UNIVERSITÉ DU QUÉBEC À MONTRÉAL

ALGORITHMES BIOINFORMATIQUES POUR LA RECONSTRUCTION D'ARBRES CONSENSUS ET DE SUPER-ARBRES MULTIPLES

THÈSE
PRÉSENTÉE
COMME EXIGENCE PARTIELLE
DU DOCTORAT EN INFORMATIQUE

PAR

NADIA TAHIRI

MAI 2019

REMERCIEMENTS

Je tiens d'abord à remercier vivement mon directeur de recherche, Dr. Vladimir Makarenkov, pour sa patience sans égale et ses nombreuses suggestions pour la réalisation de mes projets de recherche. Qu'il trouve ici toute l'expression de ma gratitude et de ma profonde reconnaissance.

Je tiens également à exprimer mes plus vifs remerciements à tous mes collègues du laboratoire de bioinformatique à l'Université du Québec à Montréal. Nous avons créé un groupe dynamique et stimulant à l'innovation.

Je désirerais aussi remercier les professeurs qui liront cette thèse. Je leur suis reconnaissante du temps qu'ils ont accordé pour lire ce travail.

Je remercie chaleureusement Brahim Tahiri, Dounia Tahiri, Nancy Badran, Valérie Hay, Bogdan Mazoure et Robert Wagner pour leurs conseils et leurs remarques au sujet de ma thèse de doctorat.

J'adresse ma totale reconnaissance au Fonds Québécois de la Recherche sur la Nature et les Technologies (FQRNT) et à la Fondation de l'Université du Québec à Montréal pour leur support financier.

À toute ma famille, mes amis et tous ceux qui m'ont soutenu durant toutes mes années de doctorat, qu'ils trouvent ici mes remerciements les plus sincères.

À mes parents, Aïcha Tahiri et Mohamed Tahiri

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

Tableau page

LISTE DES ABRÉVIATIONS

Abréviations	Signification
AA	Acide Aminé
ADN	Acide DésoxyriboNucléique
ARI	Adjusted Rand Index (mesure de qualité d'un partitionnement d'objets)
ARN	Acide RiboNucléique
ARNm	Acide RiboNucléique messager
ARNr	Acide RiboNucléique ribosomal
CH	Critère de Caliński-Harabasz (indice)
CPU	Central Processing Unit
DL	Deep Learning (apprentissage profond)
IE	Langues Indo-Européennes
LS	Least-Square distance
MCT	Multiple Consensus Trees
ML	Machine Learning (apprentissage automatique)
MSSA	Most Similar Supertree Approach
MPR	Most Parsimonious Reconstruction
mySQL	my Structured Query Language (langage de description de bases de données)
NGS	Next Generation Sequencing (Séquençage de nouvelle génération)
NJ	Neighbor-Joining (algorithme de regroupement hiérarchique)
OpenMP	Open Multi-Processing

OTU	Operational Taxonomic Unit
PCR	Polymerase Chain Reaction
PEVCCA	*Porifera*, *Echinodermata*, *Vertebrata*, *Cnidaria*, *Crustacea*, et *Annelida*
QD	Quartet distance
QFIT	Quartet Fit (Quartet maximum le plus convenable)
RAM	Random Access Memory
RF	Distance topologique de Robinson et Foulds
RI	Rand Index (mesure de qualité)
SH	Critère Silhouette (indice)
SFIT	Split Fit (Scission maximum la plus convenable)
THG	Transfert Horizontal de Gènes (HGT en anglais)
ToL	Tree of Life project (projet de reconstruction de l'arbre du vivant)
UPGMA	Unweighted Pair Group Method with Arithmetic Mean (algorithme de regroupement hiérarchique)
URL	Uniform Ressource Locator

RÉSUMÉ

Les arbres phylogénétiques (*c.-à-d.*, arbres additifs ou *X*-arbres) contiennent des informations importantes caractérisant l'évolution spécifique des familles de gènes étudiées. Cependant, un *arbre consensus d'espèces* fiable ne peut pas être inféré à partir d'un alignement de séquences multiples d'une famille de gènes unique ou de la concaténation des alignements correspondant à des familles de gènes ayant des histoires évolutives différentes. Ces histoires évolutives peuvent différer, par exemple, à cause des transferts horizontaux subis par certains gènes ou à cause de l'ancienne duplication génique qui provoque l'émergence de paralogues dans un génome. De nombreuses méthodes ont été proposées pour déduire un arbre consensus unique pour un ensemble d'arbres phylogénétiques donné. L'application de ces méthodes classiques peut donc conduire à la perte d'information sur les histoires évolutives spécifiques qui caractérisent certaines familles de gènes ou certains groupes de familles de gènes. Ainsi, la problématique d'inférence des *arbres consensus multiples* devient pertinente.

Dans cette thèse de doctorat, nous présentons de nouveaux algorithmes permettant de raffiner le processus d'inférence d'arbres consensus et de super-arbres obtenus par regroupement d'arbres phylogénétiques. L'avantage de notre approche par rapport aux approches classiques, qui retournent comme solution un arbre consensus ou un super-arbre unique, est qu'elle permet de déterminer un ou plusieurs arbres consensus ou super-arbres représentant au mieux le groupe d'arbres phylogénétiques fournis en entrée. Nous montrons comment les algorithmes de regroupement classiques, les *k*-moyennes et les *k*-médoïdes, peuvent être modifiés pour partitionner des arbres phylogénétiques définis sur un même ensemble de feuilles (problématique de l'arbre consensus) ou sur des ensembles de feuilles différents, mais mutuellement chevauchants (problématique de super-arbre). Dans notre étude, la distance topologique de Robinson et Foulds (classique et normalisée) sera utilisée pour comparer des arbres phylogénétiques. Les trois nouveautés majeures de cette thèse sont les suivantes : 1) le développement des algorithmes rapides de construction d'arbres consensus et de super-arbres multiples; 2) l'adaptation des indices de validité des clusters populaires (*c.-à-d.*, l'indice de Caliński-Harabasz, l'indice Silhouette et la statistique Gap) pour leur utilisation dans les algorithmes de regroupement d'arbres phylogénétiques; 3) l'application des algorithmes proposés aux données réelles montrant comment ils peuvent être utilisés pour détecter des transferts horizontaux de gènes ou des événements d'hybridation.

Mots clés : arbre phylogénétique, arbre consensus, super-arbre, distance de Robinson et Foulds, algorithme des *k*-moyennes, algorithme des *k*-médoïdes, partitionnement d'arbres

ABSTRACT

In this thesis, we present new algorithms to build multiple consensus trees and supertrees obtained by clustering phylogenetic trees (*i.e.*, additive trees or *X*-trees). The advantage of our new methods, compared to classical approaches, which return a single tree solution (*i.e.*, consensus tree or supertree), is that our methods return as solution one or more trees (*i.e.*, consensus trees or supertrees), according to the clustering results. We use the algorithms of *k*-means and *k*-medoids to obtain the optimal partitioning of the set of trees considered. We explain in detail how we proceeded while developing these new phylogenetic tree classification methods and provide the results of our comprehensive simulation studies used to validate our new methods. We apply the proposed algorithms to analyze several real datasets.

Keywords: phylogenetic trees, consensus trees, supertrees, Robinson and Foulds distance, *k*-means algorithm, *k*-medoids algorithm, tree partitioning

INTRODUCTION

Cette thèse porte sur la conception de nouveaux algorithmes permettant de partitionner des arbres phylogénétiques définis sur les ensembles identiques ou différents de feuilles (*c.-à-d.*, espèces ou taxons) en proposant en sortie *une ou plusieurs solutions consensus*. L'objectif premier de la classification d'arbres phylogénétiques est de regrouper des gènes ayant des histoires évolutives communes ou de construire un arbre complet (*c.-à-d.*, un super-arbre regroupant toutes les espèces considérées) à partir d'un ensemble d'arbres partiels. Cette classification sera pertinente pour l'analyse des gènes qui ont subi les mêmes événements évolutifs, ainsi que pour une meilleure inférence de la phylogénie de toutes les espèces connues (*c.-à-d.*, l'inférence de l'*Arbre de Vie*).

Pour mieux comprendre la problématique de partitionnement d'arbres phylogénétiques, nous présentons dans le chapitre I les notions de base nécessaires à sa compréhension. Nous décrivons alors brièvement la phylogénie, les mesures de comparaison d'arbres phylogénétiques, les méthodes de reconstruction d'arbres phylogénétiques et, enfin, les mesures de la validation statistique des phylogénies.

Dans le chapitre II, nous présenterons un état de l'art des méthodes pour l'inférence d'arbres consensus et de super-arbres. Nous explorerons les avantages et les inconvénients relatifs à chaque approche. Nous illustrerons les principaux concepts par des petits exemples afin de saisir les limites majeures de ces méthodes. Nous conclurons ce chapitre en expliquant la nécessité de partitionner les arbres phylogénétiques afin de réaliser leur fusion notamment lors de la construction d'un ou de plusieurs arbres phylogénétiques consensus et/ou la construction d'un ou de plusieurs super-arbres.

Après cette revue de la littérature, nous présentons au chapitre III les contributions que nous voulons apporter à ce domaine de recherche, en décrivant les nouveaux algorithmes de classification d'arbres. Ce chapitre sera divisé en trois parties.

La première partie traitera la question de partitionnement d'arbres phylogénétiques en utilisant l'algorithme des k-médoïdes et mettant en avant la possibilité d'obtenir un ou plusieurs arbres consensus pour un ensemble d'arbres phylogénétiques définis sur l'ensemble de feuilles identiques. Nous décrirons également les différents critères mis en place pour ce projet ainsi que l'algorithme employé. La seconde partie expliquera le protocole de simulations conduites afin de valider notre approche. Finalement, la troisième partie analysera les résultats obtenus et permettra ainsi de conclure sur l'approche des k-moyennes.

Au chapitre IV, nous élargirons notre approche de partitionnement d'arbres à l'algorithme des k-moyennes en considérant les mêmes critères de sélection de bons partitionnements d'arbres et en y incorporant de nouveaux. Comme dans le chapitre précédent, nous y présenterons les trois mêmes parties.

Au chapitre V, nous affinerons nos résultats obtenus au chapitre précédent en encadrant notre fonction objective par deux bornes (inférieure et supérieure) et également en élargissant la problématique afin de traiter le problème de la classification des super-arbres. Bien que cette seconde approche semble être une généralisation de la première, des considérations sur la complexité algorithmique motivent la nécessité de traiter les deux cas séparément.

Finalement, le chapitre VI, plus applicatif, montrera comment appliquer nos algorithmes à des données réelles pour identifier des ensembles de gènes ayant des histoires évolutives identiques. Par la suite, nous appliquerons nos algorithmes pour analyser des données linguistiques reliées à la classification des langues Indo-Européennes.

Cette thèse inclut le texte partiel ou complet de publications suivantes :

Chapitre III

L'article suivant a été accepté pour publication dans la revue BMC Evolutionary Biology. J'ai réalisé l'ensemble de la recherche, sous la supervision de mon directeur de recherche.

Tahiri, N., Willems, M., et Makarenkov, V., (2018). A new fast method for inferring multiple consensus trees using *k*-medoids, accepté pour publication à BMC Evolutionary Biology.

Chapitre IV

Tahiri, N., Willems, M., et Makarenkov, V., (2014). Classification d'arbres phylogénétiques basée sur l'algorithme des *k*-moyennes. Actes des XXIèmes Rencontres de la Société Francophone de Classification, CNRST et ENSA de Rabat, Maroc, pages 49-54.

Tahiri, N., Willems, M., et Makarenkov, V., (2015). A new fast method for building multiple consensus trees using *k*-means. Actes de la Première Rencontre de la Société Marocaine de Classification, ENSA de Tanger, Maroc.

Chapitre V

Tahiri, N., Willems, M., et Makarenkov, V., (2015). Inférence de super-arbres phylogénétiques multiples en utilisant l'algorithme des *k*-moyennes. Actes des

XXIIèmes Rencontres de la Société Francophone de Classification, MSH Guépin de Nantes, France, pages 110-114.

Chapitre VI

Tahiri, N., Willems, M., Boc, A., et Makarenkov, V., (2012). Classification des langues Indo-Européennes basée sur un modèle d'identification de transferts horizontaux de gènes. Actes des XIXème Rencontres de la Société Francophone de Classification, ECM et LIF de Marseille, France.

CHAPITRE I

NOTIONS DE BASE

«Les vérités différentes en apparence sont comme d'innombrables feuilles qui paraissent différentes et qui sont sur le même arbre.», Gandhi, 1930.

1.1 Introduction

Afin de proposer une solution originale et efficace au problème de l'inférence des arbres consensus multiples, nous nous sommes intéressés à différents sujets d'étude que nous présentons brièvement dans ce chapitre. Tout d'abord, nous verrons les éléments de base de la phylogénie, soient la terminologie et les mesures de comparaison d'arbres. Par la suite, nous parlerons de la reconstruction d'arbres phylogénétiques par les méthodes d'inférence les plus populaires. Enfin, nous décrirons sommairement les approches permettant la validation statistique d'arbres phylogénétiques.

1.2 La phylogénie

Le terme phylogénie fut introduit pour la première fois par Haeckel en 1866, qui l'a définie comme « *l'histoire du développement paléontologique des organismes par analogie avec l'ontogénie ou histoire du développement individuel* ». Le terme

phylogénie est composé de deux mots grecs : phylon (« tribu », « race », « espèce ») et genesis (« genèse », « création »). La phylogénie correspond à l'inférence de l'histoire évolutive de tous les êtres vivants. L'arbre phylogénétique définit le concept de « descendance des espèces avec modification de leurs caractères ». Ce concept met en avant deux mécanismes : 1) la transmission de caractères d'une génération à l'autre à travers les mécanismes de l'hérédité (« descendance »); 2) les « modifications » des caractères lors de la transmission (*p. ex.* mutations). L'arbre phylogénétique est donc une représentation graphique de la phylogénie d'un groupe de taxons (*c.-à-d.*, espèces).

Un arbre phylogénétique est composé de quatre principaux éléments :

- Les **feuilles** (ou les **nœuds externes**) sont la représentation d'unités taxonomiques – OTU (*Operational Taxonomic Unit*). Il s'agit d'espèces actuelles ou éteintes, pour lesquelles on dispose d'informations génétiques, voir l'illustration à la figure 1.1. Dans un arbre phylogénétique binaire non enraciné avec n espèces, il y a n feuilles. Un arbre enraciné T est dit binaire si chaque nœud qui n'est pas une feuille a exactement deux enfants.

- Les **branches** (ou les **arêtes**) représentent les relations entre les taxons d'un point de vue de descendance. Dans un arbre phylogénétique binaire non enraciné avec n espèces, il y a $2n–3$ branches. Les branches ont de nombreuses caractéristiques qui sont :

    - La **taille** de la branche constitue une distance d'évolution (*c.-à-d.*, le temps nécessaire aboutissant à une espèce donnée). Cette taille mesure un nombre de mutations ponctuelles que l'on va assimiler au temps évolutif.

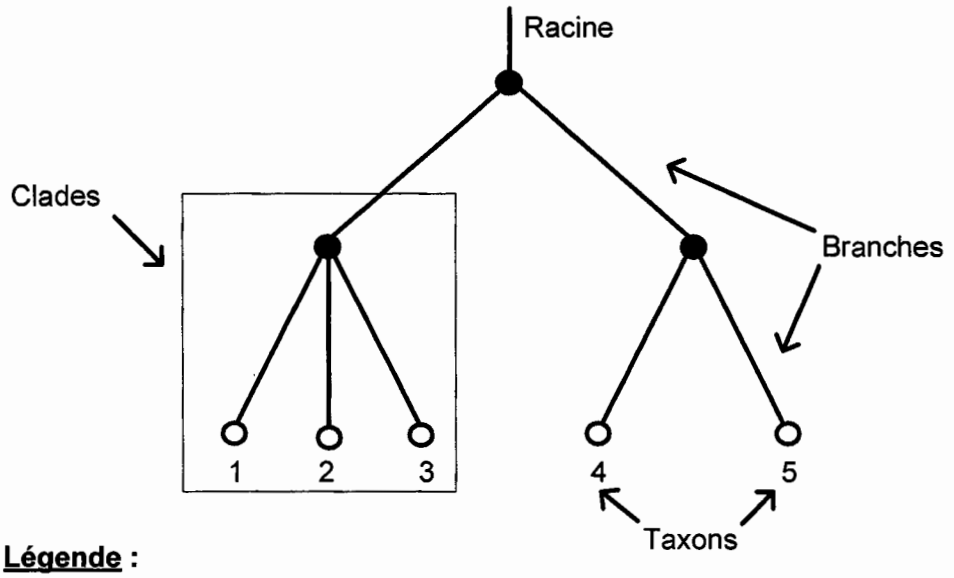
- Le **poids** correspond à une pondération qu'on affecte à une branche. Il peut s'agir de la vitesse d'évolution.

- L'**orientation** des branches illustre la direction de la branche. Par définition, la direction de la branche se fera toujours du nœud de l'ancêtre vers le nœud du descendant.

- Le **degré d'un nœud** ou le nombre de branches attachées à un sommet est le nombre de branches adjacentes à ce nœud. Les nœuds internes ont un degré 3 pour les arbres phylogénétiques résolus (ou binaires) et un degré supérieur à 3 pour les arbres phylogénétiques non résolus (représentant ainsi la divergence simultanée ou l'incertitude).

- Les **nœuds internes** sont associés à des ancêtres hypothétiques ou virtuels. Il s'agit souvent de l'ancêtre commun le plus récent; ou *most recent common ancestor* (Daskalakis et Roch, 2013). Dans un arbre phylogénétique binaire non enraciné avec n feuilles, il y a n-2 nœuds internes.

- La **racine** représente l'ancêtre commun de toutes les espèces étudiées. La racine peut être fixée par la méthode du *point médian* ou la méthode de l'*outgroup*. La première méthode consiste à affecter à chaque nœud interne de l'arbre une séquence consensus des séquences de ses nœuds fils, et à choisir comme racine le nœud dont la séquence est la plus proche de la séquence consensus de toutes les feuilles. La séquence consensus est une séquence nucléotidique ou protéique représentative (*c.-à-d.*, à chaque position de la séquence représentera l'élément le plus fréquent) de l'ensemble des séquences de chaque espèce étudiée. La deuxième méthode consiste à ajouter aux séquences des espèces traitées, une séquence très éloignée. Cette nouvelle séquence permettra de placer la racine sur la branche réunissant cette nouvelle espèce à l'ensemble des espèces étudiées.



Légende :

● Nœud interne (ancêtre hypothétique)

○ Feuille (entité éteinte ou actuelle pour laquelle on dispose des informations)

Figure 1.1 Représentation d'un arbre phylogénétique enraciné.

Nous donnons ici quelques définitions de base concernant les arbres phylogénétiques (*c.-à-d.*, arbres additifs ou *X*-arbres) et leurs métriques en utilisant la terminologie de Barthélemy et Guénoche (1988, 1991).

Définition 1.1. Soit $d(x,y)$ la distance entre le sommet x et le sommet y dans un arbre phylogénétique (*c.-à-d.*, arbre additif) T. Cette distance est définie par la somme de toutes les longueurs des arêtes du chemin unique liant le sommet x au sommet y dans T. Un tel chemin est noté (x,y). Une feuille est un sommet de degré un. Par exemple, la figure 1.4 illustre un processus d'inférence d'arbre phylogénétique à partir d'une matrice de distances du tableau 1.2.

Définition 1.2. Soit X un ensemble fini de n taxons et d la dissimilarité sur l'ensemble X, qui est une fonction sur $(X \times X)$ $R+$ telle que tout x et tout y appartiennent à X et :

$$d(x, y) = d(y, x), \tag{1.1}$$

$$\text{en plus, } d(x, x) = d(y, y) = 0, \text{ et} \tag{1.2}$$

$$d(y, y) \leq d(y, x). \tag{1.3}$$

Définition 1.3. La dissimilarité d sur l'ensemble X satisfait la condition des quatre points si pour tout x, y, z et w de X :

$$d(x, y) + d(z, w) \leq Max\{d(x, z) + d(y, w); d(x, w) + d(y, z)\}. \tag{1.4}$$

La distance satisfaisant la condition des quatre points s'appelle *la distance d'arbre.*

Le théorème principal mettant en correspondance la condition des quatre points avec la représentabilité d'une dissimilarité par un arbre phylogénétique (*c.-à-d.*, une phylogénie) est défini comme suit :

Théorème 1.1. *Toute dissimilarité d satisfaisant la condition des quatre points peut être représentée par un arbre phylogénétique T tel que pour tout x et y appartenant à X, la valeur d(x,y) est égale à la longueur du chemin liant les feuilles x et y dans T (Zarestskii, Buneman, Patrinos et Hakimi, Dobson). En plus, cet arbre phylogénétique est unique.*

On différencie deux types d'arbres : les arbres enracinés et les arbres non enracinés. Un arbre enraciné contient une racine (*c.-à-d.*, un ancêtre commun) identifiée. Cet arbre est orienté et cette orientation correspond au temps d'évolution des taxons. Cet arbre reflète les relations ancêtres – descendants pour chaque branche. Le nombre de différents arbres phylogénétiques binaires (*c.-à-d., dichotomiques*) enracinés est défini par la formule suivante :

$$T_n^{enraciné} = \frac{(2n-3)!}{(2^{(n-2)})(n-2)!}, \tag{1.5}$$

où n est le nombre de taxons ($n \geq 2$).

Souvent, il s'avère hasardeux d'identifier formellement un ancêtre commun pour l'ensemble des taxons étudiés (Fitch, 1971). C'est pour cette raison qu'il est parfois plus souhaitable d'utiliser un arbre non enraciné. Cet arbre ne contient pas une racine et ne peut définir la relation ancêtres – descendants. Le nombre de différents arbres phylogénétiques dichotomiques non enracinés se calcule comme suit :

$$T_n^{non\_enraciné} = \frac{(2n-5)!}{(2^{(n-3)})(n-3)!}, \tag{1.6}$$

où $n \geq 3$ et $T_n^{non\_enraciné} = T_{n-1}^{enraciné}$.

Le tableau 1.1 illustre l'évolution du nombre d'arbres phylogénétiques possibles en fonction du nombre de taxons pour : 1) des arbres dichotomiques enracinés et 2) des arbres dichotomiques non enracinés.

Tableau 1.1 Évolution du nombre d'arbres en fonction du nombre de taxons pour des arbres dichotomiques (enracinés ou non enracinés).

Nombres de taxons dans l'arbre phylogénétique	Nombres d'arbres enracinés possibles	Nombres d'arbres non enracinés possibles
2	1	-
3	3	1
4	15	3
5	105	15
6	945	105
7	10 395	945
8	135 135	10 395
9	2 027 025	135 135
10	34 459 425	2 027 025
15	2.134 E + 14	34 459 425

Le tableau 1.1 indique une augmentation exponentielle du nombre d'arbres possibles en fonction du nombre de taxons, et ceci aussi bien pour des arbres enracinés que pour des arbres non enracinés. La recherche exhaustive d'un arbre optimal s'avère donc très coûteuse en temps et en espace mémoire. C'est pour cela que l'utilisation d'heuristiques (*p. ex.* NJ (*c.-à-d.*, *Neighbor-Joining*) (Saitou et Nei, 1987), UPGMA (*c.-à-d.*, *Unweighted Pair Group Method with Arithmetic Mean*) (Sneath et Sokal, 1973; Zhi-Xiong *et al.*, 2015)), qui se basent sur des algorithmes de distances est préférable quand la distance de départ n'est pas une distance d'arbre.

L'arbre phylogénétique peut être représenté sous différentes formes (*p. ex.* dendrogramme, phénogramme, cladogramme ou phylogramme). Indiquons sommairement les caractéristiques de chacune de ces représentations.

- Le **dendrogramme** est une représentation de l'arbre de manière hiérarchique (Phipps, 1971; Arief *et al.*, 2017), qui est souvent obtenue par des méthodes de regroupements hiérarchiques ou ultramétriques (*p. ex.* UPGMA).

- Le **cladogramme** est une représentation de l'arbre phylogénétique retranscrivant les liens de parenté entre les taxons (Wheeler, 2003; Brower, 2016).

- Le **phénogramme** est une représentation qui est très similaire au cladogramme, mais réalisée à partir d'une taxonomie numérique où les relations expriment un degré de similitude globale (Bhatt et Singh, 2017).

- Le **phylogramme** est une représentation d'un arbre ayant des longueurs de branches proportionnelles au nombre de mutations dans les séquences (Soares, Râbelo et Delbern, 2017).

1.3 Les mesures de comparaison des arbres

Au chapitre IV, nous verrons les approches de regroupements d'arbres phylogénétiques par l'algorithme des *k*-moyennes (MacQueen, 1967). Pour la réalisation de cet algorithme, nous avons besoin au préalable de pouvoir comparer les arbres entre eux. Il existe plusieurs mesures de comparaison d'arbres phylogénétiques. Parmi les plus populaires, nous retrouvons : la distance topologique de Robinson et Foulds (*RF*) (Robinson et Foulds, 1981), la distance des moindres carrés (LS) (Gauss, 1795), la dissimilarité de bipartitions (BD) (Makarenkov, Boc et Diallo, 2007 et Boc, Philippe et Makarenkov, 2010) et la distance de quartets (QD)

(Bryant *et al.*, 2000). Rappelons brièvement les caractéristiques et les procédés associés à chacune de ces mesures.

1.3.1 La distance topologique de Robinson et Foulds

Robinson et Foulds ont proposé en 1981 une mesure permettant de comparer topologiquement deux structures d'arbres phylogénétiques.

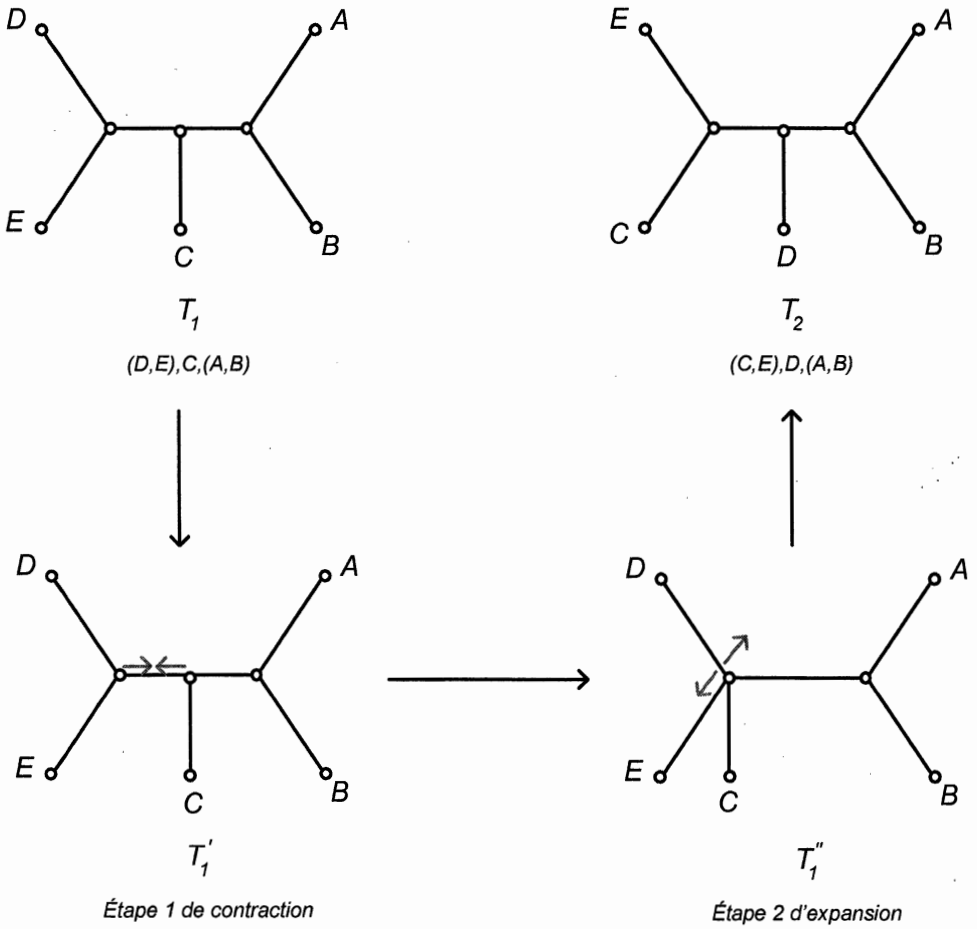


Figure 1.2 Les deux étapes de transformation topologique de l'arbre T_1 en l'arbre T_2 : une contraction de nœuds donnant le premier arbre intermédiaire T_1' et suivie d'une expansion de nœuds donnant le deuxième arbre intermédiaire T_1''. La distance de Robinson et Foulds entre T_1 et T_2 est égale à 2.

