tree. *Bioinformatics*, 17:821–828, 2001.

[179] D. Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 681–690. ACM, 2006.