La figure 1.2 illustre les deux opérations nécessaires pour transformer topologiquement l'arbre T_1 en l'arbre T_2. Le premier mouvement est la contraction de la branche adjacente à la feuille C et au *cluster* $\{DE\}$. Le deuxième mouvement est l'expansion de la branche adjacente à la feuille D du cluster $\{CDE\}$. Deux opérations élémentaires sont donc nécessaires pour transformer l'arbre T_1 en l'arbre T_2.

La distance de Robinson et Foulds entre deux arbres est égale au nombre minimum d'opérations élémentaires de contraction et d'expansion de nœuds nécessaires pour transformer un arbre phylogénétique en un autre. En même temps, la distance topologique de Robinson et Foulds (*RF*) est une mesure du nombre de bipartitions non triviales, B_1 et B_2, présentes dans les arbres phylogénétiques T_1 et T_2 comparés, où B_1 est le nombre de bipartitions non triviales présentes dans l'arbre T_1, mais absentes dans l'arbre T_2, et inversement, B_2 est le nombre de bipartitions présentes dans l'arbre T_2, mais absentes dans T_1. Les deux arbres phylogénétiques en question doivent avoir le même ensemble de taxons. Si les deux arbres ne contiennent pas le même ensemble de taxons, une étape de prétraitement, consistant en l'extraction des sous-arbres ayant des taxons communs, sera nécessaire. Nous pouvons formaliser la distance *RF* en utilisant l'équation (1.7) suivante :

$$RF = B_1 + B_2, \tag{1.7}$$

où B_1 est le nombre de bipartitions non triviales présentes seulement dans l'arbre phylogénétique T_1 et B_2 est le nombre de bipartitions non triviales présentes seulement dans l'arbre phylogénétique T_2. Cette distance est également connue comme étant une métrique de la différence symétrique (Barthélemy et McMorris, 1986; Kuhner et Yamato, 2015; Vachaspati et Warnow, 2017).

Plus deux arbres phylogénétiques sont proches topologiquement, plus la valeur de la distance *RF* est petite. Il est souvent pertinent de normaliser cette distance par la valeur maximale possible de *RF* (égale à $2n$–6 pour deux arbres binaires à n feuilles) :

$$RF_{normalisée} = \frac{RF}{2n-6} \times 100\%, \quad (1.8)$$

où n est le nombre de taxons identiques dans les deux arbres phylogénétiques comparés.

1.3.2 La distance des moindres carrés

La distance des moindres carrés (*LS*) est un critère métrique qui se définit comme suit :

$$LS = \sum_x \sum_y (d(x,y) - \delta(x,y))^2, \quad (1.9)$$

où $d(x,y)$ est la distance entre les taxons x et y dans l'arbre phylogénétique T_1 et $\delta(x,y)$ est la distance entre les taxons x et y dans l'arbre phylogénétique T_2 défini sur le même ensemble de taxons que T_1. Une version pondérée de ce critère pour le cas des arbres a été développée par Makarenkov et Leclerc en 1999.

1.3.3 La dissimilarité de bipartitions

La dissimilarité de bipartitions, introduite par Makarenkov, Boc et Diallo en 2007, est une autre mesure permettant d'estimer la différence topologique entre deux arbres phylogénétiques. Cette mesure est définie comme suit :

$$bd = \frac{\left(\sum_{a \in BT} Min(Min(d(a,b); d(a,\bar{b}))) + \sum_{b \in BT'} Min(Min(d(b,a); d(b,\bar{a})))\right)}{2}, \quad (1.10)$$

où BT est la table de vecteurs de bipartitions des branches internes de l'arbre T, BT' est la table de vecteurs de bipartitions des branches internes de l'arbre T', $d(x,y)$ est la

distance de Hamming entre les vecteurs a et b, et $\overline{a}$ et $\overline{b}$ sont respectivement les compléments de a et de b. La figure 1.3 présente un exemple de calcul de la dissimilarité de bipartitions entre l'arbre phylogénétique T et l'arbre phylogénétique T'. Chaque ligne de la table de bipartitions correspond à une branche interne de l'arbre et chaque colonne représente un taxon. Les flèches montrent les associations entre les différents vecteurs de bipartitions des deux arbres. La valeur en gras près de chaque vecteur représente la valeur minimale associée.

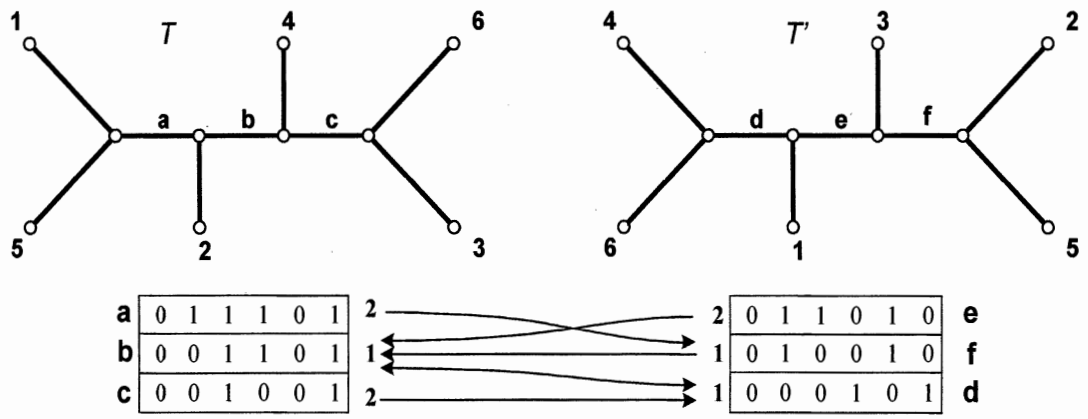


Figure 1.3 Les arbres phylogénétiques T et T' et leurs tables de bipartitions (reproduit d'après Makarenkov, Boc et Diallo, 2007).

Dans l'exemple de la figure 1.3, la dissimilarité de bipartitions entre T et T' est calculée comme suit : $bd(T,T') = \frac{((2+1+2)+(2+1+1))}{2} = 4.5$.

1.3.4 La distance de quartets

La distance de quartets (*QD*), introduite par Bryant en 2000, représente le nombre de sous-arbres à quatre feuilles (*c.-à-d.*, quartets) différents entre les deux arbres phylogénétiques comparés. De nombreuses variantes de la distance *QD* ont été

développées, voir Avni, Cohen et Snir (2014).

1.4 La reconstruction d'arbres phylogénétiques

Il existe différentes méthodes permettant de reconstruire des arbres phylogénétiques (Felsenstein, 2004). Un des enjeux des phylogénéticiens est de produire et de déployer des algorithmes efficaces pour la reconstruction d'arbres phylogénétiques adaptés le plus parfaitement possible aux connaissances biologiques. Initialement, l'inférence des arbres phylogénétiques se basait sur la comparaison morphologique, comportementale et/ou répartition géographique des espèces (Haeckel, 1866).

De nos jours, la reconstruction d'un arbre phylogénétique débute fréquemment par l'analyse des séquences nucléotidiques ou protéiques. Une séquence nucléotidique est l'assemblage linéaire de nucléotides formant l'acide désoxyribonucléique (ADN). Elle est composée de quatre types de bases : les cytosines (C) et les thymines (T) appartenant à la famille des pyrimidines (Y), les adénines (A) et les guanines (G) appartenant à la famille des purines (R). Une séquence protéique est la juxtaposition d'acides aminés (aa) qui sont au nombre de 22. La relation entre une séquence nucléotidique et une séquence d'acides aminés est que la séquence d'ADN peut représenter un gène qui sera exprimé ensuite en une protéine (séquence d'acides aminés).

L'étape de base est d'aligner les séquences homologues données. Cette étape consiste à mettre en correspondance les éléments des séquences de manière à faire correspondre un maximum d'éléments identiques (*c.-à-d.*, *match*) pour l'ensemble des séquences données. Il existe de nombreux logiciels permettant de réaliser ces alignements (*p. ex.* ClustalW (Li, 2003; Thompson, Gibson et Higgins, 2002), MUSCLE (Edgar, 2004), DIALIGN (Morgenstern, 2014), MAFFT (Katoch *et al.*, 2005)). Ils sont tous accessibles depuis le site web de T-Rex

(http://www.trex.uqam.ca/) à l'exception de DIALIGN. Le fondement de cette étape d'alignement est dû aux principes de la biologie moléculaire. Au cours de l'évolution, des mutations se produisent sur les séquences des espèces. Il existe des zones de séquences qui sont plus propices aux modifications (*c.-à-d.*, les *hotspot* (Rogozin *et al.*, 2001)). Les mutations produites sont la cause de la diversité des espèces.

L'étape d'alignement s'avère essentielle pour permettre une meilleure reconstruction de l'arbre d'évolution. Deux types majeurs d'alignements sont possibles : l'alignement global, réalisant l'alignement sur la totalité de la longueur des séquences, et l'alignement local s'appliquant à des fragments des séquences.

Il existe **quatre grandes approches** pour inférer des arbres phylogénétiques : l'approche de distances, le maximum de parcimonie, le maximum de vraisemblance et l'approche bayésienne (Felsenstein, 2004).

1. L'approche de distances ne privilégie aucune hypothèse d'évolution des espèces. Cette approche se contente de mesurer les distances ou les dissimilarités entre les espèces à partir des séquences alignées, et par la suite de reconstruire le meilleur arbre possible à l'aide d'une stratégie de regroupement hiérarchique. Cette approche a été introduite simultanément en 1967 par Cavalli-Sforza et Edwards et par Fitch et Margoliash. Les méthodes de distances sont les méthodes d'inférence d'arbres phylogénétiques les plus rapides. Ces méthodes prennent en entrée une matrice de distances entre les différentes espèces étudiées et détermine un arbre dont les feuilles sont en correspondance avec les distances indiquées (voir la figure 1.4).

	A	B	C	D
B	10			
C	14	8		
D	20	14	14	
E	18	12	12	10

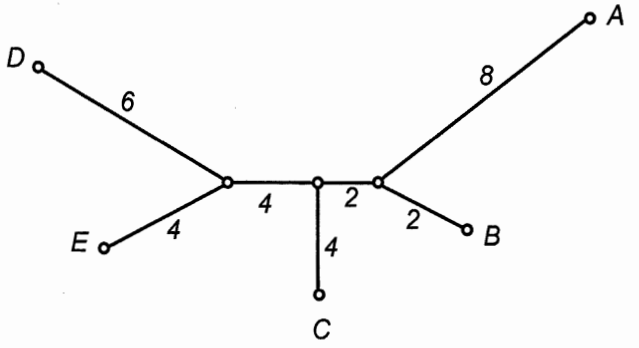


Figure 1.4 Exemple d'une matrice de distances d'arbre sur un ensemble X de 5 taxons (*c.-à-d.*, *input*) et l'arbre phylogénétique associé (*c.-à-d.*, *output*).

Les distances doivent être calculées à partir des séquences qui ont été préalablement alignées. La distance ajustée (retrouvée par la méthode de reconstruction d'arbre) entre deux feuilles correspond à la somme des longueurs des branches du chemin unique entre ces deux feuilles dans l'arbre retrouvé. Cette distance ajustée doit être la plus proche possible à la distance réelle mesurée d'après les séquences. Cependant, si le taux d'évolution n'est pas constant dans l'arbre ou si l'horloge moléculaire n'est pas vérifiée, alors il n'est pas recommandé d'inférer l'arbre phylogénétique en utilisant les méthodes de distances. Les distances peuvent être calculées à partir des séquences en utilisant différentes transformations *séquences-distances* (Jukes et Cantor, 1969; Kimura, 1980). Les deux principales méthodes de distances sont Neighbor-Joining (Saitou et Nei, 1987) et UPGMA (Sneath et Sokal, 1973).

L'algorithme de Neighbor-Joining (NJ) est de loin le plus populaire parmi les méthodes de distances. Un exemple de son exécution est présenté à la figure 1.5. Nous illustrons ici la reconstruction d'un arbre phylogénétique à cinq taxons effectuée à partir de la matrice de distances donnée par le tableau 1.2.

Tableau 1.2 Une matrice de distances pour un arbre phylogénétique à 5 taxons.

	A	B	C	D
B	10			
C	14	8		
D	20	14	14	
E	18	12	12	10

L'étape initiale de NJ considère une structure d'arbre étoilée. L'arbre phylogénétique est construit par l'agglomération successive des taxons. La reconstruction de l'arbre phylogénétique se réalise en plusieurs étapes qui sont comme suit (Saitou et Nei, 1987) :

Étape 1 : calculer la matrice **Q** (formule 1.11), en se basant sur la matrice de distances courante **D** :

$$Q(i,j)=(n-2)d(i,j)-\sum_{k=1}^{n}d(i,k)-\sum_{k=1}^{n}d(j,k), \qquad (1.11)$$

où $d(i,j)$ est la distance entre les taxons i et j.

Étape 2 : trouver la paire de taxons x et y pour lesquels $Q(x,y)$ a la valeur la plus petite. Ces taxons seront reliés à un nœud nouvellement créé (u), qui sera connecté au nœud central de l'arbre.

Étape 3 : calculer la distance entre des taxons x, y et le nouveau nœud u de la façon suivante :

$$\delta(x,u)=\frac{1}{2}d(x,y)+\frac{1}{2(n-2)}\left[\sum_{k=1}^{n}d(x,k)-\sum_{k=1}^{n}d(y,k)\right], \qquad (1.12)$$

et,

$$\delta(y,u) = d(x,y) - \delta(x,u), \tag{1.13}$$

où x et y sont des taxons pariés au nœud nouvellement créé u. Les branches joignant x à u et y à u auront les longueurs respectives $\delta(x,u)$ et $\delta(y,u)$.

Étape 4 : calculer la distance entre chaque taxon k ($k \neq x$ et $k \neq y$) de l'arbre et le nouveau nœud u :

$$d(u,k) = \frac{1}{2}[d(x,k) + d(y,k) - d(x,y)]. \tag{1.14}$$

Si le nombre de nœuds restants est supérieur à 3, recommencer de nouveau l'algorithme depuis l'Étape 1, mais en remplaçant la paire des voisins unis x et y avec un nouveau taxon auxiliaire en utilisant les distances calculées à l'Étape 4. Sinon, connecter les 3 taxons restants dans une seule topologie possible à 3 feuilles.

La figure 1.5 illustre un exemple de la mise en place de l'algorithme de NJ, appliqué à la matrice de distances du tableau 1.2.

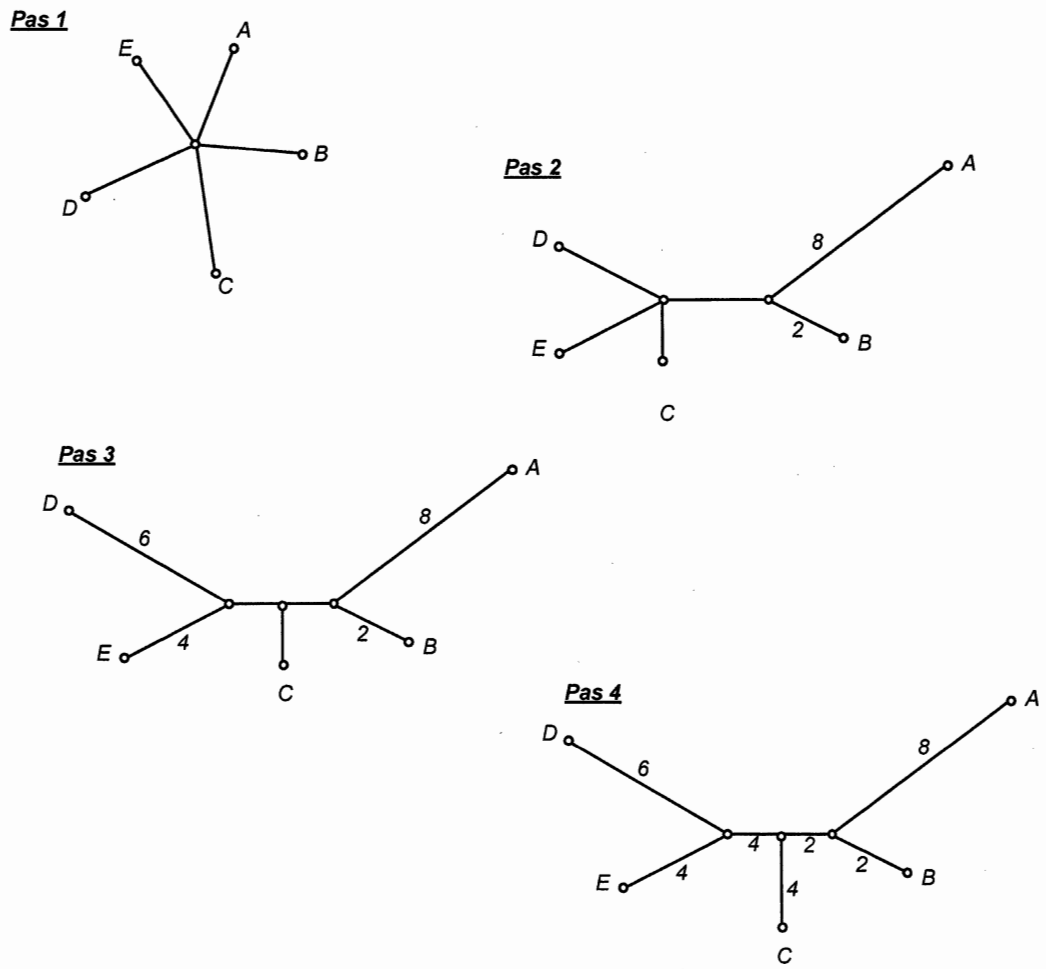


Figure 1.5 Exemple de reconstruction d'un arbre phylogénétique à 5 taxons en utilisant l'algorithme de Neighbor-joining (Saitou et Nei, 1987).

2. L'approche de parcimonie cherche à établir des relations de parenté entre les taxons en s'intéressant aux caractères (*c.-à-d.*, nucléotides ou acides aminés). Tous les scénarios d'évolution sont considérés pour inférer les caractères potentiels à chaque nœud, et l'arbre sélectionné sera celui qui aura le meilleur score de parcimonie en se basant sur un critère préalablement choisi. Contrairement à l'approche de distances qui étudie la parenté entre les taxons en s'intéressant à leur

degré de similarité, l'approche de parcimonie se base sur la généalogie (Fitch, 1971; Czelusniak *et al.*, 1990). À partir des séquences de l'ensemble d'espèces étudiées, une méthode de parcimonie va reconstituer l'arbre pour lequel le scénario d'évolution est le plus parcimonieux (*c.-à-d.*, qui nécessite le moins de modifications des séquences au cours de l'évolution). Il est possible d'associer un coût à chaque type de modification en fonction de leur fréquence et lieu d'apparition au cours de l'évolution. L'arbre ainsi obtenu minimise le nombre de mutations survenues au cours de l'évolution (Sanderson, 2002). La figure 1.6 illustre un exemple d'inférence d'un arbre phylogénétique ayant un scénario d'évolution le plus parcimonieux pour les séquences des feuilles *TTCG*, *TACG*, *TACA* et *GACA*.

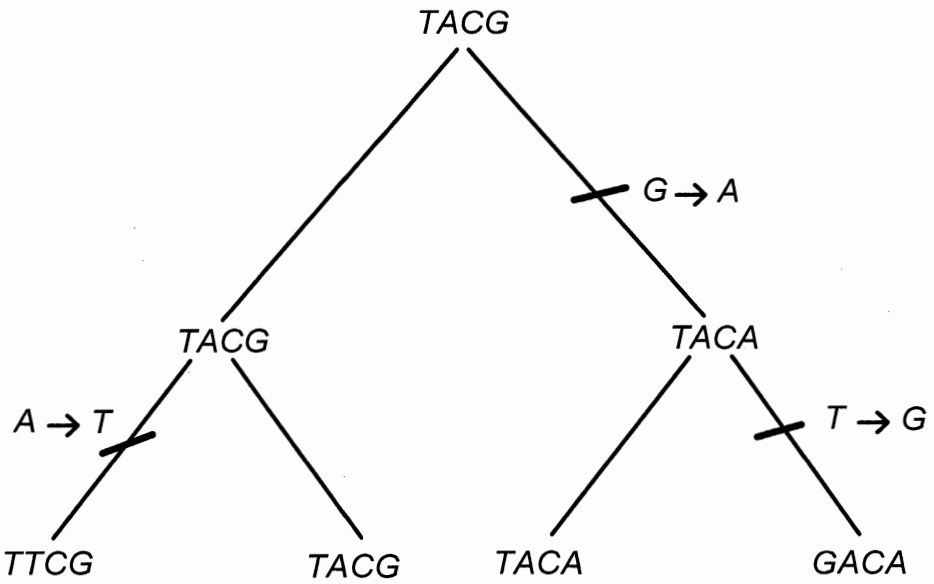


Figure 1.6 Exemple d'inférence d'un arbre phylogénétique le plus parcimonieux.

Les méthodes de maximum de parcimonie les plus fréquemment utilisées sont DNAPARS (pour des données nucléotidiques) (Felsenstein, 1990) et PROTPARS (pour des données protéiques) (Felsenstein, 1990).

3. Le maximum de vraisemblance est une approche mise au point par Felsenstein (1981). Cette approche évalue, en termes de probabilité, l'ordre des branchements et la longueur des arêtes d'un arbre sous un modèle évolutif choisi (Abfalg et Erdfelder, 2012). La probabilité ainsi calculée s'appuie sur la topologie et la longueur des arêtes d'un arbre en fonction d'un modèle évolutif. Le meilleur arbre correspond à l'arbre pour lequel la vraisemblance est maximale. Les méthodes de maximum de vraisemblance les plus connues sont PhyML (Guindon et Gascuel, 2003) et RAxML (Stamatakis, Hoover et Rougemont, 2008). La méthode RAxML est la plus rapide des méthodes de maximum de vraisemblance et est souvent la plus efficace.

4. Les méthodes basées sur l'approche bayésienne (Bernardo et Smith, 1994) sont similaires aux méthodes basées sur le principe du maximum de vraisemblance, à la différence que pour les approches bayésiennes, il faut avoir une connaissance *a priori* (*c.-à-d.*, une *prior*) et une probabilité *a posteriori* (*c.-à-d.*, une *posterior*) (Huelsenbeck et Ronquist, 2001) (formules 1.15) :

$$p(\theta|D)=\frac{p(D|\theta)\times p(\theta)}{p(D)}, \qquad (1.15)$$

où $p(\theta|D)$ représente la probabilité *a posteriori* de l'hypothèse θ sachant les données D (*c.-à-d.*, après avoir pris connaissance des données D, nous évaluerons la plausibilité de l'hypothèse θ), $p(D|\theta)$ représente la probabilité d'observer les données D sachant l'hypothèse θ (*c.-à-d.*, la vraisemblance θ, avec θ fixée), $p(D)$ représente la probabilité *a priori* des données D et $p(\theta)$ représente la probabilité *a priori* de l'hypothèse θ.

Plusieurs de ces quatre approches de reconstruction d'arbres phylogénétique sont implémentées dans la version Web du logiciel T-Rex (Makarenkov, 2001), qui est accessible à l'URL suivant : http://www.trex.uqam.ca. Il existe également d'autres logiciels permettant d'inférer les arbres phylogénétiques (*p. ex.* PAUP (Swofford, 2002) ou Phylip (Felsenstein, 2005)).

1.5 La validation des arbres phylogénétiques

La validation des arbres phylogénétiques se fait selon des procédés de rééchantillonnage des données (Efron, 1979). Les techniques utilisées en analyse phylogénétique s'appellent *bootstrap* ou *jackknife* (Felsenstein, 2004). Les scores de *bootstrap* obtenus pour les branches internes d'un arbre phylogénétiques indiquent le niveau de robustesse de l'arbre à valider. Deux cas sont envisageables :

- Si les modifications lors du rééchantillonnage des données ont une faible influence sur la topologie de l'arbre phylogénétique, cela signifiera que l'arbre estimé est robuste.

- Sinon, les modifications lors du rééchantillonnage entraîneront des modifications significatives de l'arbre. L'arbre estimé sera donc peu robuste, et de ce fait les analyses se basant sur cet arbre seront peu fiables.

Les scores de robustesse, mesurés en utilisant les techniques de *bootstrap* ou de *jackknife*, sont d'une grande importance puisqu'ils expriment formellement un taux de confiance statistique associé à l'analyse phylogénétique effectuée.

1.6 Conclusion

Dans ce chapitre, nous avons décrit, dans un premier temps, la notion de la phylogénie, puis dans un deuxième temps, présenté les différentes mesures de comparaison d'arbres phylogénétiques (la distance topologique de Robinson et Foulds, la distance des moindres carrés, la dissimilarité de bipartitions et la distance de quartets). Ensuite, nous avons décrit les différentes approches de reconstruction d'arbres phylogénétiques (l'approche de distances, l'approche de parcimonie, l'approche de vraisemblance et l'approche bayésienne). Enfin, nous avons rappelé les techniques de validation des arbres phylogénétiques (*bootstrap* et *jackknife*). Toutes ces notions, vues à travers le chapitre I, nous permettront de mieux comprendre la problématique étudiée et prépareront à l'exploration du concept de consensus d'arbres phylogénétiques. Dans le prochain chapitre, nous verrons les fondements de notre problématique de recherche à travers l'introduction à l'inférence d'arbres consensus et de super-arbres phylogénétiques.

CHAPITRE II

INFÉRENCE D'ARBRES CONSENSUS ET DE SUPER-ARBRES : ÉTAT DE L'ART

«*La normalité n'est qu'une question de consensus. Autrement dit, si la plupart des gens pensent qu'une chose est juste, elle devient juste.*», Paulo Coelho, 1998.

2.1 Introduction

Nous présentons ici une revue de la littérature sur les méthodes et les programmes développés pour le partitionnement d'arbres phylogénétiques. Nous exposerons plus en détail les arbres consensus et les méthodes de leur inférence. Ensuite, nous introduirons la notion des super-arbres. Nous conclurons ce chapitre de l'état de l'art en présentant les algorithmes existants pour la construction d'arbres consensus multiples, basés entre autres sur l'algorithme des *k*-moyennes (MacQueen, 1967).

2.2 Les arbres consensus et les méthodes de leur inférence

Au chapitre I, nous avons présenté les principales approches d'inférence d'arbres phylogénétiques. Il est fort probable que les méthodes associées à ces approches ne retournent pas en sortie un arbre phylogénétique unique, mais une collection d'arbres

différents. Il n'existe pas un critère absolu permettant de déterminer si l'un des arbres est meilleur que les autres (à l'exception de l'utilisation des critères intrinsèques, par exemple en utilisant les scores de *bootstrap*). C'est pour cette raison qu'il est préférable de rechercher une représentation d'agrément de ces arbres, telle que leurs parties concordantes apparaissent clairement par rapport aux parties discordantes. La représentation ainsi obtenue est appelée *arbre consensus*. Les méthodes de consensus traditionnelles génèrent un arbre phylogénétique unique, représentatif de l'ensemble d'arbres initiaux (Bryant, 2003). La première méthode de consensus fut proposée par Adams en 1972. Depuis ce moment, une large variété de méthodes a été développée. Il y a eu un débat considérable sur la façon dont elles doivent être utilisées (Bryant, 2003; Dong *et al.*, 2010).

Avant d'introduire les différentes méthodes permettant l'inférence de l'arbre consensus, nous présenterons ici quelques notions utiles concernant les arbres phylogénétiques (*p. ex.* la structure de l'arbre phylogénétique, la notation *Newick*).

Un triplet enraciné (*p. ex.* $AB|C$) dénote un groupement entre A et B relativement à C. Nous disons que le triplet $AB|C$ est un triplet enraciné de l'arbre T si le plus récent ancêtre commun de A et de B est un descendant du plus récent ancêtre commun de A, B et C. L'ensemble de tous les triplets enracinés de l'arbre T est dénoté par $r(T)$.

Le format parenthésé *Newick* (Felsenstein et Sober, 1986) est une représentation standard des arbres phylogénétiques utilisée dans la plupart des logiciels traitant ces arbres. Par exemple la chaîne $((A, B), (C, D))$ définit un arbre enraciné avec les partitions suivantes : $\{A\}$, $\{B\}$, $\{C\}$, $\{D\}$, $\{A, B\}$, $\{C, D\}$ et $\{A, B, C, D\}$. La racine se trouve entre les partitions $\{A, B\}$ et $\{C, D\}$. La notation pour un arbre non enraciné sera similaire, mais en ignorant la position de la racine.

Il existe deux grandes approches pour construire des arbres consensus : l'approche basée sur les scissions et les partitionnements et l'approche basée sur les intersections des partitionnements.

2.2.1 L'approche basée sur les scissions et les partitionnements

Cette approche comprend quatre méthodes principales : la méthode de l'arbre consensus strict (Sokol et Rohlf, 1981; Moon et Eulenstein, 2017), la méthode de l'arbre consensus majoritaire (Margush et McMorris, 1981), la méthode de l'arbre consensus de Nelson (Nelson, 1979; Layeghifard, Peres-Neto et Makarenkov, 2013) et la méthode de l'arbre consensus majoritaire étendu (Felsenstein, 1985).

L'*arbre consensus strict* (ou cladogramme de Nelson) est inféré en ne considérant que les regroupements de taxons qui sont identiques dans tous les arbres comparés. Les parties conflictuelles des arbres phylogénétiques sont représentées par des multifurcations dans un arbre consensus strict.

La figure 2.1 illustre un exemple d'inférence d'un arbre consensus strict à partir des trois arbres phylogénétiques enracinés entièrement dichotomiques (indiqués sur la figure 2.1 comme T_1, T_2 et T_3).

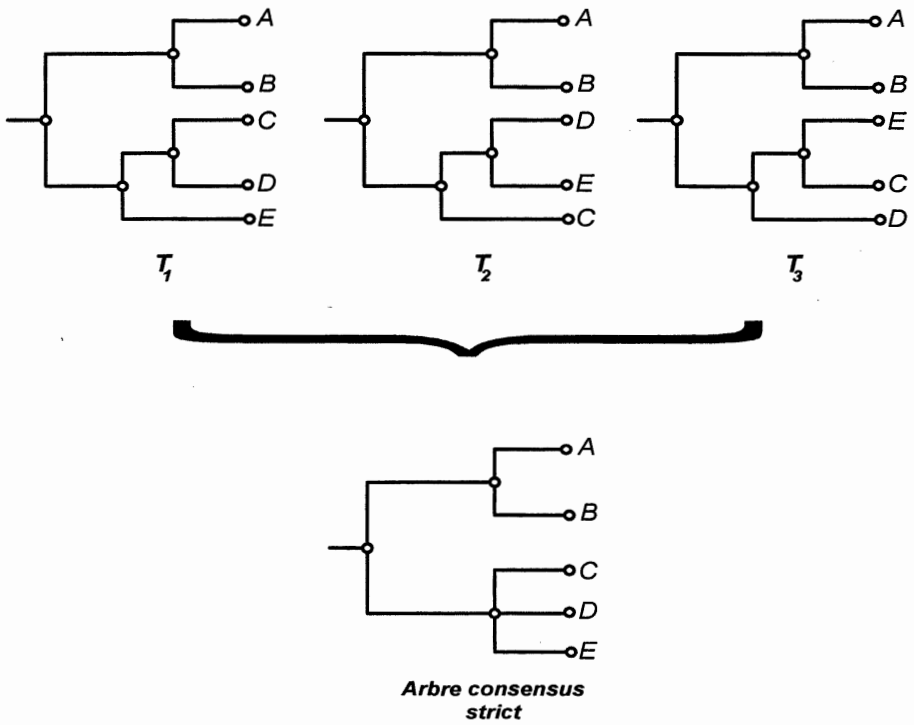


Figure 2.1 Exemple d'inférence d'un arbre consensus strict à partir d'un ensemble de trois arbres phylogénétiques à cinq taxons.

Les trois arbres phylogénétiques T_1, T_2 et T_3 sont définis sur le même ensemble de cinq taxons $\{A, B, C, D, E\}$. Le seul groupement en commun entre tous ces arbres est celui formé par le groupe $\{A, B\}$ d'un côté et par le groupe $\{C, D, E\}$ de l'autre côté. À l'exception de ce groupement, aucun autre groupement non trivial ne se retrouve simultanément dans les arbres T_1, T_2 et T_3. L'arbre consensus strict des trois arbres T_1, T_2 et T_3 regroupera d'une part les taxons A et B et d'autre part mettra au même niveau hiérarchique les taxons C, D et E. Cette multifurcation des taxons C, D et E ne signifie pas une spéciation triple indiquant qu'il existe un sommet à partir duquel les trois taxons auraient évolué simultanément. Cette multifurcation met en avant l'impossibilité de conclure quant aux relations de parenté entre les taxons C, D et E. Cette représentation met en avant uniquement ce qui est en commun entre tous les arbres. Cependant, malgré l'absence d'information sur certains groupements, l'arbre

consensus strict est très fréquemment utilisé lors des opérations de comparaisons d'arbres.

Parfois, il est plus commode d'avoir un critère moins rigoureux que celui employé par l'arbre consensus strict afin de permettre des regroupements qui ne sont pas forcément présents dans tous les arbres. Lors de la comparaison d'un ensemble d'arbres phylogénétiques ayant des topologies différentes, il est possible de rechercher des groupes monophylétiques qui se rencontrent le plus fréquemment (souvent dans plus que 50% d'arbres) parmi l'ensemble des arbres comparés. L'arbre ainsi obtenu est l'*arbre consensus majoritaire.*

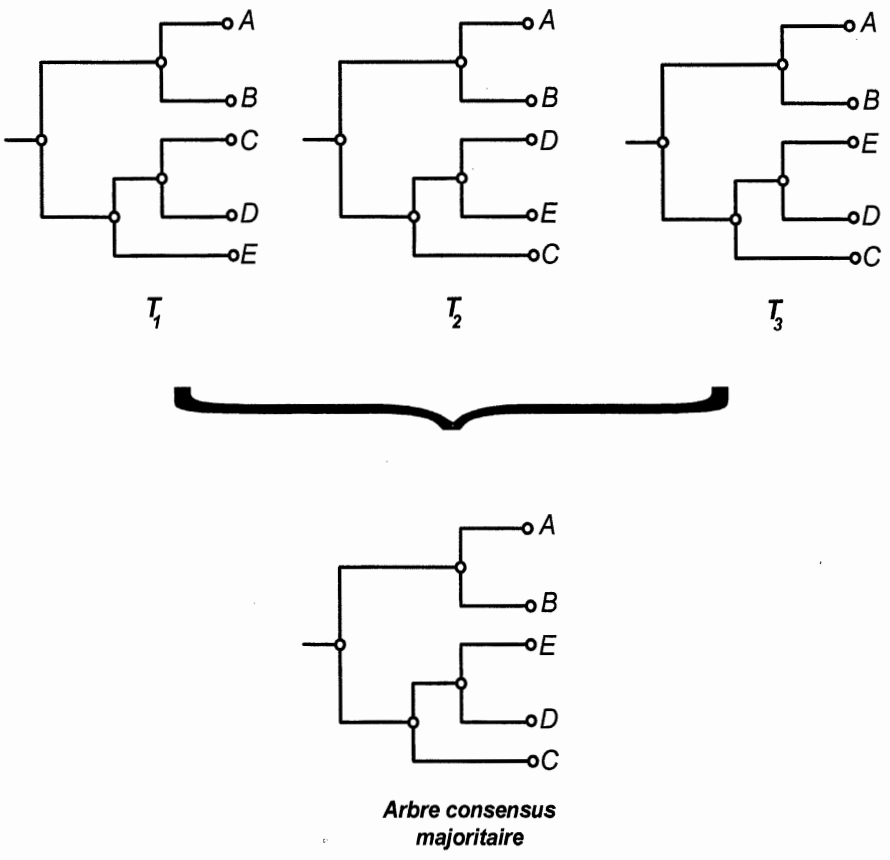


Figure 2.2 Exemple d'inférence d'un arbre consensus majoritaire (50%) à partir d'un ensemble de trois arbres phylogénétiques à cinq taxons.

La figure 2.2 illustre un exemple d'inférence d'un arbre consensus majoritaire à partir de trois arbres phylogénétiques comparés (T_1, T_2 et T_3). Les groupes $\{A, B\}$ et $\{C, D, E\}$ sont présents dans les trois arbres, ce qui se traduit par l'apparition des deux groupes distincts $\{A, B\}$ et $\{C, D, E\}$ dans l'arbre consensus majoritaire. De plus, le groupe $\{D, E\}$ est observé dans deux arbres parmi les trois arbres initiaux. Ce groupe $\{D, E\}$ est donc majoritaire et il sera représenté dans l'arbre consensus majoritaire. La valeur seuil de la règle majoritaire peut être modifiable (généralement, il s'agit de 50 % ou de 75 %). Cette méthode d'inférence d'arbres consensus est généralement utilisée dans les méthodes de rééchantillonnage (voir le chapitre I).

L'*arbre consensus majoritaire étendu* (Felsenstein, 1985) contient toutes les bipartitions majoritaires auxquelles sont ajoutées à tour de rôle les bipartitions compatibles restantes en commençant par les bipartitions les plus fréquentes pour l'ensemble des arbres donnés. Le processus s'arrête lorsqu'un arbre complètement résolu (*c.-à-d.*, binaire) est obtenu. L'arbre consensus majoritaire étendu est le plus fréquemment utilisé en biologie moléculaire, car il est toujours le mieux résolu parmi les trois types d'arbres consensus discutés jusqu'à maintenant.

La méthode de consensus majoritaire étendu fournit souvent des solutions semblables, mais pas nécessairement identiques, à celles fournies par la *méthode de consensus de Nelson* (Nelson, 1979; Page, 1989). La méthode de consensus de Nelson, décrite pour la première fois par (Nelson, 1979), puis généralisée dans (Page, 1989), s'appuie sur les techniques de la théorie des graphes pour trouver une clique de partitions mutuellement compatibles de taille maximum. L'inconvénient majeur de cette méthode de consensus est que ces cliques ne contiennent pas toujours suffisamment de groupes compatibles pour inférer un arbre phylogénétique entièrement résolu (Bryant, 2003). En outre, le problème consistant à trouver une clique de partitions compatibles de taille maximum est un problème NP-difficile (Abello *et al.*, 1994).

2.2.2 L'approche basée sur les intersections des partitionnements

Cette approche comprend une méthode principale : la méthode consensus d'Adams (Adams, 1972). La méthode consensus d'Adams consiste à comparer deux sous-ensembles de taxons qui découlent d'une dichotomie sur un ensemble d'arbres phylogénétiques en partant de la racine. S'il existe un recouvrement entre les sous-ensembles de taxons observés, cela signifie que ce recouvrement constitue *un sous-ensemble consensus* pour ces taxons.

La figure 2.3 illustre un exemple d'inférence d'un arbre consensus d'Adams pour une collection de trois arbres (T_1, T_2 et T_3). Chacun de ces arbres phylogénétiques est défini sur le même ensemble de cinq taxons $\{A, B, C, D, E\}$.

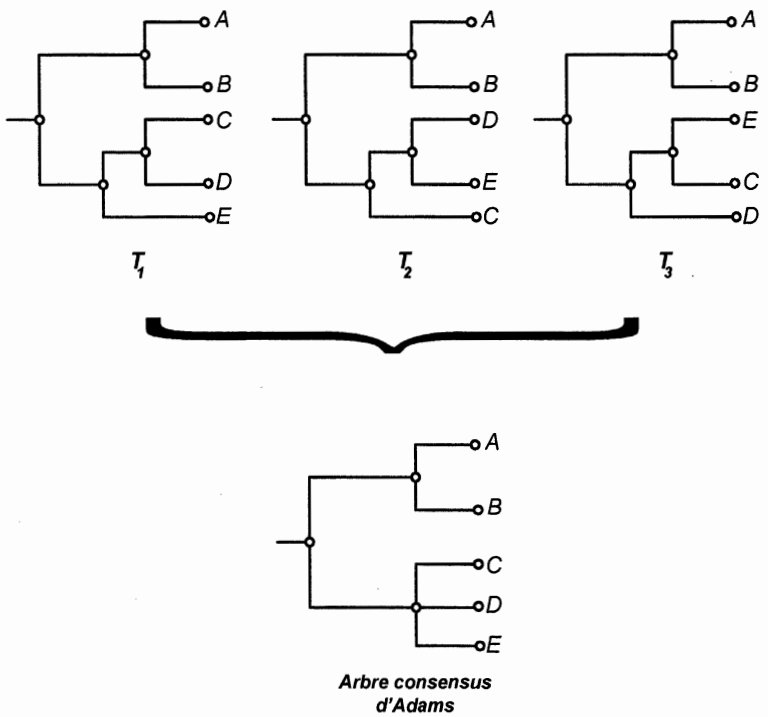


Figure 2.3 Exemple d'inférence d'un arbre consensus d'Adams à partir d'un ensemble de trois arbres phylogénétiques à cinq taxons.

Les arbres T_1, T_2 et T_3 ont deux sous-ensembles de taxons séparés par la branche de la racine qui sont $\{A, B\}$, d'un côté, et $\{C, D, E\}$, de l'autre. En croisant les sous-ensembles de l'un des arbres avec les sous-ensembles des deux autres, nous effectuons des intersections d'ensembles qui définissent, dans cet exemple, deux sous-ensembles : $\{A, B\}$ et $\{C, D, E\}$, identifiant ainsi l'arbre consensus d'Adams.

2.3 La construction de super-arbres

Les méthodes de *super-arbre* réconcilient des arbres phylogénétiques définis sur des ensembles de taxons différents, mais partiellement chevauchants, en une structure unique qui est souvent interprétée comme l'histoire commune d'évolution des espèces (Gordon, 1986; Sanderson, Purvis et Henze, 1998 ; Dong, Fernández-Baca et McMorris, 2010). La problématique de la construction d'un super-arbre est aussi ancienne que le domaine de la systématique. La construction d'un super-arbre reste actuellement la seule approche utilisée lors de l'inférence de l'*Arbre de Vie* (Letunic et Bork, 2016). Depuis le lancement du projet de l'inférence de l'Arbre de Vie (*c.-à-d.*, projet « *Tree of Life* » ou ToL (Letunic et Bork, 2006; Letunic et Bork, 2016)), le nombre d'études portant sur la théorie de super-arbre s'est accru de façon importante. Le projet ToL vise à inférer l'arbre phylogénétique de l'ensemble du vivant et nécessite la collaboration de plusieurs équipes de recherche à travers le monde. L'approche adoptée par ToL consiste à décomposer récursivement le problème de reconstruction de l'Arbre de Vie en plusieurs sous-problèmes, puis à fusionner les résultats de ces analyses individuelles (voir les figures 2.4 et 2.5).

Le terme super-arbre a été formalisé pour la première fois en 1986 par Gordon (Gordon, 1986).

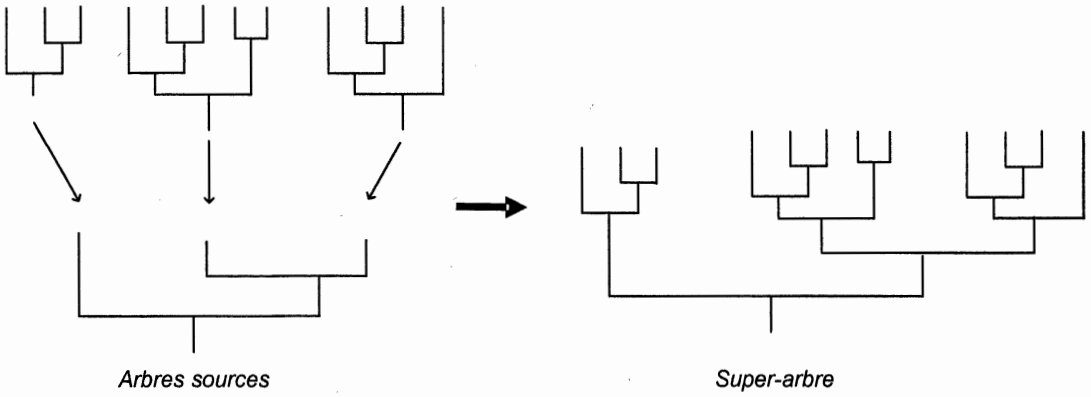


Figure 2.4 Illustration d'une reconstruction d'un super arbre dans le passé (reproduite d'après Bininda-Emonds, 2004).

La figure 2.4 illustre un exemple de reconstruction d'un super-arbre dans le passé (Bininda-Emonds, 2004). Hiérarchiquement, il s'agit de l'imbrication d'arbres phylogénétiques qui étaient collés ensemble dans une forme de substitution taxonomique afin de donner une vue globale de l'Arbre de Vie. Un des inconvénients de cette approche est que l'arbre obtenu ne peut pas expliquer facilement l'estimation des contradictions. Le super-arbre obtenu pourrait donc ne pas refléter fidèlement l'histoire évolutive des espèces. Une décision subjective concernant la meilleure estimation du super-arbre doit aussi être prise et cette décision n'est pas forcément la meilleure.

Des méthodes de reconstruction de super-arbre courantes (la méthode du consensus moyen - AV (Lapointe et Cucumel, 1997), la méthode de l'inférence du super-arbre du plus similaire - MSSA (Creevey *et al.*, 2004), la méthode avec la représentation matricielle la plus parcimonieuse - MRP (Baum, 1992; Ragan, 1992)) choisissent les groupes communs, ou incontestés, dans l'ensemble des arbres sources et trouvent un super-arbre qui se rapproche de l'ensemble des arbres sources en fonction d'un certain critère (voir la figure 2.5). La première méthode de construction de super-arbre présentée par Gordon en 1986 est similaire à la méthode aboutissant à l'arbre

consensus strict dans le sens où sa sortie contient uniquement les relations communes strictes entre tous les arbres phylogénétiques sources. Cependant, cette méthode d'inférence d'un super-arbre strict est limitée par le fait que les arbres sources devraient être compatibles entre eux. La particularité d'un bon super-arbre réside dans sa capacité de bien synthétiser de nombreuses petites disparates provenant des différentes sources d'information contenues dans la collection d'arbres phylogénétiques d'entrée.

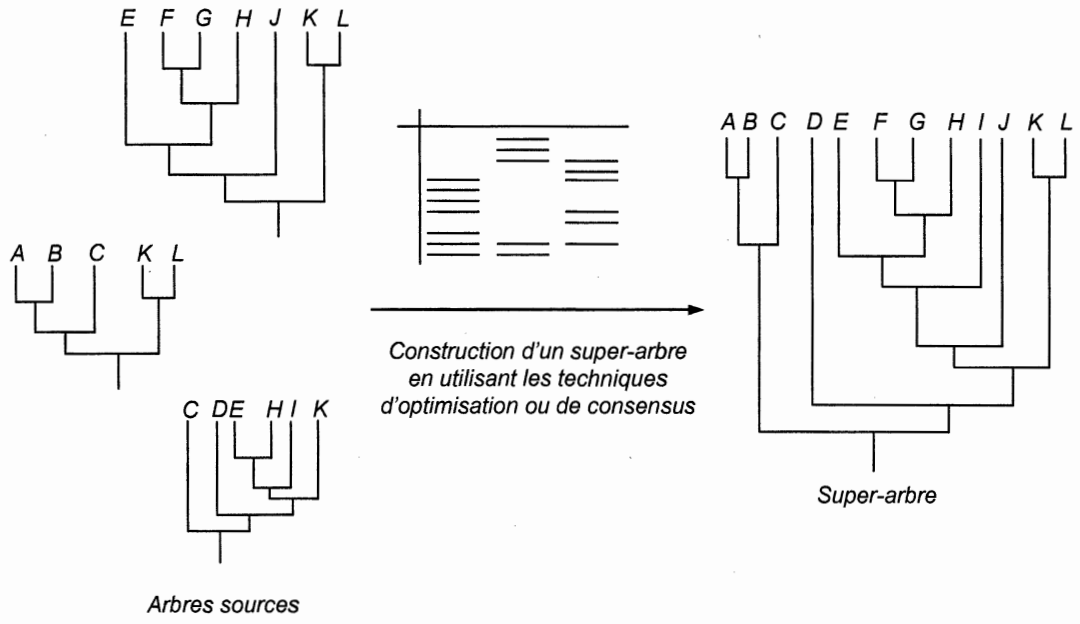


Figure 2.5 Illustration d'une reconstruction d'un super-arbre dans le présent (reproduite d'après Bininda-Emonds, 2004).

Les données sont généralement stockées dans une matrice à laquelle on applique la reconstruction la plus parcimonieuse (*c.-à-d.*, MPR ou *Most Parsimonious Reconstruction* (Ragan, 1992; Doyle, 1992 ; Wilkinson *et al.*, 2005 ; Beck *et al.*, 2006)). La méthode MPR recherche le plus petit nombre de branchements et le plus petit nombre d'événements de mutations pour expliquer les séquences

contemporaines. MPR reste de loin la méthode la plus populaire pour la reconstruction de super-arbres (voir la figure 2.6).

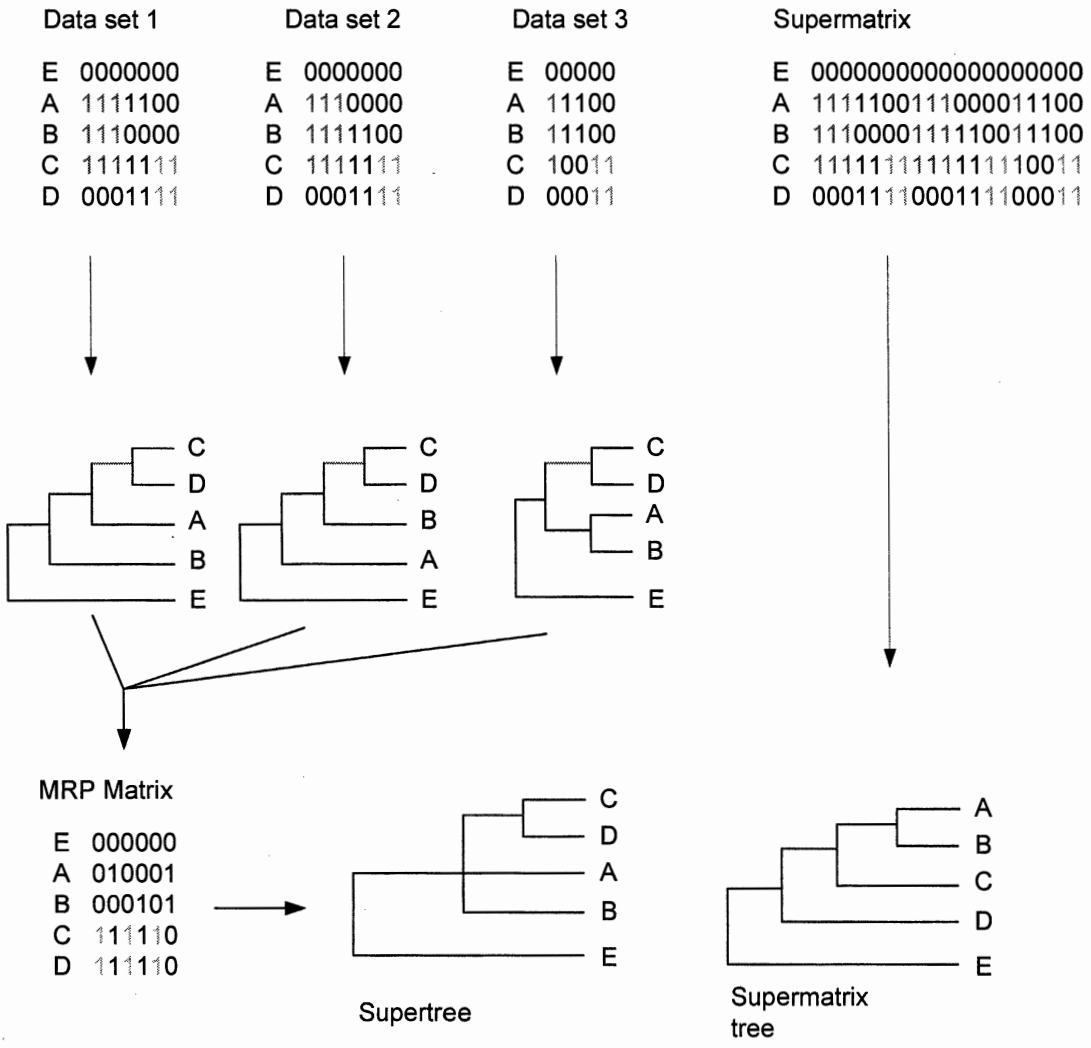


Figure 2.6 Schéma de l'approche MPR versus l'approche de la super-matrice la plus parcimonieuse appliquées à une collection de trois ensembles de données (reproduit d'après De Queiroz et Gatesy, 2007).

De nombreuses procédures ont été mises au point pour obtenir un super-arbre à partir d'un ensemble d'arbres phylogénétiques de plus petites tailles. La figure 2.6 indique

deux procédures aboutissant à deux super-arbres à partir du même ensemble de données. Le super-arbre de la partie gauche a été inféré par l'approche MPR (*c.-à-d.*, approche indirecte (Wilkinson, Hill et Gollan, 2001)) et celle de la partie droite a été obtenue via l'approche de la super-matrice (*c.-à-d.*, approche directe (Wilkinson, Hill et Gollan, 2001; McMahon et Sanderson, 2006)). Le super-arbre obtenu par l'approche MPR n'est pas entièrement résolu. Ceci est dû au fait que nous ne tenons pas compte de toutes les informations pour reconstruire le super-arbre, mais uniquement de l'information minimale permettant de discriminer chaque clade. Cette information discrimine les taxons de part et d'autre des branches internes des arbres phylogénétiques. La représentation matricielle, en codage binaire, constitue la base de la plupart des approches d'inférence de super-arbres. D'autres approches, qui se basent sur la notion de sous-arbres à quatre feuilles (*c.-à-d.*, des quartets), ont été proposées par l'équipe de Swenson *et al.* (Swenson *et al.*, 2011). Ces chercheurs ont montré qu'il existe des méthodes d'inférence de super-arbres qui améliorent les résultats de la méthode MPR.

La différence majeure entre l'arbre consensus et le super-arbre se situe en nombre de feuilles des arbres considérés en entrée de l'algorithme. Lorsque chaque arbre phylogénétique de la collection d'entrée contient le même ensemble de feuilles, alors on parle d'un arbre consensus. Par contre, si les ensembles d'espèces présentes dans des arbres d'entrée ne sont que partiellement chevauchants, alors on parle d'un super-arbre. Lors d'une analyse d'un ensemble de données génétiques réelles, c'est l'utilisation des méthodes de construction de super-arbres qui s'impose. Dans la pratique, il est très peu probable que tous les gènes considérés ont été séquencés pour les mêmes ensembles d'espèces.

L'inférence de super-arbres dans notre travail se fera à l'aide des méthodes implémentées dans le logiciel Clann (Creevey et McInerney, 2004). Notamment, nous utiliserons le logiciel Clann pour inférer les super-arbres issus des données réelles

(voir le chapitre VI pour plus de détails). Clann contient quatre méthodes de base, qui sont : 1) la représentation matricielle utilisant le principe de parcimonie (*c.-à-d.*, MPR ou *Most Parsimonious Reconstruction* (Ragan, 1992; Doyle, 1992)); 2) le super-arbre le plus similaire (*c.-à-d.*, MSSA ou *Most Similar Supertree* (Creevey *et al.*, 2004; Steel et Penny, 1993)); 3) le Quartet maximum le plus convenable (*c.-à-d.*, QFIT) (Creevey et McInerney, 2009); et 4) la scission maximum la plus convenable (*c.-à-d.*, SFIT) (Creevey et McInerney, 2004). L'approche MPR (voir la figure 2.6) consiste à coder les arbres sous forme matricielle. Les lignes de la matrice représentent l'ensemble des espèces et les colonnes de la matrice représentent l'ensemble des branches internes de l'arbre phylogénétique source. Chaque branche interne de l'arbre source permet de diviser (ou partitionner) les espèces en deux groupes (voir la figure 2.7). Le score de "1" est donné à une espèce si celle-ci est présente au sous-arbre T_1 et le score de "0" sinon (Eulenstein *et al.*, 2004).

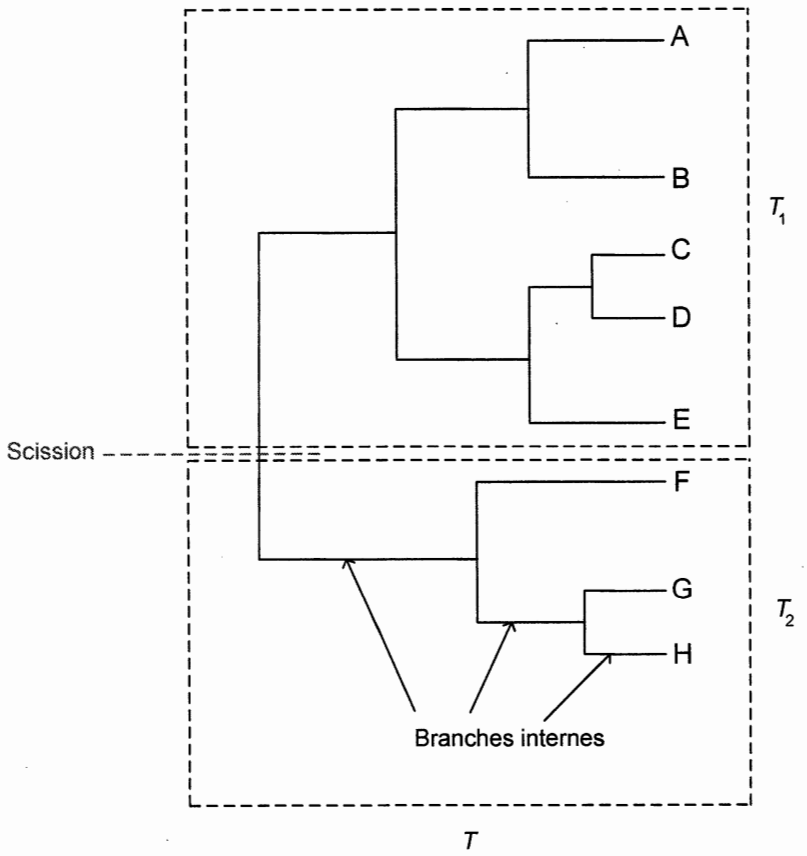


Figure 2.7 Illustration d'une scission d'une branche interne dans un arbre phylogénétique T aboutissant à deux sous-arbres phylogénétiques (T_1 et T_2).

L'approche MSSA compare chaque arbre phylogénétique source séparément du super-arbre en y confrontant la matrice des longueurs de branches correspondant à un arbre phylogénétique source à une autre matrice de distances correspondant au super-arbre élagué (Steel et Penny, 1993; Creevey et McInerney, 2004).

Dans les deux autres approches, QFIT et SFIT, chaque arbre phylogénétique source est individuellement comparé au super-arbre proposé en considérant tous les quartets (QFIT) ou toutes les scissions (SFIT, voir la figure 2.7), respectivement, de l'arbre phylogénétique source et du super-arbre élagué (Creevey et McInerney, 2004).

2.4 Reconstruction d'arbres consensus multiples – les méthodes existantes

Les méthodes classiques que nous avons présentées ci-dessus ne permettent d'inférer qu'un arbre consensus ou un super-arbre unique. Cependant, dans plusieurs situations pratiques, il est préférable de rechercher *des arbres consensus ou des super-arbres multiples* pour mieux caractériser l'évolution des familles de gènes données. Ces situations pratiques seront discutées ici, de même que, plus en détails, dans le chapitre III.

Les arbres phylogénétiques contiennent des informations importantes caractérisant l'évolution spécifique des familles de gènes étudiées. Cependant, un arbre consensus d'espèces fiable ne peut pas être inféré à partir d'un alignement de séquences multiples d'une famille de gènes unique ou de la concaténation des alignements correspondant à des familles de gènes ayant des histoires évolutives différentes. Ces histoires évolutives peuvent différer, par exemple, à cause des transferts horizontaux subis par certains gènes ou à cause de l'ancienne duplication génique qui provoque l'émergence de paralogues dans un génome. De nombreuses méthodes ont été proposées pour inférer un arbre consensus unique pour un ensemble d'arbres phylogénétiques donné. L'application de ces méthodes classiques peut donc conduire à la perte d'information sur les histoires évolutives spécifiques qui caractérisent certaines familles de gènes ou certains groupes de familles de gènes. Ainsi, la problématique d'inférence des arbres consensus multiples devient pertinente.

L'idée de reconstruire plusieurs arbres consensus a été initialement formulée par Maddison (Maddison, 1991), qui a trouvé que les arbres consensus de certains sous-ensembles d'arbres d'entrée peuvent différer et qu'ils sont généralement mieux résolus que l'arbre consensus unique de l'ensemble. Ensuite, Stockham *et al.* (Stockham, Wang et Warnow, 2002) ont proposé deux versions d'un algorithme de regroupement d'arbres, basées sur l'algorithme des *k*-moyennes, pour inférer un *ensemble d'arbres*

consensus stricts (appelés arbres caractéristiques) minimisant la perte d'information. Cependant, les deux versions des *k*-moyennes proposées par Stockham *et al.* restaient très coûteuses en termes de temps d'exécution car les arbres consensus multiples devaient être déterminés pour chaque ensemble de clusters dans toutes les solutions de partitionnement intermédiaires testées lors de l'exécution de l'algorithme des *k*-moyennes (voir aussi le chapitre III pour plus de détails). En outre, Bonnard *et al.* (Bonnard, Berry et Lartillot, 2006) ont décrit une méthode, appelée consensus multipolaire (ou *Multipolar Consensus*), pour afficher toutes les divisions (scissions ou *splits*) d'un ensemble d'arbres phylogénétiques donné ayant un support au-dessus d'un seuil prédéfini, en utilisant un nombre minimum possible d'arbres consensus. Les derniers auteurs ont indiqué que les signaux secondaires biologiquement pertinents, qui sont normalement ignorés par un arbre consensus classique, peuvent être capturés par la méthode du consensus multipolaire, fournissant ainsi un outil exploratoire pratique pour l'analyse phylogénétique. La méthode de Bonnard *et al.*, de même que celle de Stockham *et al.*, permet le plus souvent d'afficher plus de signaux évolutifs secondaires que la méthode classique de la règle majoritaire étendue, sans rendre possible les choix arbitraires qui sont habituellement faits dans cette méthode de consensus.

Dans son article récent, Guénoche (Guénoche, 2013) a présenté une méthode de partitionnement d'arbres phylogénétiques en un seul groupe, lorsque les arbres de gènes sont homogènes, ou en plusieurs groupes, lorsque les arbres de gènes sont divergents. Un score généralisé pour un partitionnement d'arbres est calculé par la méthode de Guénoche afin de déterminer le nombre optimal de classes d'arbres pour un ensemble d'arbres de gènes donné. Guénoche a validé sa méthode sur des données simulées, *c.-à-d.*, des ensembles aléatoires d'arbres organisés en groupes topologiques différents, ainsi que des données réelles, *c.-à-d.*, un ensemble d'arbres de gènes non homogènes de 30 souches de *E.coli* (ou *Escherichia coli*) supposées être affectées par des transferts horizontaux de gènes. Le programme MCT (*Multiple Consensus Trees*)

développé par l'auteur reste l'un des rares logiciels permettant d'inférer des arbres consensus multiples, qui est disponible gratuitement pour la communauté scientifique.

2.5 Conclusion

À travers ce chapitre, nous avons revu les approches fondamentales d'inférence d'arbres consensus et des super-arbres. Pour chacune des approches, nous avons mis en relief les problématiques sous-jacentes. Finalement, nous avons présenté les algorithmes existants pour la construction d'arbres consensus multiples.

Dans le prochain chapitre, nous allons ainsi proposer une première version de notre algorithme de classification d'arbres phylogénétiques en nous basant sur l'algorithme des *k*-médoïdes. Nous y mentionnerons les différents critères de validation de cluster étudiés.

CHAPITRE III

PARTITIONNEMENT D'ARBRES PHYLOGÉNÉTIQUES À L'AIDE DE LA MÉTHODE DES K-MÉDOÏDES

«Une technologie qui se limiterait à la classification des formes des outils et à l'analyse des états d'une fabrication entretiendrait vis-à-vis de l'ethnologie les mêmes rapports que la zoologie systématique vis-à-vis de la biologie animale.», André Leroi-Gourhan, 1965.

3.1 Préface

Dans ce chapitre, nous introduirons de nouveaux critères de partitionnement d'arbres phylogénétiques dans le cadre de l'algorithme des *k*-médoïdes (Kaufman et Rousseeuw, 1990) afin d'inférer les arbres consensus multiples. Dans un premier temps, nous adapterons l'algorithme des *k*-médoïdes au problème de partitionnement d'arbres phylogénétiques en considérant deux critères de validité des clusters populaires : celui de Caliński-Harabasz (*CH*) (Caliński et Harabasz, 1974) et celui de Silhouette (*SH*) (Rousseeuw, 1987). Nous testerons deux variantes de notre algorithme : 1) en utilisant la distance topologique de Robinson et Foulds (*RF*) (Robinson et Foulds, 1981), et 2) en utilisant la distance topologique de Robinson et Foulds mise au carré (comme suggéré par Stockham *et al.* (Stockham, Wang et Warnow, 2002)). Dans un deuxième temps, nous évaluerons la performance des deux critères sur des données simulées, ainsi que sur des données réelles (voir le chapitre

VI pour ces résultats). Enfin, nous comparerons la performance de chacune des deux versions des *k*-médoïdes avec une variante de l'algorithme de Stockham *et al.* (2002), qui permet de partitionner un ensemble d'arbres phylogénétiques en utilisant les *k*-moyennes, et sur le programme *Consense* du package PHYLIP de Felsenstein (2005), qui permet d'inférer les arbres consensus majoritaires.

3.2 Introduction

L'implémentation du projet « Tree of Life » (ToL) (Letunic et Bork, 2006; Letunic et Bork, 2016) visant la reconstruction de l'arbre phylogénétique du vivant a nécessité la collaboration de plusieurs équipes de recherche à travers le monde. L'approche adoptée par ToL consiste à réduire récursivement le problème de reconstruction de l'Arbre de Vie en plusieurs sous-problèmes, produisant des milliers d'arbres candidats, puis à fusionner les résultats. Ainsi, cette stratégie requiert l'inférence des arbres phylogénétiques consensus et des super-arbres. Plusieurs algorithmes ont été proposés pour construire un arbre consensus (Bryant, 2003) ou un super-arbre (Bininda-Emonds, Gittleman et Steel, 2002; Bininda-Emonds, 2004) d'un ensemble d'arbres phylogénétiques. Ces arbres phylogénétiques peuvent donc être définis sur un même ensemble d'espèces ou sur des ensembles d'espèces différents et partiellement chevauchants.

L'inférence des arbres consensus se base essentiellement sur trois algorithmes : l'algorithme du consensus strict, l'algorithme du consensus majoritaire et l'algorithme du consensus majoritaire étendu (voir le chapitre II, section 2.2.1). L'arbre consensus strict contient seulement les arêtes qui sont communes à tous les arbres. L'arbre consensus majoritaire contient les bipartitions qui sont présentes dans plus de 50% des arbres d'entrée, bien que d'autres pourcentages (> 50%) puissent aussi être considérés. Évidemment, si nous souhaitons considérer certaines bipartitions qui sont communes à moins de (ou exactement à) 50% des arbres d'entrées, nous pouvons

obtenir certaines incompatibilités. Dans ce cas, nous devons utiliser le principe du consensus majoritaire étendu. L'arbre consensus majoritaire étendu contient toutes les bipartitions majoritaires auxquelles sont ajoutées, à tour de rôle, les bipartitions compatibles restantes, en commençant par les bipartitions les plus fréquentes pour l'ensemble d'arbres donné. L'arbre consensus majoritaire étendu est le plus fréquemment utilisé en biologie et en bioinformatique car il est toujours le mieux résolu parmi les trois types d'arbres consensus mentionnés, ce qui veut dire que c'est toujours un arbre binaire. Dans notre étude, nous utiliserons l'arbre consensus majoritaire à cause de ses propriétés qui le relie à la distance topologique de Robinson et Foulds. La plus importante de ces propriétés étant que l'arbre consensus majoritaire est un *arbre médian* dans le sens de la distance de Robinson et Foulds (Barthélemy et McMorris, 1986).

L'*unicité de l'arbre consensus* fourni en sortie par les algorithmes d'inférence d'arbres consensus est l'élément commun à la plupart de ces algorithmes classiques. Cependant, dans plusieurs situations pratiques, il est préférable de prévoir une stratégie où plusieurs arbres consensus sont proposés comme solution. En biologie, il s'avère souvent hasardeux de regrouper des arbres phylogénétiques construits à partir de plusieurs gènes différents. Chaque gène traduit une histoire évolutive qui lui est propre et qui peut être différente des autres histoires. Par exemple, certains gènes peuvent subir des transferts horizontaux, qui sont souvent les mêmes pour un groupe de gènes, et leurs histoires évolutives seront représentées par des arbres phylogénétiques distincts des arbres de gènes non affectés par des transferts horizontaux. L'objet de cette thèse est de proposer une approche alternative d'inférence d'arbres consensus permettant d'obtenir un ou plusieurs arbres consensus en sortie de l'algorithme. La figure 3.1 illustre le cas de quatre arbres phylogénétiques T_1, T_2, T_3 et T_4 définis sur un même ensemble de sept feuilles. Dans cet exemple, la solution à deux arbres consensus majoritaires, T_{12} et T_{34}, est plus appropriée que celle de l'arbre consensus majoritaire unique, T_{1234}, proposée par l'approche classique.

Dans ce cas, nous observons que la solution à deux arbres consensus majoritaires minimise la perte d'information.

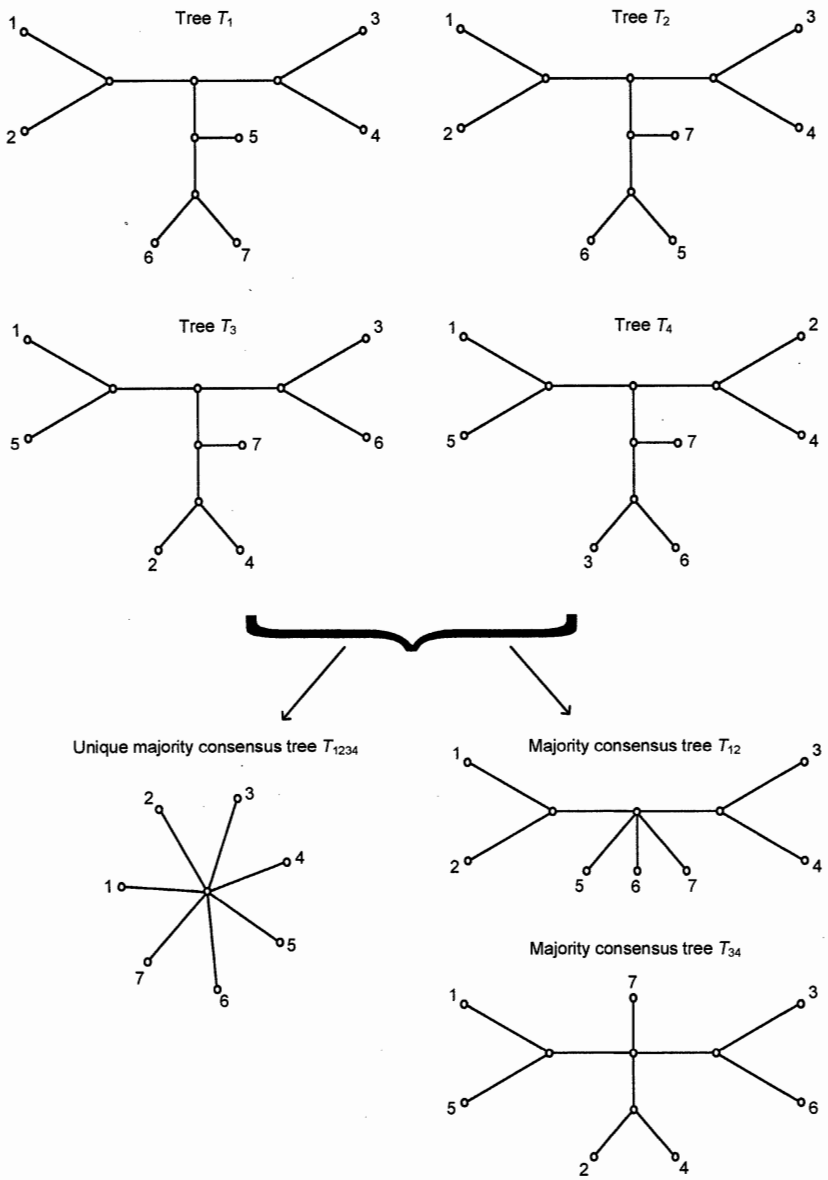


Figure 3.1 Quatre arbres phylogénétiques T_1, T_2, T_3 et T_4 définis sur un ensemble de sept feuilles; T_{1234} est l'arbre consensus majoritaire unique de T_1, T_2, T_3 et T_4; la solution à deux arbres consensus majoritaires T_{12} et T_{34} obtenus à partir des arbres (T_1, T_2) et (T_3, T_4), respectivement.

L'idée d'inférer plusieurs arbres consensus a été initialement formulée par Maddison (Maddison, 1991), qui a trouvé que les arbres consensus de certains sous-ensembles d'arbres sont souvent très différents et ont une meilleure résolution que l'arbre consensus de l'ensemble des arbres donnés. Plus récemment, Guénoche (Guénoche, 2013) a introduit une nouvelle méthode, MCT (*Multiple consensus trees*), pour déterminer si la représentation d'un ensemble d'arbres phylogénétiques doit se faire par un seul ou par plusieurs arbres consensus majoritaires. En 2002, l'équipe de Stockham *et al.* a proposé d'inférer l'arbre consensus strict pour chaque sous-ensemble du partitionnement en utilisant l'algorithme des *k*-moyennes afin de comparer ces arbres consensus avec l'arbre consensus strict de l'ensemble des données. Ces méthodes existantes sont discutées en détail dans la section 2.4.

3.3 Méthodologie

3.3.1 Partitionnement ou *clustering*

Le partitionnement est une approche de division d'un ensemble d'éléments (ou de taxons) en un ensemble significatif de groupes d'éléments appelés *clusters* (ou classes). Le but du partitionnement est de trouver des groupes d'éléments similaires selon une mesure de similarité donnée. À travers la littérature, nous énumérons quatre grandes approches de partitionnement pouvant servir à regrouper les données d'un ensemble d'éléments autour : 1) d'un centre de gravité, soit l'algorithme des *k*-moyennes (MacQueen, 1967), où *k* désigne le nombre désiré de clusters; 2) d'une médiane géométrique, soit *k*-médianes (Bradley, Mangasarian et Street, 1997); 3) d'un centre contenant les modes les plus fréquents, soit *k*-modes (Huang, 1998); 4) d'un médoïde qui minimise la somme des distances entre un élément spécifique et chacun des autres éléments de l'ensemble, soit *k*-médoïdes (Kaufman et Rousseeuw, 1990).

Le choix d'un de ces algorithmes dépend de plusieurs facteurs, tels que la nature des données sur lesquelles l'algorithme devra être appliqué, le résultat souhaité et le temps disponible pour l'obtention de ce résultat. Dans le cadre de ce projet doctoral, nous nous concentrerons sur l'algorithme des *k*-médoïdes dans un premier temps (voir le chapitre III), puis sur l'algorithme des *k*-moyennes dans un second temps (voir les chapitres IV et V).

3.3.2 L'algorithme des *k*-médoïdes

L'algorithme des *k*-médoïdes (Kauffman et Rousseeuw, 1990) est un algorithme de partitionnement des données tout comme l'algorithme des *k*-moyennes (voir le chapitre V - section 4.3.2). Les *k*-médoïdes divisent automatiquement *n* éléments en *K* groupes en passant par des centres de regroupement (*c.-à-d.*, les médoïdes). L'algorithme sélectionne un élément comme le centre d'un regroupement. Cet élément, pris parmi l'ensemble des données initiales, sera choisi pour minimiser la distance intragroupe. L'algorithme des *k*-médoïdes est comme suit :

Algorithme 3.1 des *k*-médoïdes (Kauffman et Rousseeuw, 1990)

Étape 1. Initialisation – Choisir, comme médoïdes, K éléments pris au hasard parmi les N éléments donnés.

Étape 2. Assigner chaque élément donné au médoïde le plus proche.

Tant que le score de la fonction objective $\sum_{k=1}^{K}\sum_{x\in C_k}\left\|x-m_k\right\|^2$ (voir ci-dessous pour plus détails) diminue et le nombre maximum d'itérations, I, n'a pas été atteint **faire**

Pour chaque médoïde m **faire** :

Pour chaque non médoïde o **faire** :

Étape 3. Échanger m et o, recalculer le coût de la fonction objective pour cette nouvelle configuration.

Si le coût de la fonction objective augmente **alors** :

Étape 4. Retourner à la configuration précédente.

Fin

La formule $\sum_{k=1}^{K}\sum_{x\in C_k}\left\|x-m_k\right\|^2$ représente la somme au carré du total des distances intragroupes, où K est le nombre de clusters, x est un élément des données, C_k est le cluster k et m_k est le médoïde du cluster C_k.

Le nombre maximum d'itérations, I, est fréquemment égal à 100. L'algorithme est généralement exécuté plusieurs fois (généralement 100, 500 ou 1000) avec les partitions de départ aléatoires. Le meilleur partitionnement est ensuite sélectionné selon le score de l'indice de validité des clusters choisi, *p.ex.* l'indice de Caliński-Harabasz (voir la section 3.3.4) ou l'indice Silhouette (voir la section 3.3.5).

La complexité algorithmique des *k*-médoïdes est $O(I\times K\times M\times(N-K)^2)$ où K est le nombre de clusters; N est le nombre d'éléments, I est le nombre maximum d'itérations

dans la boucle des *k*-médoïdes et *M* est le nombre de variables caractérisant les éléments de l'ensemble.

Notons que, contrairement à la méthode des *k*-moyennes, l'algorithme des *k*-médoïdes n'est pas sensible aux valeurs aberrantes (*outliers*). De plus, cet algorithme est certes plus robuste, mais plus complexe à implémenter que l'algorithme des *k*-moyennes (MacQueen, 1967; Kaur, Kaur et Singh, 2014).

L'approche PAM (*Partitioning Around Medoids*) fut le premier algorithme implémenté par Kaufman et Rousseeuw en 1990 en utilisant le principe des *k*-médoïdes. Un des désavantages majeurs de cet algorithme est qu'il n'est pas applicable à des données volumineuses (*Big Data*) (Jiang et Zhang, 2014). La complexité algorithmique de PAM est de $O(I \times M \times K \times (N - K)^2)$, ce qui est supérieur à la complexité des *k*-moyennes. Pour des données volumineuses, *N* est généralement plus grand que *K* et *I*, entraînant ainsi une diminution de l'efficacité algorithmique. Plusieurs variantes de cet algorithme ont été ensuite proposées afin de réduire sa complexité (Jiang et Zhang, 2014).

Pour traiter le cas des ensembles de données volumineuses, Kaufman et Rousseeuw (Kaufman et Rousseeuw, 1990) ont développé l'algorithme CLARA (*Clustering LARAge Application*). CLARA sélectionne aléatoirement 5 échantillons de taille $40 + 2K$ en utilisant la stratégie de PAM pour satisfaire les résultats (Zhang et Couloigner, 2005). L'efficacité de CLARA dépend donc de la méthode d'échantillonnage utilisée ainsi que de la taille des échantillons. CLARA ne peut pas toujours trouver le meilleur partitionnement (Zhang et Couloigner, 2005). Il est donc difficile de déterminer la taille optimale de l'échantillon. Une version optimisée de CLARA a été proposée en 2002 par Ng et Han (Ng et Han, 2002). Il s'agit de CLARANS. Cette nouvelle variante de l'algorithme réduit de la taille des échantillons en randomisant la recherche des médoïdes (Ng et Han, 2002). La complexité algorithmique de CLARANS est de $O(N^2 \times M)$.

L'algorithme rapide des *k*-médoïdes, qui se base sur le modèle de traitement en parallèle et sur l'augmentation des nœuds de calcul pour diminuer linéairement la complexité algorithmique, a été récemment introduit. Notamment, en 2014, Jiang et Zhang proposent les HK-médoïdes basés sur *Hadoop* dont une partie du traitement utilise la stratégie de *MapReduce* (Jiang et Zhang, 2014). Il s'agit d'un patron d'architecture proposé par le groupe Google dont les traitements des données s'effectuent en parallèle ou en distribué. Ce système a un code source ouvert (*open source*) et peut être facilement incorporé dans des projets en empruntant les approches de *map* et de réduction (*MapReduce*).

3.3.3 La fonction objective des *k*-médoïdes pour le cas des arbres phylogénétiques

L'algorithme des *k*-médoïdes partitionne un ensemble d'objets donnés en K (avec $K \geq 2$) classes en utilisant une fonction objective (*p.ex.* basée sur la distance euclidienne ou sur la distance de Minkowski) et l'indice de validité des clusters choisi (*p.ex.* Caliński-Harabasz (Caliński et Harabasz, 1974), Silhouette (Rousseeuw, 1987), Ball et Hall (Ball-Hall, 1967) ou Dunn (Dunn, 1974)). La plupart des indices de validité des clusters traditionnels prennent en compte à la fois des évaluations intra et extragroupes. Cependant, nous ne pouvons pas utiliser la version de la fonction objective standard, ni même des indices de validité des clusters standards, des *k*-médoïdes lors du partitionnement d'un ensemble d'arbres phylogénétiques. En effet, ces critères ont été conçus pour fonctionner dans l'espace euclidien. Dans ce projet doctoral, nous proposerons des changements à apporter à l'algorithme des *k*-médoïdes afin de l'adapter à la classification d'arbres.

La fonction objective dans le cas de partitionnement d'arbres phylogénétiques, basée sur les propriétés des *k*-médoïdes, peut être définie comme suit :

$$FO_{med} = \sum_{k=1}^{K} \sum_{i=1}^{N_k} RF(T_k^m, T_{ki}), \tag{3.1}$$

où K est le nombre de groupes (classes ou clusters), N_k est le nombre d'arbres phylogénétiques dans le groupe k, RF est la distance topologique de Robinson et Foulds entre deux arbres phylogénétiques avec n feuilles (Robinson et Foulds, 1981), T_{ki} est l'arbre phylogénétique i du groupe k et T_k^m est l'arbre médoïde du groupe k. L'arbre médoïde T_k^m du groupe k est un arbre appartenant à ce groupe, tel que la somme des distances RF entre cet arbre et tous les autres arbres du groupe k soit minimale.

3.3.4 Le critère de validité des clusters : Caliński-Harabasz

Le premier critère de validité des clusters que nous considérons est le critère de Caliński-Harabasz (*CH*). Ce critère, parfois appelé le critère de ratio de variance, est défini comme suit :

$$CH(K) = \frac{SS_B}{SS_W} \times \frac{N-K}{K-1}, \tag{3.2}$$

où SS_B est l'indice d'évaluation intergroupe et SS_W est l'indice d'évaluation intragroupe. Ce critère peut être utilisé pour les valeurs de K telles que : $2 \leq K \leq N-1$ où K est le nombre de clusters et N est le nombre d'objets (arbres phylogénétiques dans notre cas). Le nombre optimal de clusters correspond à la valeur maximale de *CH*.

Dans la version originale de *CH*, le coefficient SS_B (voir l'équation 3.3) est évalué à l'aide de la norme L^2, soit la distance euclidienne.

$$SS_B = \sum_{k=1}^{K} N_k \left\| m_k - m \right\|^2, \tag{3.3}$$

où m_k (k =1 ... K) est le médoïde du cluster k, m est le médoïde de l'ensemble des données et N_k est le nombre d'éléments dans le cluster k. Dans le cas du partitionnement d'arbres phylogénétiques, l'indice SS_B utilisera la distance topologique de Robinson et Foulds en s'appuyant sur la Propriété 3.1 de Barthélemy et McMorris (Barthélemy et McMorris, 1986).

Propriété 3.1 : L'*arbre consensus majoritaire* d'un groupe d'arbres phylogénétiques définis sur un même ensemble de feuilles est un *arbre médian* de ce groupe d'arbres dans le sens de la distance topologique de Robinson et Foulds (Barthélemy et McMorris, 1986).

L'indice SS_B sera formulé comme indiqué dans l'équation suivante :

$$SS_B = \sum_{k=1}^{K} N_k \times RF(T^m, T_k^m), \tag{3.4}$$

où N_k est le nombre d'arbres phylogénétiques dans le groupe k, T_k^m est l'arbre médoïde du groupe k et T^m est l'arbre médoïde de l'ensemble des données.

Le coefficient SS_W (voir l'équation 3.5) est classiquement calculé de la manière suivante :

$$SS_W = \sum_{k=1}^{K} \sum_{i=1}^{N_k} \left\| x_{ki} - m_k \right\|^2, \tag{3.5}$$

où x_{ki} est l'objet i du cluster k.

Tout comme dans le cas de l'indice SS_B, nous avons adapté au cas de partitionnement d'arbres l'équation 3.5 de l'indice SS_W, en passant par la distance RF :

$$SS_W = \sum_{k=1}^{K} \sum_{i=1}^{N_k} RF(T_k^m, T_{ki}), \tag{3.6}$$

où T_{ki} est l'arbre i du cluster k.

Une bonne solution d'un algorithme de partitionnement correspond à un écart important entre les clusters et en même temps à une petite variance au sein des clusters. Plus le ratio de l'indice *CH* est grand, meilleur est le partitionnement des données.

3.3.5 Le critère de validité des clusters : Silhouette

L'indice Silhouette (Rousseeuw, 1987) est un autre indice de classification populaire permettant d'évaluer la qualité d'un partitionnement donné.

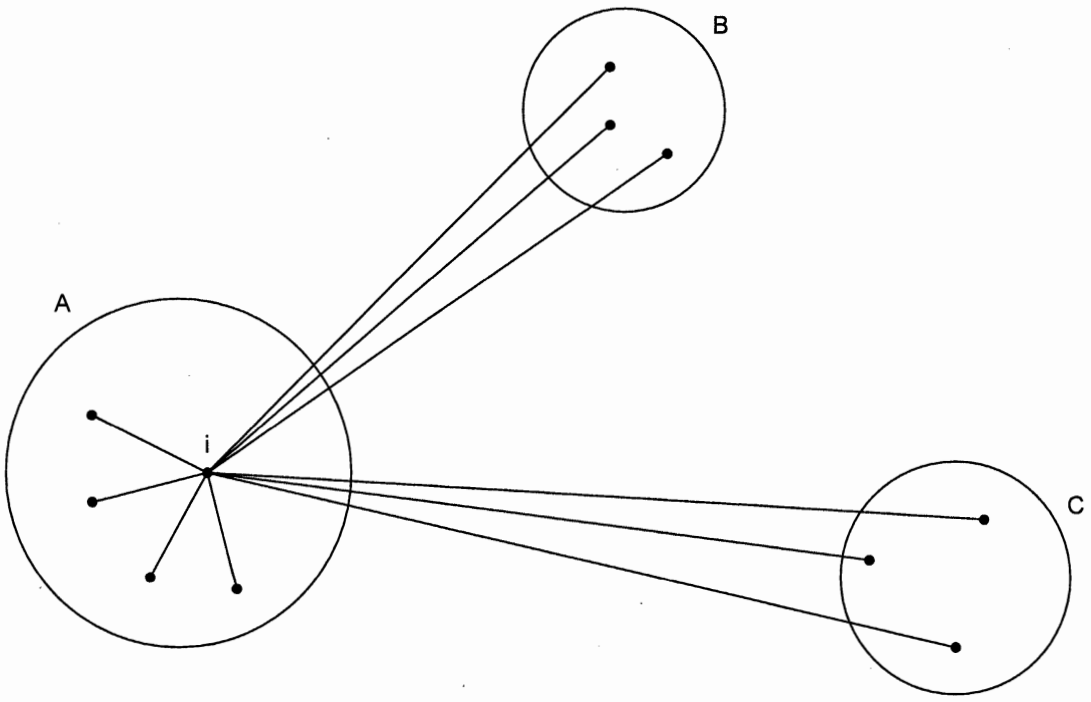


Figure 3.2 Illustration des éléments impliqués dans le calcul de $s(i)$, où l'objet i appartient au cluster A (reproduit d'après Rousseeuw, 1987).

Pour chaque objet i, $a(i)$ est la moyenne des distances de i à tous les objets du même cluster, soit le cluster A de la figure 3.2. La distance $d(i,C)$ est la moyenne des distances entre l'objet i et tous les objets du cluster C, tels que C est différent de A. Aussi, la distance $b(i)$ représente le minimum des distances $d(i,C)$.

La mesure Silhouette, $s(k)$, du cluster k est définie comme suit :

$$s(k) = \left[\sum_{i=1}^{N_k} \frac{b(i)-a(i)}{\max(a(i),b(i))} \right] \Big/ N_k . \tag{3.7}$$

En adaptant l'équation 3.7 aux cas du partitionnement des arbres, nous obtenons les équations 3.8 et 3.9 suivantes, représentant respectivement $a(i)$ et $b(i)$:

$$a(i) = \frac{\sum_{j=1}^{N_k} RF(T_{ki}, T_{kj})}{N_k}, \tag{3.8}$$

où N_k est le nombre d'arbres phylogénétiques dans la classe k, T_{ki} est l'arbre i de la classe k, T_{kj} est l'arbre j de la classe k et $RF(T_{ki}, T_{kj})$ est la distance topologique de Robinson et Foulds (Robinson et Foulds, 1981) entre l'arbre T_{ki} et l'arbre T_{kj}.

Et :

$$b(i) = \min_{1 \le k' \le K \text{ et } k' \ne k} \frac{\sum_{j=1}^{N_{k'}} RF(T_{ki}, T_{k'j})}{N_{k'}}, \tag{3.9}$$

où $N_{k'}$ est le nombre d'arbres phylogénétiques dans la classe k' et $T_{k'j}$ est l'arbre j de la classe k'.

La mesure Silhouette globale d'un partitionnement en K classes donné, $\bar{s}(k)$, est la moyenne de toutes les mesures des clusters individuelles $s(k)$. Elle est calculée comme suit :

$$\bar{s}(K) = \frac{\sum_{k=1}^{K} s(k)}{K}. \qquad (3.10)$$

3.3.6 L'algorithme des *k*-médoïdes adapté au partitionnement d'arbres phylogénétiques

Nous présentons ici la version de l'algorithme des *k*-médoïdes adaptée au partitionnement d'arbres phylogénétiques (voir l'Algorithme 3.2). Dans notre algorithme, le médoïde correspond à un arbre phylogénétique tel que la somme des distances *RF* entre cet arbre et tous les autres arbres du groupe *k* soit minimale.

Algorithme 3.2 des *k*-médoïdes adapté aux arbres phylogénétiques

---

Étape 1. Initialisation – Choisir, comme médoïdes, *K* arbres pris au hasard parmi les *N* arbres phylogénétiques donnés.

Étape 2. Assigner chaque arbre donné au médoïde le plus proche dans le sens de la distance topologique de Robinson et Foulds.

Tant que le score de la fonction objective $FO_{med} = \sum_{k=1}^{K} \sum_{i=1}^{N_k} RF(T_k^m, T_{ki})$ diminue ou le nombre maximum d'itérations, *I*, n'a pas été atteint **faire** :

Pour chaque arbre médoïde *m* **faire** :

Pour chaque arbre non médoïde *o* **faire** :

Étape 3. Échanger *m* et *o*, recalculer le coût de la fonction objective pour cette nouvelle configuration.

Si le coût de la fonction objective augmente **alors** :

Étape 4. Retourner à la configuration précédente.

Fin

La complexité algorithmique des *k*-médoïdes adapté au partitionnement d'arbres phylogénétiques est $O(I \times K \times (N-K)^2 + nN^2)$, où $O(nN^2)$ est la complexité temporelle du précalcul de la matrice des distances *RF* de taille $(N \times N)$ entre les paires d'arbres phylogénétiques donnés, K est le nombre de clusters, N est le nombre d'arbres phylogénétiques donnés, I est le nombre d'itérations maximum dans la boucle principale des *k*-médoïdes et n est le nombre d'espèces dans chaque arbre phylogénétique donné.

3.3.7 L'indice Rand ajusté

La qualité de nos résultats a été évaluée à l'aide de l'indice Rand ajusté (*ARI*). L'indice Rand classique (Rand, 1971) est très sensible au nombre de clusters. C'est pourquoi, une version ajustée de l'indice Rand a été mise en place (Hubert et Arabie, 1985; Steinley, Brusco et Hubert, 2016). L'*ARI* a une espérance nulle si les partitionnements comparés sont sélectionnés au hasard. Les valeurs d'*ARI* sont situées dans l'intervalle [-1; 1]. Lorsque deux partitionnements sont exactement les mêmes, la valeur correspondante d'*ARI* est de 1. L'*ARI* est souvent utilisé dans les simulations pour comparer les clusters originaux connus avec celles générés par les méthodes étudiées. Étant donné un ensemble de n objets et de deux partitionnements de ces objets, à savoir $X = X_1, X_2, ..., X_r$ avec r clusters et $Y = Y_1, Y_2, ..., Y_s$ avec s clusters, le chevauchement entre X et Y peut être résumé en utilisant une matrice de contingence $[n_{ij}]$, où chaque entrée n_{ij} indique le nombre d'objets en commun entre les clusters X_i et Y_j.

Tableau 3.1 La table des contingences pour comparer deux partitionnements $X = X_1, X_2, ..., X_r$ avec r clusters et $Y = Y_1, Y_2, ..., Y_s$ avec s clusters (reproduit d'après Yeung et Ruzzo, 2001).

Clusters	Y_1	Y_2	...	Y_s	Sums
X_1	n_{11}	n_{12}	...	n_{1s}	a_1
X_2	n_{21}	n_{22}	...	n_{2s}	a_2
⋮	⋮	⋮		⋮	⋮
X_r	n_{r1}	n_{r2}	...	n_{rs}	a_r
Sums	b_1	b_2	...	b_s	

L'*ARI* est évalué comme suit :

$$ARI = \frac{\text{indice} - \text{indice attendu}}{\text{indice maximum} - \text{indice attendu}}. \tag{3.12}$$

Plus spécifiquement :

$$ARI = \frac{\sum_{ij}\binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}\right] \Big/ \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \left[\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}\right] \Big/ \binom{n}{2}}, \tag{3.13}$$

où $n_{ij} = \left|X_i \cap Y_j\right|$, $a_i = \sum_{j=1}^{s}\left|X_i \cap Y_j\right|$, $b_j = \sum_{j=1}^{r}\left|X_i \cap Y_j\right|$ et X_i et Y_j sont les ensembles d'objets dans les clusters i et j, respectivement. La fonction de l'*ARI* a été écrite en C++, voir l'Appendice B, la section B.4.

3.4 Génération des données pour les simulations

Nous avons réalisé quelques simulations avec des données synthétiques, en plus des données réelles (voir le chapitre VI), afin de valider notre algorithme. Dans cette section, nous décrivons les ensembles de données synthétiques examinées dans notre étude, ainsi que la procédure de génération de ces données.

Nous avons d'abord testé notre nouvel algorithme de partitionnement d'arbres phylogénétiques (voir la section 3.3.6) basé les *k*-médoïdes (voir la section 3.3.2) avec des données simulées. Le protocole de nos simulations inclut deux étapes. La première étape consiste à générer aléatoirement K arbres phylogénétiques binaires, T_1, ..., T_K, avec n feuilles chaque et K clusters *a priori*, où $K = 1, ..., 10$ et $n = 8, 16, 32, 64$. Dans la deuxième étape, pour chaque arbre phylogénétique T_i (où $i = 1 ... K$) obtenu à la première étape, nous avons généré aléatoirement un ensemble de 100 arbres correspondant à la classe i, pour chacun des quatre intervalles de bruit définis ci-dessous. Chaque élément T de la classe i était un arbre phylogénétique, tel que les pourcentages de similarités, mesurés à l'aide de la distance *RF*, entre T et T_i variaient de 0% à 10%, de 10% à 25%, de 25% à 50%, ou de 50% à 75%. La distance *RF* maximale mesurée entre deux arbres phylogénétiques binaires à n feuilles est égale à $2n - 6$. Nous avons normalisé toutes les distances *RF* obtenues par ce facteur. De ce fait, la distance *RF* normalisée était comprise entre 0 et 1, inclusivement. Notons également que si la distance *RF* normalisée vaut 1, alors les deux arbres comparés sont totalement différents et, inversement, lorsque cette distance vaut 0, cela indique que les deux arbres sont identiques. Nous avons multiplié par 100% la distance *RF* normalisée pour avoir sa correspondance en termes de pourcentage d'éloignement. Nos intervalles correspondent donc au niveau de bruit (10%, 25%, 50% et 75%).

3.5 Résultats des simulations

Dans cette section, nous présentons les résultats de nos premières simulations effectuées avec notre nouvel algorithme des *k*-médoïdes pour les arbres. Dans nos simulations, nous avons testé différentes versions de notre algorithme de partitionnement d'arbres phylogénétiques, soit celles basées sur les critères de validation des clusters de Caliński-Harabasz (*CH*) et de Silhouette (*SH*), et celles utilisant la fonction objective (voir la formule 3.1) dont la distance de Robinson et Foulds (*RF*) a été mise au carré ou non mise au carré. Deux types d'expériences ont été réalisées : (i) lorsque le nombre de clusters était connu *a priori* et (ii) quand ce nombre était inconnu (soit lors des expériences avec les données réelles, voir le chapitre VI). La qualité des partitionnements a été évaluée en utilisant l'indice Rand ajusté (*ARI*).

Les résultats de nos simulations sont illustrés sur les figures 3.3, 3.4 et 3.5. Les résultats présentés sont les moyennes prises sur toutes les combinaisons de nos paramètres (voir la section 3.4); 100 différents ensembles de données ont été testés pour chaque combinaison de paramètres.

La figure 3.3 montre une augmentation des valeurs d'*ARI* avec l'augmentation du nombre de clusters, *K*. Tous les critères sont moins performants lorsque $K = 2$ avec la moyenne de l'*ARI* variant de 0.7566 et 0.6586 pour les critères *SH* et *CH*, respectivement. La performance de l'algorithme développé se stabilise pour de plus grandes valeurs de *K* pour atteindre 0.9627 et 0.8467 pour *SH* et *CH*, respectivement. Nous constatons par ailleurs que les variantes de l'algorithme sans la mise au carré de la distance *RF* sont plus performantes, en termes d'*ARI* moyen, que celles avec la mise au carré de cette distance. De plus, nous observons que le critère *SH* est plus performant que le critère *CH*. L'algorithme atteint une certaine stabilité des résultats pour des données ayant 5 clusters ou plus, avec cependant une légère baisse pour le cas de 10 clusters.

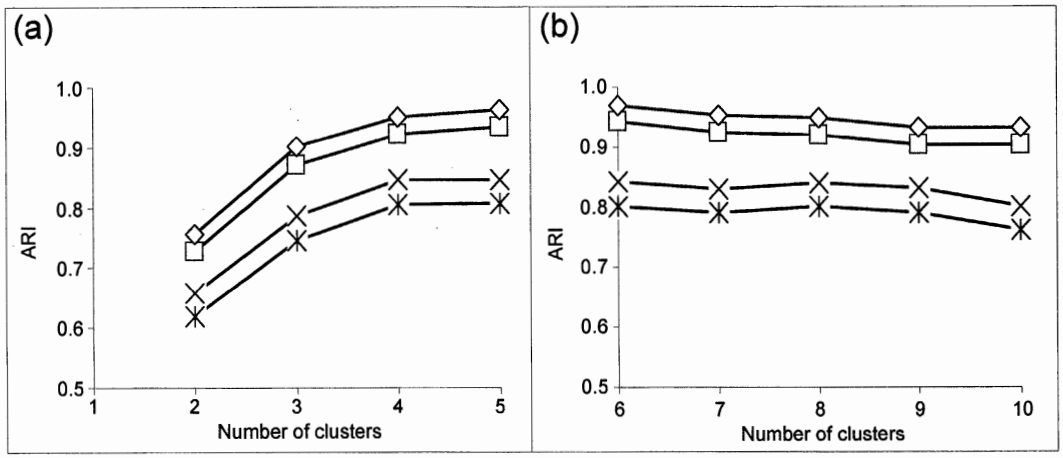


Figure 3.3 Les performances des quatre versions de notre algorithme des *k*-médoïdes pour le partitionnement d'arbres phylogénétiques, mesurées à l'aide de l'indice Rand ajusté (*ARI*), en fonction du nombre de clusters K : (a) pour K allant de 2 à 5 et (b) pour K allant de 6 à 10. Nous avons utilisé deux critères de validité des clusters *CH* et *SH* : 1) SANS la mise au carré de la distance topologique *RF* pour *SH* (◊) et pour *CH* (×) et 2) AVEC la mise au carré de la distance topologique *RF* pour *SH* (□) et pour *CH* (✳).

La figure 3.4 illustre les résultats obtenus en fonction du nombre de feuilles. Nous avons traité des arbres phylogénétiques ayant 8, 16, 32 et 64 feuilles. Tous les quatre critères considérés montrent une bonne performance pour les arbres à 64 feuilles. En effet, les résultats varient beaucoup en fonction du nombre de feuilles. L'*ARI* moyen est de 0.8075 pour le critère *SH* et de 0.7025 pour le critère *CH* lorsque le nombre de feuilles est égal à 8, pour atteindre 0.9424 pour *SH* et 0.8358 pour *CH* lorsque le nombre de feuilles est égal à 64.

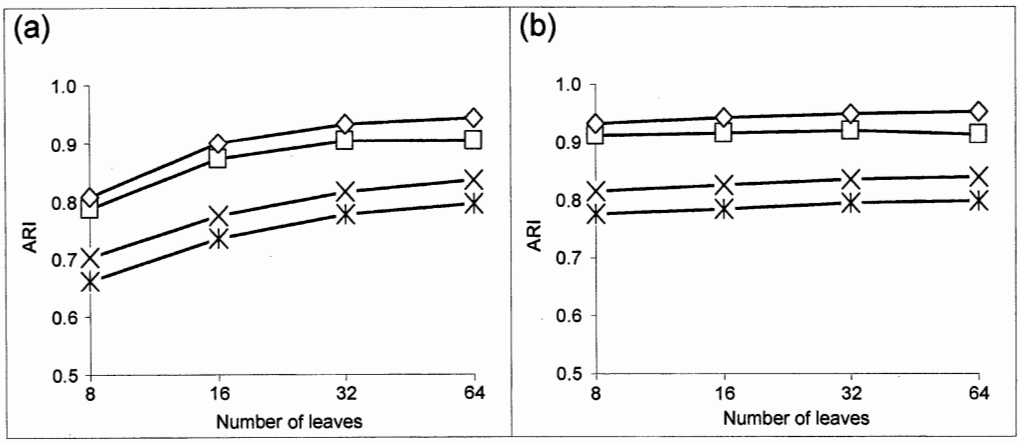


Figure 3.4 Les performances des quatre versions de notre algorithme des *k*-médoïdes pour le partitionnement d'arbres phylogénétiques, mesurées à l'aide de l'indice Rand ajusté (*ARI*), en fonction du nombre de feuilles dans les arbres phylogénétiques considérés : (a) pour (le nombre de clusters) K allant de 2 à 5 et (b) pour K allant de 6 à 10. Nous avons utilisé deux critères de validité des clusters *CH* et *SH* : 1) SANS la mise au carré de la distance topologique *RF* pour *SH* (◊) et pour *CH* (×) et 2) AVEC la mise au carré de la distance topologique *RF* pour *SH* (□) et pour *CH* (✳).

Enfin, nous avons évalué les performances de notre algorithme en fonction de l'hétérogénéité des données (voir la figure 3.5). Pour cela, nous avons ajouté le bruit dans les arbres phylogénétiques générés (voir la section 3.4). Les performances de partitionnement d'arbres sont presque optimales pour les données homogènes (soit 0% de bruit) et décroissent drastiquement avec 75% de bruit pour atteindre 0.5646 pour le critère *CH* et de 0.77 pour le critère *SH*, pour les données ayant de 2 à 5 clusters (voir la figure 3.5a). Sur la figure 3.5(b), représentant les données ayant de 6 à 10 clusters, nous observons une légère amélioration des résultats. Avec 75% de

bruit et avec les données ayant de 5 à 10 clusters, l'*ARI* moyen est de 0.8797 pour le critère *SH* et de 0.6646 pour le critère *CH*.

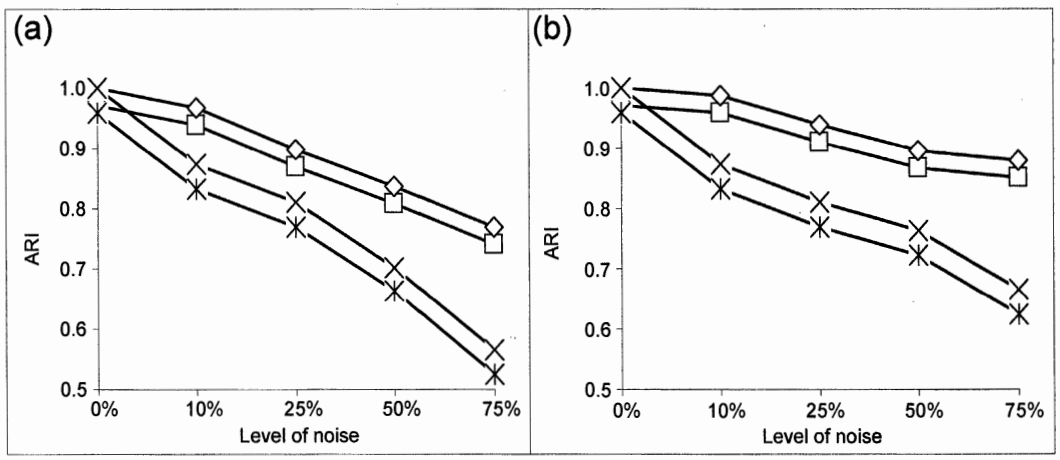


Figure 3.5 Les performances des quatre versions de notre algorithme des *k*-médoïdes pour le partitionnement d'arbres phylogénétiques, mesurées à l'aide de l'indice Rand ajusté (*ARI*), en fonction du niveau de bruit : (a) pour *K* (nombre de clusters) allant de 2 à 5 et (b) pour *K* allant de 6 à 10. Pour chaque cas, nous avons utilisé deux variantes des deux critères *CH* et *SH* : 1) SANS la mise au carré de la distance topologique *RF* pour *SH* (◊) et pour *CH* (×) et 2) AVEC la mise au carré de la distance topologique *RF* pour *SH* (□) et pour *CH* (✳).

Dans notre deuxième simulation, nous avons comparé notre nouvel algorithme, prenant sa version basée sur l'indice de validité des clusters (*SH*) et la distance topologique *RF* non mise au carré, à l'algorithme de partitionnement d'arbres de Stockham *et al.* (2002) basé sur les *k*-moyennes. Ce dernier algorithme procède par l'inférence d'arbres consensus multiples, représentant les centroïdes des clusters, à chaque opération de base des *k*-moyennes (voir Stockham *et al.*, 2002). Même si Stockham *et al.* ont présenté une version de leur algorithme permettant d'obtenir les arbres consensus stricts, nous l'avons adapté à l'inférence des arbres consensus majoritaires. Le programme *Consense* de Felsenstein (1985) a été utilisé pour obtenir

les arbres consensus majoritaires à chaque étape de base des *k*-moyennes. Notre comparaison a été faite en termes de qualité des partitionnements obtenus (voir les figures 3.6a et 3.6b) et du temps d'exécution des deux algorithmes (voir les figures 3.6c et 3.6d).

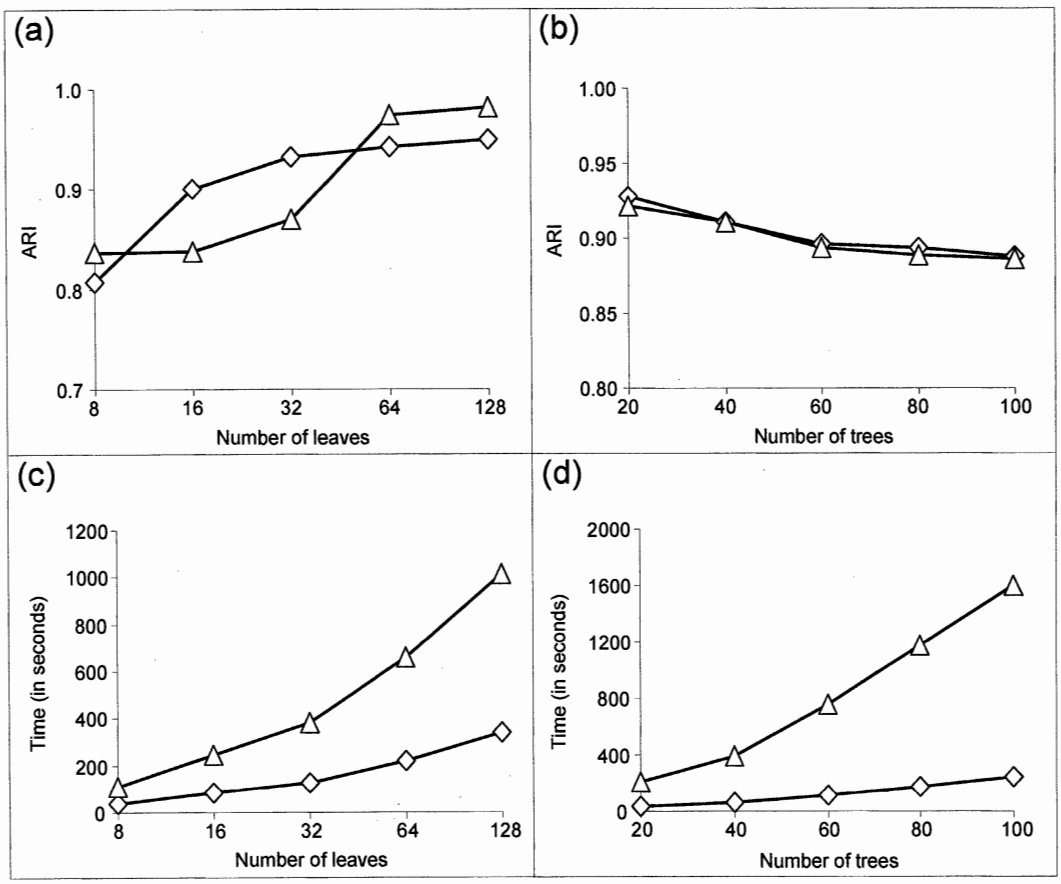


Figure 3.6 Comparaison de notre algorithme des *k*-médoïdes pour le partitionnement d'arbres phylogénétiques (basé sur l'indice de validité des clusters *SH* et la distance *RF* non mise au carré (◊)) à l'algorithme de partitionnement d'arbres de Stockham *et al.* (2002) (basé sur les *k*-moyennes (Δ)). Cette comparaison a été réalisée en fonction de la qualité des partitionnements obtenus (a et b) et de la complexité temporelle (c et d). Les paramètres utilisés dans notre simulations étaient le nombre de feuilles (a et c) et le nombre d'arbres phylogénétiques (b et d).

Nos tests ont été effectués en utilisant un ordinateur de 64-bit, équipé avec un processeur Intel i5-4690T CPU (2.5 GHz) et 8 Gb de RAM (ou mémoire vive). Le nombre de feuilles des arbres variait de 8 à 128 et le niveau de bruit variait entre 0% à 75% dans cette simulation. Les autres paramètres étaient ceux présentés à la section 3.4. Une fois de plus, les 100 jeux de données différents ont été générés pour chaque combinaison de paramètres. En examinant les résultats de cette deuxième simulation, nous pouvons constater que les performances de notre algorithme, mesurées en termes d'*ARI* moyen, sont sensiblement identiques à celles de l'approche de Stockham *et al.*, nécessitant le recalcul des arbres consensus à chaque itération de base de l'algorithme, voir les figures 3.6a et 3.6b. Cependant, en termes de complexité temporelle, notre nouvel algorithme s'est avéré beaucoup plus performant que l'approche de Stockham *et al.* pour les deux paramètres de cette simulation, à savoir, le nombre de feuilles et le nombre d'arbres (voir respectivement les figures 3.6c et 3.6d).

3.6 Conclusion

Dans ce chapitre, nous avons présenté un nouvel algorithme de partitionnement d'arbres phylogénétiques basé sur la méthode des *k*-médoïdes. La distance topologique de Robinson et Foulds a été utilisée dans notre approche pour comparer la similarité de deux arbres phylogénétiques. Deux critères de validité des clusters, celui de Caliński-Harabasz et celui de Silhouette, ont été adaptés au cas de partitionnement d'arbres phylogénétiques. Notre algorithme de partitionnement d'arbres a une complexité algorithmique de $O(nN^2+K\times(N-K)^2\times I)$, où $O(nN^2)$ est la complexité temporelle du précalcul de la matrice des distances de Robinson et Foulds de taille $(N\times N)$ entre les paires d'arbres phylogénétiques donnés, K est le nombre de clusters, N est le nombre d'arbres phylogénétiques donnés, I est le nombre d'itérations maximum dans la boucle principale des *k*-médoïdes et n est le nombre d'espèces dans

chaque arbre phylogénétique donné. Notre algorithme est beaucoup plus rapide que l'approche de partitionnement d'arbres de Stockham *et al.* (2002) basée sur les *k*-moyennes, tout en fournissant des performances de partitionnement semblables. Les bonnes performances du nouvel algorithme en termes de qualité de partitionnement et du temps de calcul le rendent bien adapté à l'analyse de grands jeux de données génomiques et phylogénétiques. Un programme C++, appelé KMTC (*K-Medoids Tree Clustering*), a été développé et mis à la disponibilité de la communauté scientifique à l'adresse URL suivante : https://github.com/TahiriNadia/CKMedoidsTreeClustering. Nous pouvons constater que le cas des données homogènes (où le nombre de clusters $K = 1$) n'était pas couvert par notre algorithme. Dans les chapitres suivants nous présenterons de nouveaux critères permettant de traiter ce type de données dans le cas de partitionnement d'arbres phylogénétiques.

CHAPITRE IV

UTILISATION DE LA DISTANCE EUCLIDIENNE DANS LE CADRE DES K-MOYENNES POUR IDENTIFIER LES ARBRES CONSENSUS MULTIPLES

«Je dis seulement que nous nous trouvons en présence de deux espèces de fruits, aucune n'est bonne ni mauvaise. Ce sont deux espèces qui ne relèvent pas d'une même évolution et qu'on tente de mettre dans le même milieu.», Jean Morisset, 2009.

4.1 Préface

Dans ce chapitre, nous introduirons de nouveaux critères de classification d'arbres phylogénétiques mis en place dans le cadre de ce projet doctoral. Au chapitre précédent, nous avons présenté le nouvel algorithme que nous avons développé pour partitionner des arbres phylogénétiques en nous appuyant sur l'algorithme des *k*-médoïdes (Kaufman et Rousseeuw, 1990). Pour cela, nous avons utilisé deux indices de validité des clusters : Silhouette (*SH*) (Rousseew, 1987) et Caliński-Harabasz (*CH*) (Caliński-Harabasz, 1974). Pour chacun de ces deux indices, nous avons évalué deux variantes de la distance topologique de Robinson et Foulds (*RF*) (Robinson et Foulds, 1981) : avec ou sans sa mise au carré. Les résultats de cet algorithme, évalués en termes d'*ARI* moyen et les indices *SH* et *CH*, montrent que l'algorithme proposé est aussi compétitif que l'algorithme faisant appel au programme *Consense* du package Phylip (Felsenstein, 1985) qui recalcule les arbres consensus à chaque itération de

l'algorithme des *k*-moyennes employé selon la stratégie de Stockham *et al.* (voir les figures 3.6a et 3.6b). En termes de complexité temporelle, notre nouvel algorithme (voir les figures 3.6c et 3.6d) s'est révélé être bien plus avantageux que l'algorithme de Stockham *et al.* Dans ce chapitre, nous souhaitons améliorer la performance de notre algorithme de partitionnement d'arbres basé sur les *k*-médoïdes. Pour ce faire, nous allons introduire sa nouvelle variante, qui est basée sur la méthode des *k*-moyennes (MacQueen, 1967). Nous utiliserons ici quatre critères de validité des clusters, qui sont : *CH*, *SH*, *W* et la *statistique Gap* (Tibshirani, Walther et Hastie, 2001), adaptés au partitionnement d'arbres par les *k*-moyennes.

4.2 Introduction

Ces dernières années, le séquençage de dernières générations (NGS ou *Next Generation Sequencing*) a révolutionné la biologie moléculaire et l'écologie. Le NGS est une technique de séquençage rapide, fiable et peu coûteuse qui produit et augmente le nombre de séquences moléculaires disponibles d'une façon exponentielle (Mardis, 2008; Metzker, 2010; Glöckle *et al.*, 2014 ; Bertolini *et al.*, 2014). En effet, la production peu coûteuse de gros volumes de données de séquences stimule l'apparition des méthodes rapides d'alignement de séquences et d'inférence d'arbres phylogénétiques. Les méthodes de partitionnement d'arbres phylogénétiques en plusieurs groupes doivent aussi être accélérées pour pouvoir traiter des données de séquences volumineuses disponibles aujourd'hui.

Le partitionnement d'arbres phylogénétiques est un problème fondamental en bioinformatique. Souvent, les arbres phylogénétiques portent des informations importantes sur des histoires évolutives spécifiques qui caractérisent l'évolution des familles de gènes correspondantes (Hendy *et al.*, 1988). Il est donc souvent prématuré de sélectionner une seule et unique histoire d'évolution pour un groupe d'espèces données.

Afin de pallier ce problème, nous proposons un nouvel algorithme de partitionnement d'arbres basé sur la méthode des *k*-moyennes (Lloyd, 1957; MacQueen, 1967; Bock, 2007). Quatre indices de validité des clusters seront proposés et testés dans notre étude : le critère de Caliński-Harabasz (*CH*) (Caliński et Harabasz, 1974), noté également comme *E-CH*, le critère *W*, le critère Silhouette (*SH*) (Rousseeuw, 1987) et le critère *Gap* (Tibshirani, Walther et Hastie, 2001).

Le reste du chapitre est organisé en sections suivantes. Dans la section 4.3, nous présentons en détail notre nouvelle méthode de partitionnement d'arbres phylogénétiques basée sur l'algorithme des *k*-moyennes. Nous y décrivons l'algorithme des *k*-moyennes classique, indiquons la fonction objective classique, puis celle adaptée à des données d'arbres. Enfin, nous détaillons nos différents critères de sélection du meilleur partitionnement (indices de validité des clusters), en soulignant leurs avantages et leurs limites. Dans la section 4.4, nous décrivons nos simulations, ainsi que les ensembles de données générées pour tester notre nouvel algorithme. Au chapitre VI, nous considérerons des données réelles, telles que des données biologiques et des données linguistiques, et comparerons nos résultats à ceux fournis par les méthodes existantes. Dans la section 4.5, nous testons les différentes variantes d'indices de validité des clusters (*c.-à-d.*, *E-CH*, *SH*, *W* et la *statistique Gap*) utilisés avec notre nouvel algorithme. Dans un premier temps, nous comparons les indices *E-CH* et *SH* entre eux, car ces deux indices admettent une même limite, à savoir l'impossibilité de traiter des données homogènes (*c.-à-d.*, le cas $K = 1$), puis les indices *W* et *Gap* entre eux, puisque ces deux derniers indices peuvent identifier le cas des données homogènes. Dans un deuxième temps, nous utilisons les meilleurs indices issus de la première étape pour les tester avec les fonctions objectives différentes des *k*-moyennes (*c.-à-d.*, avec et sans la mise au carré de la distance topologique de Robinson et Foulds). Dans un troisième temps, nous sélectionnons les deux meilleures variantes de notre algorithme (une variante pour les données homogènes et une variante pour les données hétérogènes) pour les tester sur des

données bruitées (*noise data*). Enfin, dans un quatrième temps, nous comparons notre nouvel algorithme à la stratégie de Stockham *et al.* (2002) basée sur l'inférence d'arbres consensus à chaque étape des *k*-moyennes. Dans la section 4.6, nous rappelons les principaux concepts et résultats établis dans ce chapitre et donnons quelques pistes pour le travail futur.

4.3 Méthodologie

Notre algorithme prendra en entrée un ensemble d'arbres phylogénétiques Π définis sur un même ensemble d'espèces et retourne en sortie un ou plusieurs arbres consensus. Comme dans le chapitre III, chaque arbre consensus représente un sous-ensemble d'arbres phylogénétiques de Π. Pour chaque cluster identifié, l'algorithme retourne une liste d'éléments (*c.-à-d.*, arbres phylogénétiques) et l'arbre consensus correspondant. La sortie s'accompagne également de quelques résultats statistiques (*c.-à-d.*, les valeurs des indices *W*, *E-CH*, Silhouette et Gap). L'algorithme proposé emploie la méthode des *k*-moyennes adaptée au cas d'arbres phylogénétiques (*c.-à-d.*, pour regrouper itérativement *N* arbres phylogénétiques en *K* clusters disjoints). La répartition des données dans chaque cluster est choisie en minimisant la distance intragroupe. Généralement, les distances les plus communément utilisées avec les *k*-moyennes sont la distance euclidienne, la distance de Manhattan et la distance de Minkowski. La distance de Robinson et Foulds (*RF*) (Robinson et Foulds, 1981; Makarenkov et Leclerc, 2000) sera encore une fois utilisée pour comparer les topologies d'arbres phylogénétiques. Cette distance ne prend pas en considération les longueurs de branches. Rappelons-nous que l'arbre consensus majoritaire d'un ensemble d'arbres est un arbre médian de cet ensemble dans le sens de la distance de Robinson et Foulds (Barthélemy et McMorris, 1986). C'est une des raisons pour laquelle nous avons utilisé cette distance dans notre projet.

La problématique de trouver un partitionnement optimal selon le critère des *k*-moyennes est connu pour être NP-difficile (Drineas et *al.*, 2004). Plusieurs méthodes heuristiques, fonctionnant souvent en temps polynomial de $O(K \times N \times I \times M)$ pour trouver une solution approximative, ont été proposées, où I est le nombre d'itérations dans la boucle principale de l'algorithme et M est le nombre des variables caractérisant chacun des N objets considérés. Dans notre projet doctoral, nous avons utilisé la version des *k*-moyennes implémentée par Makarenkov et Legendre (Makarenkov et Legendre, 2001).

Nous avons constaté que l'un des inconvénients des *k*-moyennes dans le cas de partitionnement d'un ensemble d'arbres phylogénétiques survient lorsque les barycentres des clusters, qui n'ont pas de réelle importance pour les partitionnements obtenus, doivent être recalculés à chaque itération. C'est pour cette raison que nous proposerons de nouvelles versions de la fonction objective des *k*-moyennes permettant de réduire la complexité temporelle de la méthode.

Comme nous utiliserons activement la distance de Robinson et Foulds dans ce chapitre, nous établirons d'abord certaines de ses propriétés qui seront exploitées par la suite dans l'algorithme de partitionnement d'arbres.

4.3.1 La distance topologique de Robinson et Foulds n'est pas de type euclidien

Ici, nous présentons un contre-exemple montrant que la distance topologique de Robinson et Foulds n'est pas une distance de type euclidien.

4.3.1.1 Un contre-exemple

Nous présentons ici le contre-exemple suivant. Soit un ensemble de quatre arbres phylogénétiques $\{T_1, T_2, T_3, T_4\}$ à cinq feuilles (étiquettes) {1, 2, 3, 4, 5}. C'est le cas

le plus simple possible, car pour les arbres à quatre feuilles, la distance de Robinson et Foulds a les propriétés euclidiennes.

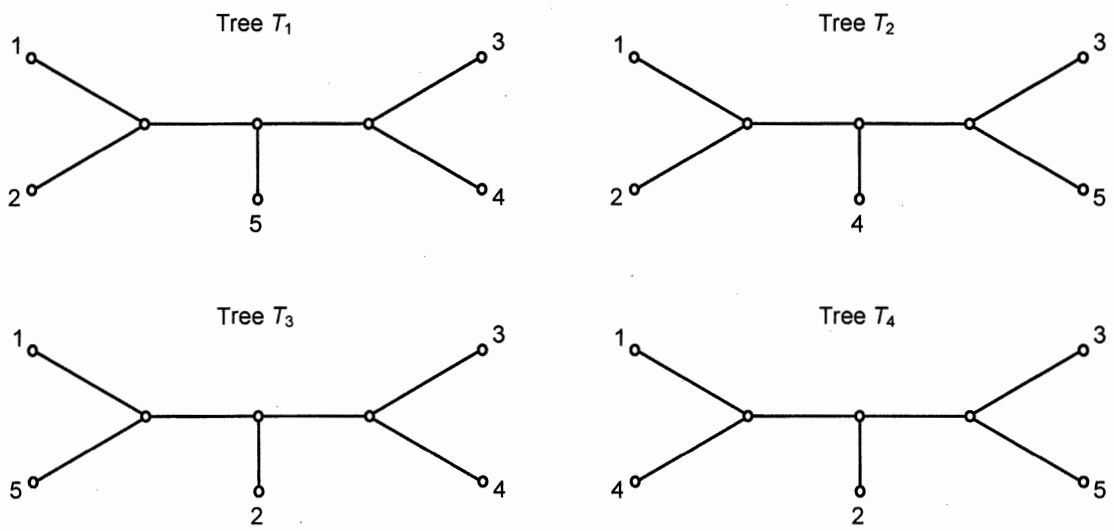


Figure 4.1 Quatre arbres phylogénétiques non enracinés : T_1, T_2, T_3 et T_4 à cinq feuilles utilisés pour montrer que la distance topologique de Robinson et Foulds n'est pas une distance euclidienne.

Les 1-bipartitions correspondent aux feuilles des arbres et sont communes aux quatre arbres de la figure 4.1.

On examine maintenant les 2-bipartitions, chacune définie pour un sous-ensemble à 2 éléments, et son complément. Pour ces quatre arbres, les 2-bipartitions sont définies comme suit : T_1 : {1, 2}, {3, 4}, T_2 : {1, 2}, {3, 5}, T_3 : {1, 5}, {3, 4}, et T_4 : {1, 4}, {3, 5}.

D'où : $d_{RF}(T_1, T_2) = 2$, $d_{RF}(T_2, T_3) = 4$ et $d_{RF}(T_1, T_3) = 2$.

Ce qui, dans le cas d'une figure euclidienne, placerait T_1 au centre de l'intervalle [T_2, T_3] (voir la figure 4.2).

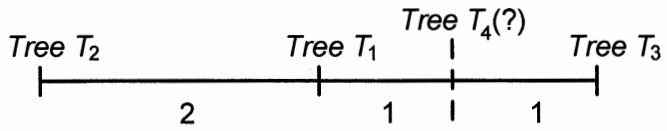


Figure 4.2 Illustration représentant la position des quatre arbres de la figure 4.1, utilisée pour montrer que la distance topologique de Robinson et Foulds n'est pas une distance euclidienne.

Il est à noter qu'on a aussi $d_{RF}(T_1, T_4) = 4$ et $d_{RF}(T_3, T_4) = 4$, plaçant T_4 sur la médiatrice de $[T_1, T_3]$, ce qui est incompatible avec le fait que $d_{RF}(T_2, T_4) = 2$.

4.3.2 L'algorithme des *k*-moyennes classique

L'algorithme des *k*-moyennes (ou *k-means*) (Lloyd, 1957; MacQueen, 1967) est un algorithme fréquemment utilisé pour l'analyse des données qui doivent être partitionnées en plusieurs groupes (Wu *et al.*, 2008). Ce qui rend l'algorithme des *k*-moyennes le plus utilisé est sa simplicité d'implémentation, ainsi que la rapidité d'exécution. Le partitionnement se réalise à la fois par un support statistique, de même que par un apprentissage automatique (*c.-à-d.*, apprentissage non supervisé). C'est une méthode dont le but est de diviser N observations (objets ou espèces) en K classes (K clusters) pour que chaque observation donnée appartienne à la classe avec le centroïde le plus proche. L'algorithme des *k*-moyennes se déroule comme suit :

Algorithme 4.1 des *k*-moyennes (MacQueen, 1967; Lloyd, 1957)

Étape 1. Choisir K centroïdes pris au hasard parmi l'ensemble de définitions des observations données.

Étape 2. Assigner à chaque observation donnée le centroïde le plus proche (*p. ex.* en appliquant le diagramme de Voronoï (Aurenhammer, 1991)).

Étape 3. Mettre à jour les coordonnées du centroïde de chaque partition (version de Lloyd). Notons que la mise à jour des centroïdes se fait à l'étape 2, après chaque déplacement d'une observation dans l'algorithme de MacQueen.

Répéter les étapes 2 et 3 jusqu'à la convergence qui sera atteinte soit lorsqu'il n'y aura plus de modifications dans l'ensemble des k centroïdes, soit lorsqu'on aura atteint le nombre d'itérations maximum I (le seuil préfixé).

L'algorithme est généralement exécuté plusieurs fois (généralement 100, 500 ou 1000) avec les centroïdes de départ aléatoires. Le meilleur partitionnement est ensuite sélectionné selon le score de l'indice de validité des clusters choisi.

L'algorithme des *k*-moyennes partitionne N observations données en K clusters en se basant sur la distance euclidienne au carré (voir la Formule 4.1). Une des principales contributions de cette thèse est d'élargir l'utilisation des *k*-moyennes aux distances non euclidiennes.

Notons par ailleurs que l'algorithme des *k*-moyennes, comme la plupart des algorithmes de partitionnement, ont les limitations suivantes :

- Le partitionnement final dépend fortement des centroïdes initiaux, qui sont généralement pris au hasard;
- Le nombre de clusters, K, doit être connu à l'avance;

- Chaque observation contribue de la même manière au critère de partitionnement, quel que soit son degré de pertinence;
- Le partitionnement des données homogènes se fait avec la condition $K \geq 2$.

4.3.3 La fonction objective des *k*-moyennes classiques

La fonction objective (*OF*) des *k*-moyennes correspond à la somme des distances euclidiennes au carré entre chaque objet *i* du groupe *k* et le centroïde de ce groupe (voir la formule 4.1). Le meilleur partitionnement sera celui qui minimise la valeur de la fonction objective. Dans la version traditionnelle des *k*-moyennes, la fonction objective est évaluée en utilisant la norme L^2 :

$$OF_{classique} = \sum_{k=1}^{K} \sum_{i=1}^{N_k} \left\| x_{ki} - m_k \right\|^2 , \tag{4.1}$$

où x_{ki} est l'objet *i* du cluster *k*, m_k (avec $k = 1 \ldots K$) est le centroïde du cluster *k* et N_k est le nombre d'objets dans le cluster *k*.

4.3.4 Fonction objective pour le cas d'arbres phylogénétiques

La fonction objective que nous considérons d'abord dans le cas de partitionnement d'un ensemble d'arbres phylogénétiques est définie comme suit :

$$OF = \sum_{k=1}^{K} \sum_{i=1}^{N_k} RF_{norm}(T_k^c, T_{ki}) . \tag{4.2}$$

où K est le nombre de clusters, N_k est le nombre d'arbres phylogénétiques dans le cluster *k*, RF_{norm} est la distance topologique de Robinson et Foulds entre deux arbres phylogénétiques (T_{ki} et T_k^c) normalisée par $2n - 6$ (il s'agit de la valeur maximale de

la distance *RF* entre deux arbres phylogénétiques binaires avec n feuilles), T_{ki} est l'arbre phylogénétique i du cluster k et T_k^c est l'arbre consensus majoritaire (centroïde) du cluster k. Cependant, le recalcul de l'arbre consensus majoritaire ou de l'arbre consensus majoritaire étendu pour chaque cluster et à chaque itération de l'algorithme est coûteux en termes de temps d'exécution. La complexité temporelle de la méthode d'inférence de l'arbre consensus majoritaire ou majoritaire étendu est $O(n^2 + nN^2)$ (Wareham, 1985). Même si un algorithme optimal pour calculer l'arbre consensus de la règle majoritaire, avec une complexité de $O(nN)$, a été récemment proposé (Jansson, Shen et Sung, 2013), cette complexité temporelle ne peut être obtenue quand les arbres phylogénétiques sont définis par leurs chaînes de caractères Newick comme c'est typiquement le cas en biologie computationnelle (Felsenstein, 2004). Ainsi, la complexité temporelle d'un algorithme simple de partitionnement d'arbres, basé sur les k-moyennes, tel que l'algorithme de Stockham *et al.* (Stockham, Wang et Warnow, 2002) est de $O(K \times n \times (n + N^2) \times I)$. Rappelons que l'algorithme de Stockham *et al.* recalcule l'arbre consensus après chaque opération basique de l'algorithme des k-moyennes, qui consiste à relocaliser un objet (*c.-à-d.*, l'arbre phylogénétique) d'un cluster à un autre et ensuite réassigner la valeur de la fonction objective (voir la formule 4.2).

Afin d'accélérer le processus de partitionnement d'arbres phylogénétiques, nous proposons d'utiliser la fonction objective suivante :

$$OF_{EA} = \sum_{k=1}^{K} \frac{1}{N_k} \sum_{i=1}^{N_{k-1}} \sum_{j=i+1}^{N_k} RF_{norm}(T_{ki}, T_{kj}), \tag{4.3}$$

qui peut être considérée comme une version approximative euclidienne (*Euclidean Approximation*) de la fonction objective définie par la formule 4.2. L'avantage principal de l'utilisation de cette fonction objective approximative (OF_{EA}) est de ne pas recalculer les arbres consensus pour les deux clusters impliqués dans chaque

opération basique de déplacement d'un élément dans l'algorithme des *k*-moyennes (voir l'Algorithme 4.1, version MacQueen). En effet, la complexité algorithmique d'une telle opération de base, dans un algorithme utilisant la formule (4.3), sera seulement de $O(N)$ puisque toutes les distances *RF* entre les arbres phylogénétiques dans Π peuvent être précalculées en $O(n \times N^2)$ (Sul et Williams, 2008). Cela mènera à la complexité algorithmique globale de $O(n \times N^2 + K \times N \times I)$ pour notre algorithme des *k*-moyennes de partitionnement d'arbres suivant la règle majoritaire étendu.

4.3.4.1 Relation entre les critères *OF* et OF_{EA}

Considérons les deux arbres phylogénétiques de la figure 4.3.

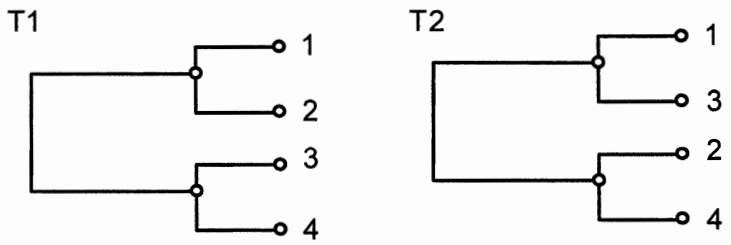


Figure 4.3 Deux arbres phylogénétiques T_1 = ((1,2),(3,4)) et T_2 = ((1,3),(2,4)), donnés par leurs chaînes Newick.

Ces deux arbres sont des arbres binaires non enracinés avec quatre feuilles. Soit Π = {Arbre 1 = T_1, Arbre 2 = T_1, Arbre 3 = T_2} l'ensemble de trois arbres phylogénétiques avec quatre feuilles dans lequel nous considérons la solution triviale consistant en un cluster d'arbres (*c.-à-d.*, $K = 1$). La matrice des distances *RF* entre les paires d'arbres dans Π est comme suit :

Tableau 4.1 Matrice des distances topologiques de Robinson et Foulds entre les paires d'arbres phylogénétiques de l'ensemble $\{T_1, T_2, T_3\}$.

	Arbre 1	Arbre 2	Arbre 3
Arbre 1	0	0	2
Arbre 2	0	0	2
Arbre 3	2	2	0

L'arbre consensus de la règle majoritaire de l'Arbre 1, l'Arbre 2 et l'Arbre 3 est évidemment l'arbre T_1. L'équation 4.3 donne $OF = 0 + 2 + 2 = 4$. Toutefois, la formule euclidienne donne: $OF_{EA} = \sum_{k=1}^{K} \frac{1}{N_k} \sum_{i=1}^{N_k - 1} \sum_{j=i+1}^{N_k} RF(T_{ki}, T_{kj}) = (0+2+2)/3 = 4/3$.

Ce contre-exemple nous montre que la fonction objective figurant dans la formule 4.3 n'est qu'une approximation de la fonction objective figurant dans la formule 4.2.

4.3.5 Le critère de validité des clusters de Caliński-Harabasz adapté au partitionnement d'arbres phylogénétiques par les *k*-moyennes

Si nous souhaitons utiliser les formules (3.3) et (3.5) du chapitre III (voir la section 3.3.4) pour le partitionnement d'un ensemble d'arbres, nous devons définir la notion de centroïde (barycentre ou arbre médian) pour un ensemble d'arbres donné en entrée. La procédure médiane (Barthélemy et Monjardet, 1981) est définie comme suit : l'ensemble d'arbres médians $Md(\Pi)$ de l'ensemble d'arbres donnés $\Pi = \{T_1, \ldots, T_l\}$ (définis sur le même ensemble de feuilles S) est l'ensemble de tous les arbres T définis sur S, tels que : $\sum_{i=1}^{l} RF(T, T_i)$ est minimum. Si l est impair, alors l'arbre consensus majoritaire $Maj(\Pi)$ de Π est un arbre unique dans $Md(\Pi)$. Si l est pair, alors $Md(\Pi)$ est composé de $Maj(\Pi)$ et quelques arbres plus résolus (voir Barthélemy

et McMorris (1986), pour plus de détails). Nous pouvons donc définir *Maj*(Π) comme le centroïde de tout ensemble d'arbres Π, puis utiliser les formules (3.2), (3.3) et (3.5) du chapitre III afin de définir l'indice de Caliński-Harabasz (*CH*) pour une partition donnée d'un ensemble d'arbres. Comme l'inférence d'un arbre consensus est très coûteuse en temps de calcul, nous avons décidé d'utiliser des versions euclidiennes de la fonction objective (la formule 4.3) et des coefficients SS_B et SS_W, qui ne requièrent pas le recalcul du centroïde (dans notre cas, il s'agira de l'arbre consensus majoritaire) de chaque cluster à chaque itération de l'algorithme.

Dans le cas de la distance euclidienne, chaque terme $SS_W^k = \sum_{i=1}^{N_k} \|x_{ik} - m_k\|^2$ dans la formule (3.5) du chapitre III peut être modifié da la manière suivante (nous nous restreignons à l'espace d'une dimension afin de simplifier la notation) :

$$SS_W^k = \sum_{i=1}^{N_k} \|x_{ik} - m_k\|^2 = \sum_{i=1}^{N_k} \left(x_{ik} - \frac{1}{N_k} \sum_{j=1}^{N_k} x_{jk} \right)^2 = \frac{1}{N_k^2} \sum_{i=1}^{N_k} \left(N_k x_{ik} - \sum_{j=1}^{N_k} x_{jk} \right)^2$$

$$= \frac{1}{N_k^2} \sum_{i=1}^{N_k} \left(N_k^2 x_{ik}^2 - 2N_k x_{ik} \sum_{j=1}^{N_k} x_{jk} + \left(\sum_{j=1}^{N_k} x_{jk} \right)^2 \right)$$

$$= \frac{1}{N_k^2} \left(N_k^2 \sum_{i=1}^{N_k} x_{ik}^2 - 2N_k \left(\sum_{j=1}^{N_k} x_{jk} \right)^2 + N_k \left(\sum_{j=1}^{N_k} x_{jk} \right)^2 \right)$$

$$= \frac{1}{N_k} \left(N_k \sum_{i=1}^{N_k} x_{ik}^2 - \left(\sum_{j=1}^{N_k} x_{jk} \right)^2 \right) = \frac{1}{N_k} \left((N_k - 1) \sum_{i=1}^{N_k} x_{ik}^2 - 2 \sum_{i=1}^{N_k - 1} \sum_{j=i+1}^{N_k} x_{ik} x_{jk} \right)$$

$$= \frac{1}{N_k} \left(\sum_{i=1}^{N_k - 1} \sum_{j=i+1}^{N_k} (x_{ik} - x_{jk})^2 \right) = \frac{1}{N_k} \left(\sum_{i=1}^{N_k - 1} \sum_{j=i+1}^{N_k} \|x_{ik} - x_{jk}\|^2 \right).$$

Ensuite, en remplaçant $\|x_{ik} - x_{jk}\|^2$ par $RF(T_{ki},T_{kj})$, nous obtenons la nouvelle variance globale intragroupe, SS_W, qui pourrait être calculée comme suit dans le cas de partitionnement d'arbres :

$$SS_W = \sum_{k=1}^{K} \frac{1}{N_k} \sum_{i=1}^{N_k-1} \sum_{j=i+1}^{N_k} RF(T_{ki}, T_{kj}), \tag{4.4}$$

où N_k est le nombre d'arbres dans la classe k, RF est la distance topologique de Robinson et Foulds entre deux arbres phylogénétiques, T_{ki} est l'arbre i de la classe k, T_{kj} est l'arbre j de la classe k. De plus, comme la distance RF n'est pas une distance euclidienne, cette formule reste une formule approximative.

Dans le cas où les arbres ne sont pas définis sur le même ensemble de feuilles (*c.-à-d.*, le cas de super-arbre), nous avons besoin de normaliser la distance RF. Nous obtenons donc la relation suivante :

$$RF_{norm}(T_i, T_j) = \begin{cases} \dfrac{RF(T_i, T_j)}{2n(T_i, T_j) - 6} & \text{si } n(T_i, T_j) > 3 \\ 0 & \text{si } n(T_i, T_j) \leq 3 \end{cases} \tag{4.5}$$

où $n(T_i,T_j)$ est le nombre de feuilles communes entre les arbres T_i et T_j. Ensuite, nous utiliserons la version normalisée de l'équation 4.4 (voir l'équation 4.6) pour couvrir le cas de super-arbre.

$$SS_W = OF_{EA} = \sum_{k=1}^{K} \frac{1}{N_k} \sum_{i=1}^{N_k-1} \sum_{j=i+1}^{N_k} RF_{norm}(T_{ki}, T_{kj}). \tag{4.6}$$

Dans le même ordre idée, dans le cas d'une distance euclidienne, la formule 3.3 du chapitre III peut être simplifiée de la manière suivante :

$$SS_B + SS_W = \sum_{k=1}^{K} N_k \|m_k - m\|^2 + \sum_{k=1}^{K}\sum_{i=1}^{N_k} \|x_{ik} - m_k\|^2$$

$$= \sum_{k=1}^{K} N_k \left(m_k^2 - 2mm_k + m^2\right) + \sum_{k=1}^{K}\sum_{i=1}^{N_k} \left(x_{ik}^2 - 2x_{ik}m_k + m_k^2\right)$$

$$= \sum_{k=1}^{K} \left(N_k m_k^2\right) - 2Nm^2 + Nm^2 + \sum_{k=1}^{K}\sum_{i=1}^{N_k} \left(x_{ik}^2\right) - 2\sum_{k=1}^{K} \left(N_k m_k^2\right) + \sum_{k=1}^{K} \left(N_k m_k^2\right)$$

$$= \sum_{k=1}^{K}\sum_{i=1}^{N_k} \left(x_{ik}^2\right) - Nm^2 = N\sum_{i=1}^{N} (x_i - m)^2 = \frac{1}{N}\left(\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} \|x_i - x_j\|^2\right).$$

Dans la dernière ligne du calcul ci-dessus, la première égalité est relative à la formule classique de $Var(X) = E(X^2) - E(X)^2$, alors que la seconde égalité est la conséquence de la simplification de SS_W^k (voir ci-dessus) appliquée à l'ensemble du jeu de données.

Donc, la nouvelle variance globale intergroupe, SS_B, peut être calculée comme suit (voir l'équation 4.7) :

$$SS_B = \frac{1}{N}\left(\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} RF_{norm}(T_i, T_j)\right) - SS_W . \qquad (4.7)$$

Une fois de plus, nous devons mentionner que la formule (4.7) n'est pas équivalente à la valeur exacte de SS_B dans le cas de partitionnement d'arbres (voir l'équation 3.3 au chapitre III). Le critère de validité des clusters de Caliński-Harabasz basé sur les formules euclidiennes approximatives (4.6) et (4.7) sera noté *E-CH.*

4.3.6 Le critère de validité des clusters Silhouette adapté au partitionnement d'arbres phylogénétiques par les *k*-moyennes

Nous allons réutiliser le critère Silhouette *SH* (Rousseeuw, 1987), décrit en détails dans la section 3.3.5 du chapitre III, lors du partitionnement d'arbres phylogénétiques par les *k*-moyennes. Ce critère a montré une meilleure performance en termes d'*ARI* moyen que le critère de Caliński-Harabasz dans le cas de l'algorithme des *k*-médoïdes (Kauffman et Rousseeuw, 1990); voir les figures 3.4, 3.5 et 3.6 au chapitre III.

Notons que les critères de Caliński-Harabasz et Silhouette ne nous permettent pas de comparer la solution consistant en un arbre consensus unique ($K = 1$) aux solutions admettant des arbres consensus multiples ($K \geq 2$). Il est évidemment préférable de trouver un arbre consensus unique dans le cas des données homogènes. C'est pour cette raison que nous définirons ensuite les critères de validité des clusters *W* et Gap permettant d'identifier des données d'arbres homogènes.

4.3.7 Le critère de validité des clusters : *W*

Nous définissons un nouveau critère *W* permettant de considérer le cas des données homogènes ($K = 1$). La valeur de la fonction, *W*, suivante (voir les équations 4.8 et 4.9) sera minimisée à chaque itération de l'algorithme :

$$W(\Pi) = \frac{1}{N-K}\sum_{k=1}^{K}\delta_k \rightarrow Min\,, \tag{4.8}$$

où :

$$\delta_k = \begin{cases} \dfrac{2}{N_k \times (N_k - 1)} \displaystyle\sum_{i=1}^{N_k-1}\sum_{j=i+1}^{N_k} RF_{norm}(T_{ki}, T_{kj}) & \text{Si } N_k > 1 \\ 0 & \text{Si } N_k = 1 \end{cases} \tag{4.9}$$

Chaque terme δ_k est la moyenne des distances *RF* entre tous les arbres appartenant au cluster *k*. Dans le cas où N_k est égal à 1, δ_k vaut 0. Nous divisons ensuite la somme des δ_k par le nombre de degrés de liberté, soit $N - K$.

4.3.8 Le critère de validité des clusters : la statistique *Gap*

La statistique *Gap* a été mise au point par Tibshirani *et al.* (Tibshirani, Walther et Hastie, 2001) pour estimer le nombre de clusters dans les données. Pour un ensemble de données $\{x_{iv}\}$, où $i = 1, 2, ..., N$ et $v = 1, 2, ..., M$, avec N représentant les observations indépendantes et M les traits (variables) mesurés pour chaque observation. Soit d_{ij} la distance entre les observations i et j. Le choix le plus commun pour d_{ij} est évidemment la distance euclidienne mise au carré : $\sum_v (x_{iv} - x_{jv})^2$. Supposons que nous ayons partitionné les données en K groupes, $(C_1, C_2, ..., C_K)$, où C_K représente le $k^{\text{ième}}$ groupe et $N_k = |C_k|$. La formule (4.11) donne la somme totale des distances *RF* entre les différents arbres phylogénétiques du cluster k :

$$D_k = \sum_{i,j \in C_k} d_{ij} = \sum_{i=1}^{N_k} \sum_{j=1}^{N_k} RF(T_{ki}, T_{kj}) . \tag{4.11}$$

Ensuite, la somme de toutes les moyennes des distances intragroupe, V_k, peut être calculée comme suit :

$$V_K = \sum_{k=1}^{K} \frac{1}{2N_k} D_k . \tag{4.12}$$

Enfin, la statistique *Gap*, qui reflète la qualité d'une solution de partitionnement donné avec K clusters, peut être définie comme suit :

$$Gap_N(K) = E_N^* \{\log(V_K)\} - \log(V_K) , \tag{4.13}$$

où E_N^* est une espérance pour un échantillon de taille N de la distribution de référence. La formule suivante (voir Tibshirani *et al.* (2001) pour l'explication complète concernant la définition de $\log(V_K)$), a été utilisée dans notre algorithme:

$$E_N^*\{\log(V_K)\} = \log(Nn/12) - (2/n)\log(K), \tag{4.14}$$

où n est le nombre de feuilles dans l'arbre.

La plus grande valeur de la statistique *Gap* correspond au partitionnement optimal.

4.4 L'algorithme des *k*-moyennes adapté au partitionnement d'arbres phylogénétiques

Dans la section 4.3.2, nous avons décrit l'algorithme des *k*-moyennes classique (Lloyd, 1957; MacQueen, 1967) et évoqué la différence entre deux principales versions (de Lloyd et de MacQueen) de cet algorithme de partitionnement populaire. La version de Lloyd met à jour les centroïdes à l'étape 3, tandis que la version de MacQueen met à jour les centroïdes plus fréquemment - aux étapes 2 et 3 (voir l'Algorithme 4.1). Nous avons approximé la fonction objective des *k*-moyennes pour le cas d'arbres phylogénétiques (voir l'équation 4.3), en évitant ainsi le recalcul des centroïdes lors de chaque tentative de déplacement d'élément entre les clusters et en diminuant en même temps la complexité algorithmique de la méthode. Notre algorithme des *k*-moyennes adapté au partitionnement d'arbres phylogénétiques se déroule donc comme suit :

Algorithme 4.2 des *k*-moyennes adapté au partitionnement d'arbres phylogénétiques

Étape 1. Partitionner aléatoirement les N arbres phylogénétiques donnés en K clusters

Calculer OF_{EA} _initiale en utilisant la formule (4.3)

OF_{EA}_meilleure = OF_{EA}_initiale

Étape 2. Pour chaque arbre phylogénétique *tree* **faire** :

Pour chaque cluster *groupe* **faire** :

Déplacer *tree* dans *groupe*

Si ($OF_{EA} = \sum_{k=1}^{K} \frac{1}{N_k} \sum_{i=1}^{N_{k-1}} \sum_{j=i+1}^{N_k} RF_{norm}(T_{ki}, T_{kj}) < OF_{EA}$_meilleure) **alors** :

Accepter le déplacement de *tree* dans *groupe*

OF_{EA}_meilleure = $OF_{EA} = \sum_{k=1}^{K} \frac{1}{N_k} \sum_{i=1}^{N_{k-1}} \sum_{j=i+1}^{N_k} RF_{norm}(T_{ki}, T_{kj})$

Sinon

Annuler le déplacement de *tree*.

Fin Si

Répéter l'étape 2 jusqu'à la convergence qui sera atteinte soit lorsqu'il n'y aura plus de modifications dans les k groupes d'arbres formés, soit lorsqu'on aura atteint le nombre maximum d'itérations I (seuil préfixé).

L'algorithme 4.2 sera exécuté plusieurs fois avec les partitionnements de départ aléatoires. Le meilleur partitionnement sera ensuite choisi en fonction de la valeur de l'indice de validité des clusters retenu (*E-CH*, *SH*, *W* ou *Gap*, dans notre cas).

Notons que dans nos simulations (voir la section suivante) nous avons également testé une version alternative de cet algorithme en remplaçant la distance RF par RF^2 dans l'équation (4.3).

4.5 Protocole des simulations pour l'algorithme des *k*-moyennes

Dans cette section, nous décrirons les ensembles des données simulées et examinées dans notre étude. Ensuite, nous présenterons en détail les résultats de nos simulations. Nous avons réalisé trois séries de simulations qui sont comme suit :

4.5.1 Description de la première série de simulations

La *première série de simulations* consistait à tester les différentes variantes de notre nouvel algorithme de partitionnement d'arbres par les *k*-moyennes. Notre protocole de simulations incluait trois étapes. Durant la première étape, nous avons généré aléatoirement K arbres phylogénétiques binaires, T_1, ..., T_K, avec n feuilles chacun, où K = 1, ..., 10 et n = 8, 16, 32, 64. Durant la deuxième étape, pour chaque arbre phylogénétique T_i (où i = 1 ... K) obtenu lors de la première étape, nous avons généré aléatoirement un ensemble de 100 arbres correspondant à la classe i, pour chacun des quatre intervalles de bruit définis ci-dessous. Chaque élément T de la classe i était un arbre phylogénétique tel que les pourcentages de similarités entre T et T_i (mesurés à l'aide de la distance *RF* normalisée) variaient comme suit : de 0% à 10%, de 10% à 25%, de 25% à 50%, et de 50% à 75%. Ces intervalles correspondaient au niveau de bruit (10%, 25%, 50% ou 75%), respectivement. Durant la troisième étape, nous avons appliqué notre algorithme à nos jeux de données.

4.5.2 Description de la deuxième série de simulations

La *deuxième série de simulation* consistait à supprimer aléatoirement des feuilles dans les arbres phylogénétiques issues de nos données d'entrée. Le pourcentage de feuilles supprimées était respectivement 10%, 25% et 50% de l'ensemble des feuilles de l'arbre phylogénétique source. Pour cela, nous avons suivi le protocole de notre première série de simulations (voir la section 4.5.1), auquel nous avons ajouté une dernière étape consistant à enlever au hasard certaines feuilles dans les ensembles

d'arbres simulés. Par conséquent, nous avons évalué la capacité de notre algorithme de résoudre le problème des données manquantes, ou le problème de construction de super-arbres, qui est une problématique courante en biologie computationnelle moderne (voir le chapitre V pour plus de détails sur cette problématique).

4.5.3 Description de la troisième série de simulations

La *troisième série de simulations* servait à comparer notre nouvel algorithme des *k*-moyennes (utilisant les critères présentés) avec la version des *k*-moyennes de Stockham *et al.* (2002) (utilisant le recalcul des arbres consensus). Nous avons réalisé 100 simulations avec des niveaux de bruits différents ajoutés aux arbres phylogénétiques : 10%, 25%, 50% et 75% de bruits, respectivement (selon la distance *RF* normalisée). Dans cette série de simulations, nous avons testé les arbres avec 8, 16, 32, 64 et 128 feuilles. Nous avons traité uniquement le cas $K = 5$ dans cette dernière série de simulations.

4.5.4 Environnement de programmation

Pour réaliser nos trois séries de simulations décrites aux points 4.5.1 à 4.5.3 ci-dessus, nous avons utilisé le serveur HPC (*High Performance Computing*) de Calcul Québec (Guillimin) avec CentOS de 64-bit, release 6.6, comme infrastructure, équipé avec le processeur Intel® Xeon® E5-2650 CPU (2.00 GHz) et 24 Gb de RAM.

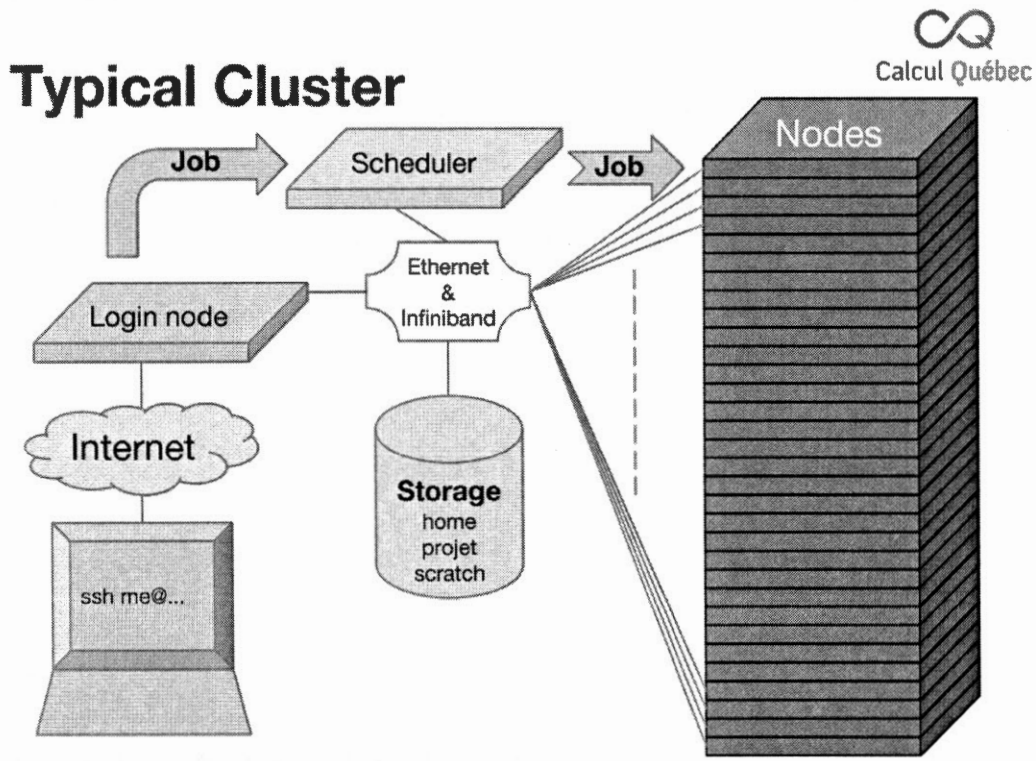


Figure 4.4 Flux de travail standard permettant l'utilisation du serveur Guillimin (figure reproduite d'après l'atelier du 29 septembre 2016 de McGill-HPC, intitulé "*Introduction to Advanced Research Computing*"[1]).

Le programme de simulations a été écrit en C/C++, compilé en utilisant le compilateur gcc-4.7.2 et parallélisé à l'aide de OpenMP. Nous avons soumis les tâches au serveur tel que décrit sur la figure 4.3. Enfin, nous avons traité les résultats en utilisant initialement mySQL, puis R-3.1.2.

---

[1] http://www.hpc.mcgill.ca/index.php/training#intro_linux

4.6 Résultats des simulations pour l'algorithme des *k*-moyennes

Les résultats de notre *première série de simulations* sont illustrés sur les figures 4.5 à 4.9. Les résultats présentés sont les moyennes prises sur toutes les combinaisons de nos paramètres (voir la section 4.5.1) pour 100 ensembles de données différents.

Nous avons d'abord testé, puis comparé, les critères *E-CH* et *SH* (voir les figures 4.5 et 4.6) utilisés dans notre algorithme de partitionnement d'arbres pour choisir le nombre de clusters. Dans un premier temps, les données ont été générées pour K allant de 2 à 5 (voir la figure 4.5), puis dans un deuxième temps pour K allant de 6 à 10 (voir la figure 4.6). Les deux critères *E-CH* et *SH* ne peuvent pas traiter le cas des données homogènes. C'est pour cela que la figure 4.5 ne contient pas les résultats de l'*ARI* moyen pour $K = 1$. Pour les deux critères, l'*ARI* moyen croît quand le nombre de feuilles augmente (voir les figures 4.5b et 4.6b). La figure 4.5(a) montre également la stabilité des résultats pour le critère *E-CH* indépendamment du nombre de clusters pour K allant de 2 à 5, alors que les résultats obtenus à l'aide du critère *SH* s'améliorent légèrement, passant de l'*ARI* moyen de 0.9075 pour $K = 2$ à 0.9651 pour $K = 5$. Quand K varie de 6 à 10, nous constatons que les courbes des deux critères, *E-CH* et *SH*, décroissent quand le nombre de clusters augmente (voir la figure 4.6a). Enfin, l'*ARI* moyen décroît pour les deux critères quand le niveau de bruit augmente (voir les figures 4.5c et 4.6c). Notons une meilleure performance de l'algorithme lorsque K varie de 2 à 5 (voir la figure 4.5c) comparativement au cas où K varie de 6 à 10 (voir la figure 4.6c). Dans le premier cas, l'*ARI* moyen est de 0.9892 pour le critère *E-CH* et de 0.9465 pour le critère *SH*, alors que dans le deuxième cas il est de 0.9171 pour *E-CH* et de 0.8989 pour *SH*.

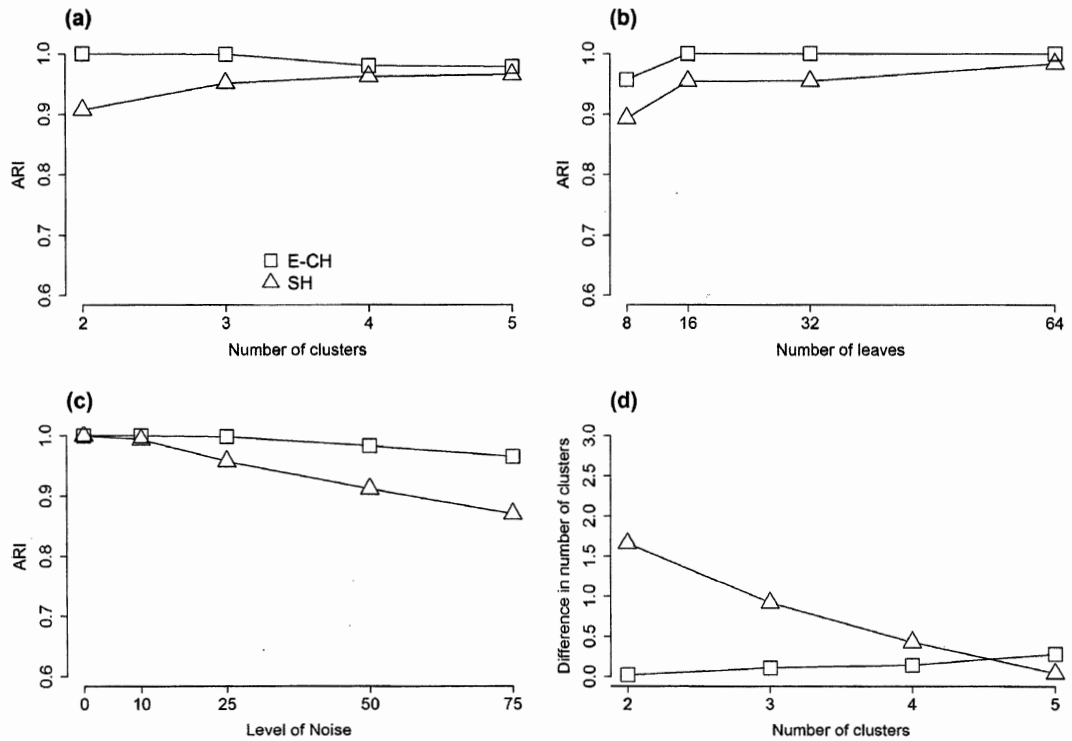


Figure 4.5 Évolution de l'*ARI* moyen en fonction : (a) du nombre de clusters, (b) du nombre de feuilles dans l'arbre phylogénétique, et (c) du niveau de bruit, quand le nombre de clusters, *K*, varie de 2 à 5. Le graphique (d) représente la moyenne des différences absolues entre le nombre de clusters trouvés par notre algorithme et le nombre de clusters générés, pour *K* variant de 2 à 5. Nous avons comparé les deux critères de validité des clusters suivants : 1) le critère *CH* Euclidien (*E-CH*) (□) et 2) le critère *SH* (Δ), avec la distance *RF* non mise au carré (dans les deux cas).

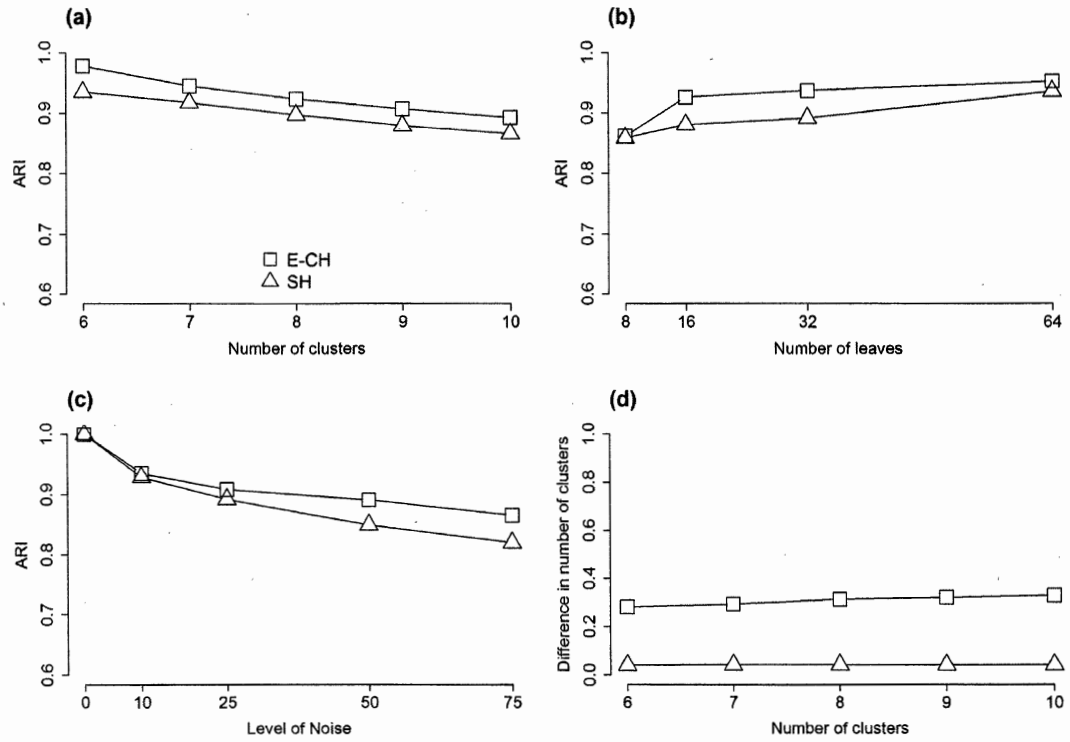


Figure 4.6 Évolution de l'*ARI* moyen en fonction : (a) du nombre de clusters, (b) du nombre de feuilles dans l'arbre phylogénétique, et (c) du niveau de bruit, quand le nombre de clusters, K, varie de 6 à 10. Le graphique (d) représente la moyenne des différences absolues entre le nombre de clusters trouvés par notre algorithme et le nombre de clusters générés, pour K variant de 6 à 10. Nous avons comparé les deux critères de validité des clusters suivants : 1) le critère *CH* Euclidien (*E-CH*) (□) et 2) le critère *SH* (Δ), avec la distance *RF* non mise au carré (dans les deux cas).

Nous avons comparé les critères *E-CH* et *SH* afin de retenir le critère le plus performant pour le traitement des données hétérogènes (*c.-à-d.*, quand $K \geq 2$). Dans l'ensemble des résultats (voir les figures 4.5 et 4.6), nous pouvons constater que le critère *E-CH* est plus performant que le critère *SH* quand le partitionnement d'arbres par les *k*-moyennes est effectué. En conséquence, nous n'utiliserons désormais que le

critère *E-CH* dans nos prochaines simulations pour traiter le cas des données hétérogènes.

Les figures 4.7 et 4.8 montrent l'évolution des valeurs de l'*ARI* moyen pour les critères *W* et *Gap*. Ici, nous avons d'abord réalisé les simulations pour *K* variant de 1 à 5 (voir la figure 4.7), puis pour *K* variant de 6 à 10 (voir la figure 4.8). Contrairement aux critères *E-CH* et *SH*, les critères *W* et *Gap* permettent de traiter le cas des données homogènes (*c.-à-d.*, quand $K = 1$). Pour les deux critères, l'*ARI* moyen croît quand le nombre de feuilles augmente (voir les figures 4.7b et 4.8b), mais il se stabilise pour le critère *W* lorsque *K* varie de 6 à 10 (voir la figure 4.8b). La figure 4.7a montre un très bon résultat du critère *W* lorsque $K = 1$. Cependant, les performances de *W* décroissent quand le nombre de clusters augmente pour se stabiliser au delà du cas $K = 5$ (voir la figure 4.8a). Enfin, l'*ARI* moyen décroît pour les deux critères quand le niveau de bruit augmente (voir les figures 4.7c et 4.8c). Notons une meilleure performance de l'algorithme lorsque *K* varie entre 1 et 5 (voir la figure 4.7c) comparativement au cas où *K* varie entre 6 et 10 (voir la figure 4.8c).

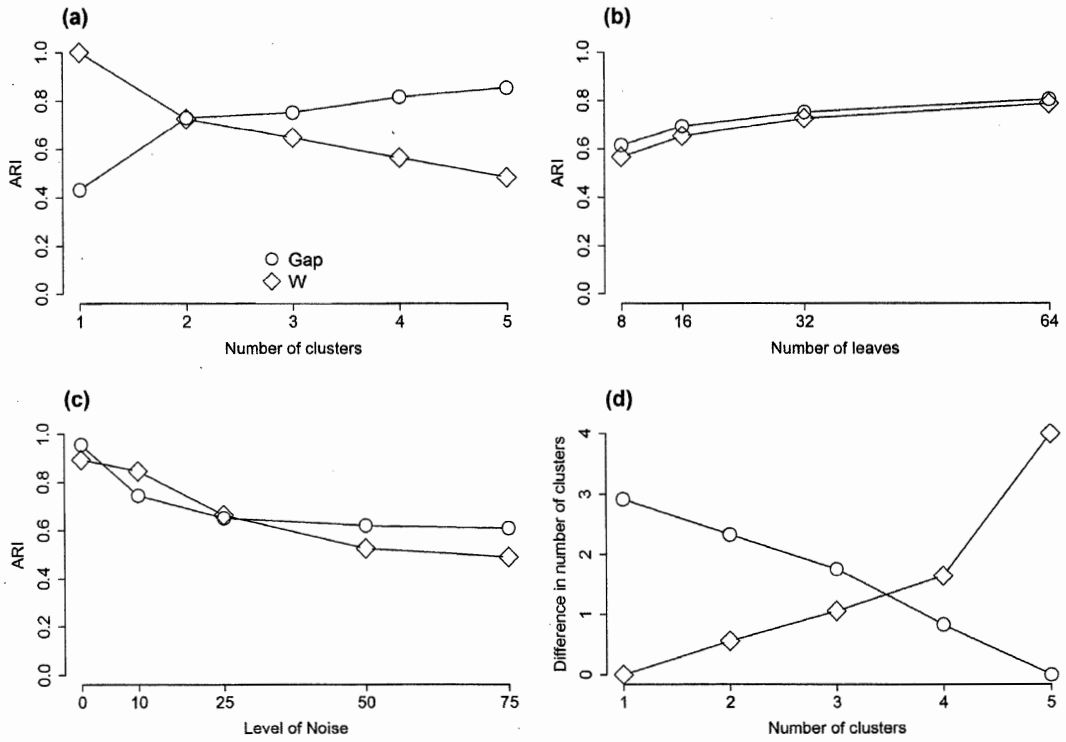


Figure 4.7 Évolution de l'*ARI* moyen en fonction : (a) du nombre de clusters, (b) du nombre de feuilles dans l'arbre phylogénétique, et (c) du niveau de bruit, quand le nombre de clusters, K, varie de 1 à 5. Le graphique (d) représente la moyenne des différences absolues entre le nombre de clusters trouvés par notre algorithme et le nombre de clusters générés, pour K variant de 1 à 5. Nous avons comparé les deux critères de validité des clusters suivants : 1) le critère *Gap* (○) et 2) le critère *W* (◊), avec la distance *RF* non mise au carré (dans les deux cas).

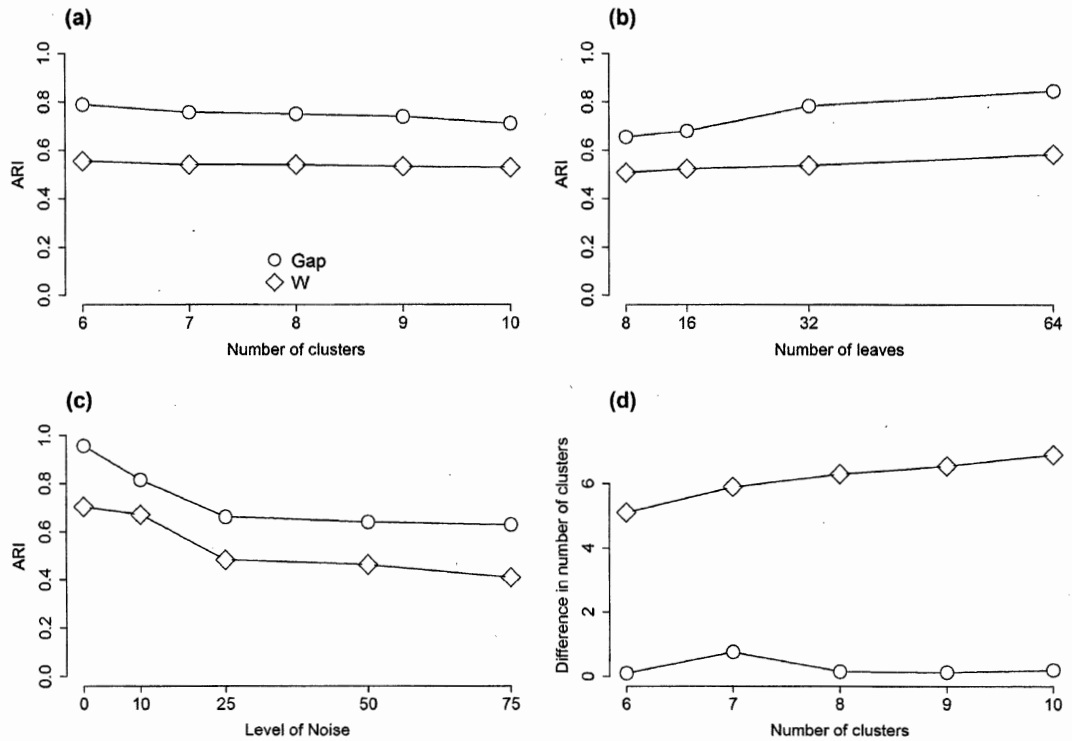


Figure 4.8 Évolution de l'*ARI* moyen en fonction : (a) du nombre de clusters, (b) du nombre de feuilles dans l'arbre phylogénétique, et (c) du niveau de bruit, quand le nombre de clusters, K, varie de 6 à 10. Le graphique (d) représente la moyenne des différences absolues entre le nombre de clusters trouvés par notre algorithme et le nombre de clusters générés, pour K variant de 6 à 10. Nous avons comparé les deux critères de validité des clusters suivants : 1) le critère *Gap* (○) et 2) le critère *W* (◊), avec la distance *RF* non mise au carré (dans les deux cas).

Nous pouvons donc conclure que les deux critères comparés, *Gap* et *W*, sont pertinents pour traiter le cas des données homogènes, mais lorsque $K \geq 2$, ces deux critères ne sont pas aussi efficaces, comparativement aux critères *E-CH* et *SH* examinés précédemment.

Dans l'ensemble des résultats (voir les figures 4.7 et 4.8), nous pouvons constater que le critère *W* était généralement plus performant que le critère *Gap*. Nous utiliserons donc désormais le critère *W* pour traiter le cas des données homogènes.

La figure 4.9 résume les résultats obtenus en utilisant les quatre critères employés dans le cadre de partitionnement d'arbres phylogénétiques par la méthode des *k*-moyennes lorsque les arbres sont définis sur un même ensemble de feuilles.

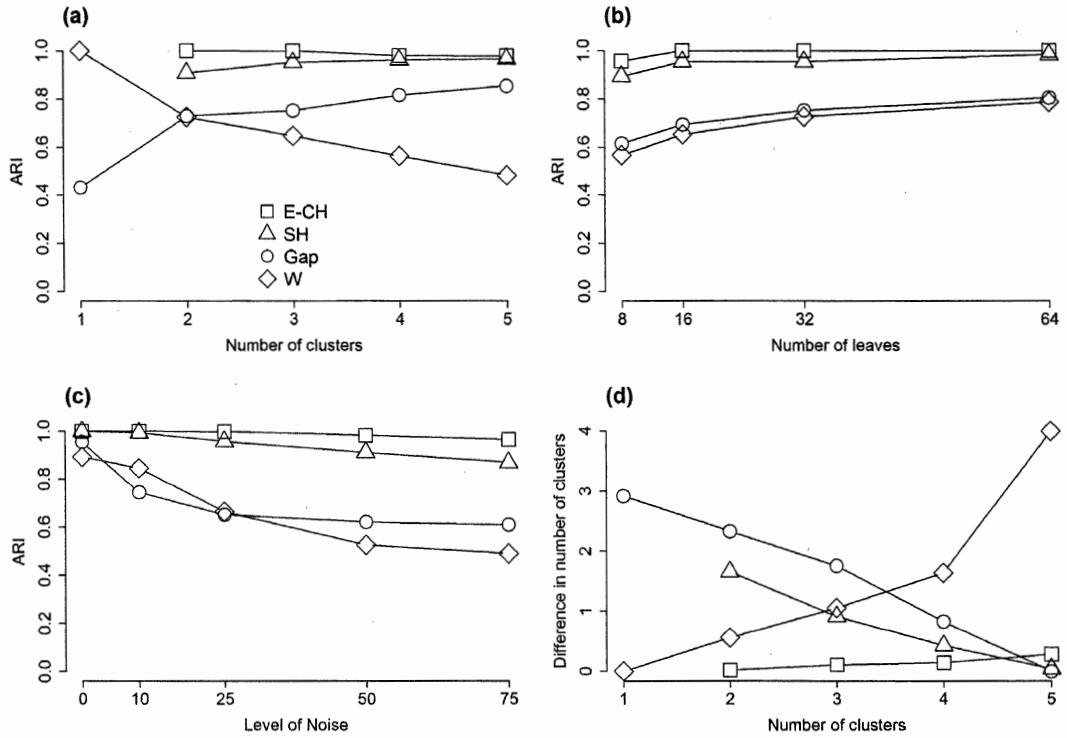


Figure 4.9 Évolution de l'*ARI* moyen en fonction : (a) du nombre de clusters, (b) du nombre de feuilles dans l'arbre phylogénétique, et (c) du niveau de bruit, quand le nombre de clusters, *K*, varie de 1 (ou 2) à 5. Le graphique (d) représente la moyenne des différences absolues entre le nombre de clusters trouvés par notre algorithme et le nombre de clusters générés, pour *K* variant de 1 (ou 2) à 5. Nous avons utilisé quatre indices de validité des clusters : 1) l'indice *E-CH* (□), 2) l'indice *SH* (Δ), 3) l'indice

Gap (○) et 4) l'indice *W* (◊), avec la distance *RF* non mise au carré (dans tous les cas).

Les résultats présentés sur la figure 4.9a suggèrent que la qualité de clustering de l'algorithme basée sur notre nouveau critère *E-CH* ne dépend pas du nombre de clusters. D'autre part, les valeurs de l'*ARI* moyen pour le critère *W* sont plus petites pour un plus grand nombre de clusters (*ARI* = 0.974 pour *K* = 1 et *ARI* = 0.4831 pour *K* = 5). Ces tendances sont confirmées par les résultats présentés sur la figure 4.9d. Si les données sont homogènes (*c.-à-d.*, *K* = 1), le critère *W* peut être recommandé. La figure 4.9b montre une légère augmentation des valeurs de l'*ARI* à mesure que le nombre de feuilles augmente. Cependant, la figure 4.9c montre que le critère *W* est beaucoup plus affecté par le bruit que le critère *E-CH*.

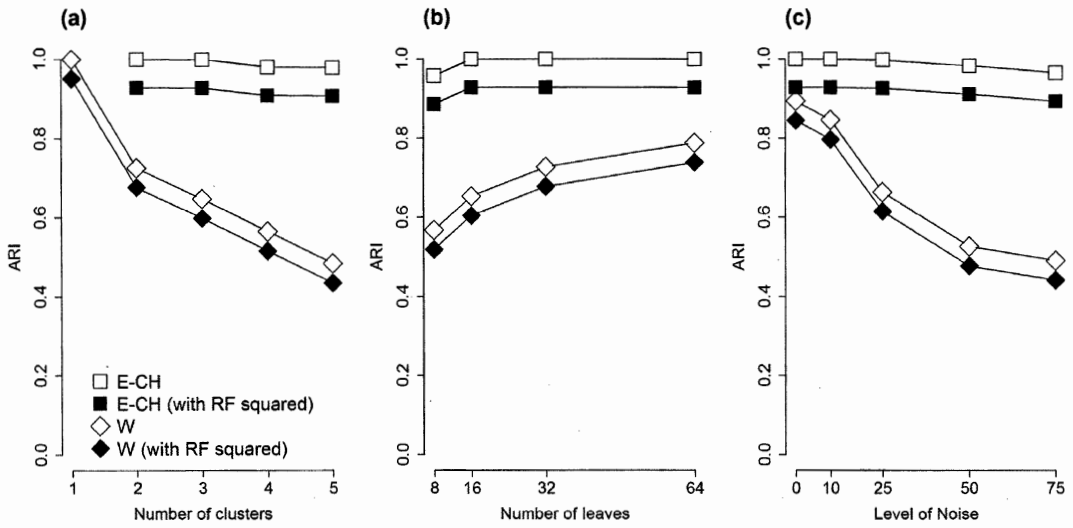


Figure 4.10 Les performances des quatre versions de notre algorithme des *k*-moyennes de partitionnement d'arbres phylogénétiques, mesurées en termes d'*ARI* moyen, en fonction du : (a) nombre de clusters, (b) nombre de feuilles et (c) niveau de bruit, lorsque le nombre de clusters, *K*, variant de 1 (ou 2) à 5, est supposé être connu. Les quatre versions de notre algorithme étaient basées sur les critères

suivants : 1) *E-CH* avec *RF* (□), 2) *E-CH* avec *RF* mise au carré (■), 3) *W* avec *RF* (◊), et 4) *W* avec *RF* (♦) mise au carré.

De plus, on peut remarquer que les versions de notre algorithme basées sur les critères *E-CH* et *W* et la distance *RF* non mise au carré donnent des meilleurs résultats que leurs analogues utilisant la distance *RF* mise au carré (voir la figure 4.10). Il convient de noter que Stockham *et al.* (Stockham, Wang et Warnow, 2002) ont utilisé la métrique *RF* quadratique dans leur algorithme. Dans tous les cas, sauf le cas $K = 1$ lorsque l'indice *E-CH* ne peut pas être calculé, les algorithmes de partitionnement d'arbres basés sur *E-CH* fournissent des meilleurs résultats que ceux basés sur *W*.

Notre *deuxième série de simulations* consistait à évaluer les performances de notre algorithme après l'élimination aléatoire des feuilles (espèces) dans les arbres phylogénétiques d'entrée, répondant à la problématique d'inférence de super-arbres. Les pourcentages de feuilles enlevées que nous avons considérés étaient : 0%, 10%, 25% et 50%. Les résultats obtenus en utilisant le critère *E-CH* sont indiqués sur la figure 4.11. La figure 4.11a présente la variation de l'*ARI* moyen par rapport au nombre de clusters et montre que la performance en termes d'*ARI* moyen baisse légèrement quand le nombre de clusters augmente. Avec 0%, 10%, 25% et 50% des feuilles enlevées, les valeurs de l'*ARI* moyens baissent légèrement lorsque le nombre de clusters augmente. Par exemple, le score de l'*ARI* moyen pour 50% de feuilles enlevées des arbres phylogénétiques était de 0.88 pour 2 clusters et de 0.81 pour 5 clusters. Quand aucune feuille n'a été enlevée des arbres, l'*ARI* moyen obtenu était de 1.0 pour 2 clusters et de 0.97 pour 5 clusters (voir la figure 4.11a). La figure 4.11b montre que l'impact de l'élimination des feuilles est particulièrement perceptible pour les petites phylogénies (*c.-à-d.*, pour $n = 8$ et $n = 16$). La figure 4.11c montre l'évolution de l'*ARI* moyen en fonction du bruit ajouté.

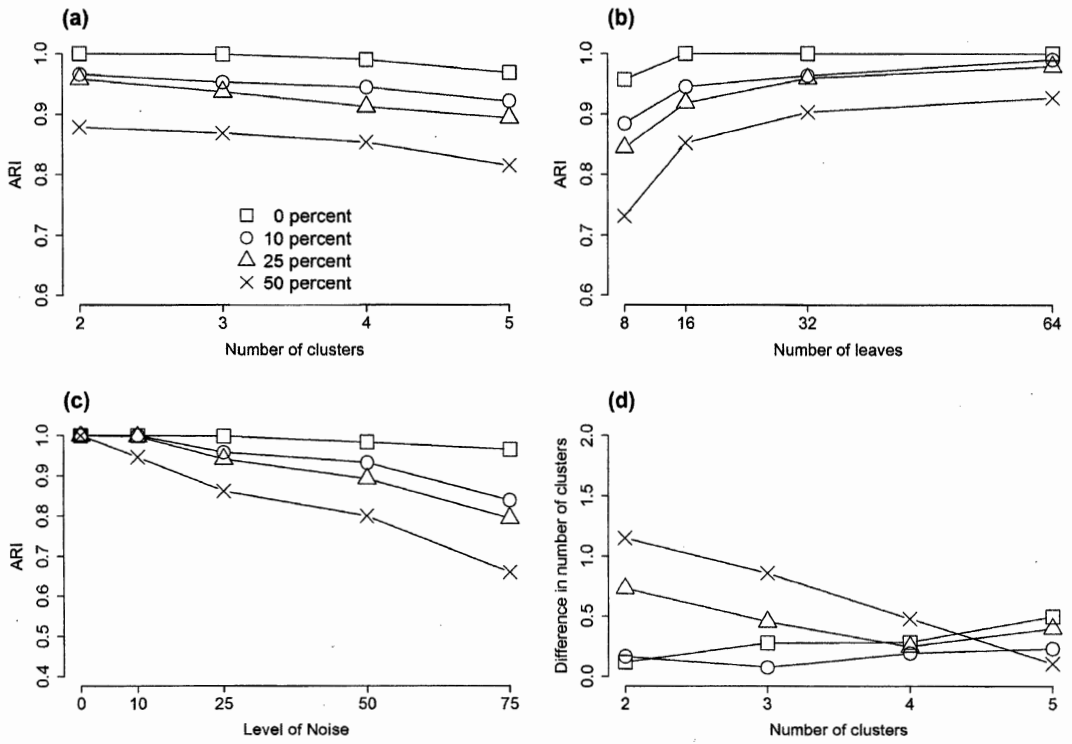


Figure 4.11 Les performances de notre algorithme, mesurées par l'*ARI* moyen, basé sur l'indice de validité des clusters *E-CH* et la distance *RF* non mise carré, en fonction du : (a) nombre de clusters, (b) nombre de feuilles dans l'arbre phylogénétique, et (c) niveau de bruit, quand le nombre de clusters, *K*, variait de 2 à 5. Le graphique (d) représente la moyenne des différences absolues entre le nombre de clusters trouvés par notre algorithme et le nombre de clusters généré. Nous avons fait varier le pourcentage de feuilles enlevées des arbres phylogénétiques : 0% (□), 10% (○), 25% (Δ) et 50% (×) de feuilles ont été enlevées des arbres.

D'autre part, les résultats obtenus en utilisant le critère *W* (voir les figures 4.12a - 4.12d) paraissent moins discriminants que les résultats obtenus en utilisant le critère *E-CH* dans le cas d'arbres phylogénétiques avec les feuilles enlevées.

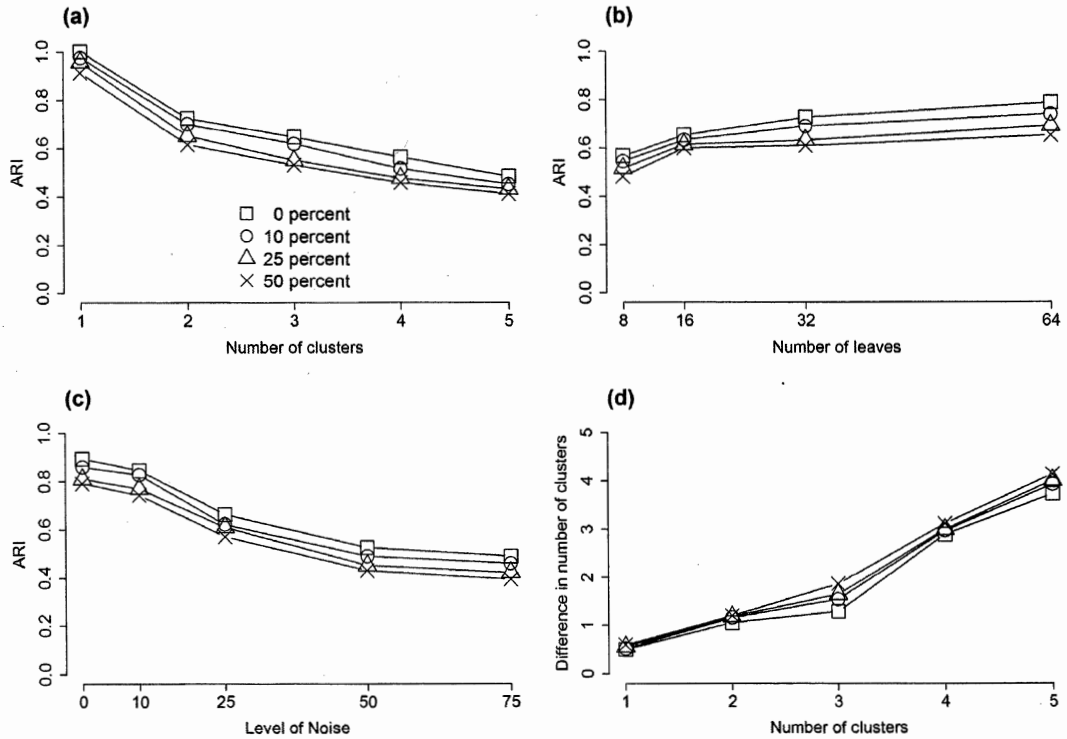


Figure 4.12 Les performances de notre algorithme, mesurées par l'*ARI* moyen, basé sur l'indice de validité des clusters *W* et la distance *RF* non mise carré, en fonction du : (a) nombre de clusters, (b) nombre de feuilles dans l'arbre phylogénétique, et (c) niveau de bruit, quand le nombre de clusters, *K*, variait de 1 à 5. Le graphique (d) représente la moyenne des différences absolues entre le nombre de clusters trouvés par notre algorithme et le nombre de clusters généré. Nous avons fait varier le pourcentage de feuilles enlevées des arbres phylogénétiques : 0% (□), 10% (○), 25% (Δ) et 50% (×) de feuilles ont été enlevées des arbres.

Dans la *troisième série de simulations*, nous avons comparé notre algorithme basé sur l'indice *E-CH* et la distance *RF* non mise au carré (voir la formule 4.3) à l'approche de partitionnement d'arbres de Stockham *et al.* (Stockham, Wang et Warnow, 2002).

L'approche de Stockham *et al.* procède par l'inférence d'arbres consensus, représentant les centroïdes des clusters, à chaque opération de base de l'algorithme des *k*-moyennes (voir la formule 4.2 avec la distance *RF* mise au carré, comme c'était implémenté par Stockham *et al.*). Notre comparaison a été faite en fonction de la qualité des partitionnements obtenus (voir les figures 4.13a et 4.13b) et du temps d'exécution (voir les figures 4.14a et 4.14b). Le nombre de feuilles dans les arbres phylogénétiques variait de 8 à 128 et le niveau de bruit de 10% à 75%. Les autres paramètres étaient ceux présentés dans la section 4.5.1. Une fois de plus, tout comme lors de la première série de simulations, 100 jeux de données ont été générés pour chaque combinaison de paramètres. Les résultats présentés sur les figures 4.13 (a et b) suggèrent que le nouvel algorithme basé sur le critère *E-CH*, défini par la variance globale intragroupe indiquée par la formule 4.6 et la variance globale intergroupe indiquée par la formule 4.7, a généralement surpassé l'approche de partitionnement d'arbres de Stockham *et al.*, qui recalcule les arbres consensus à chaque étape de l'algorithme. La différence dans les résultats était plus grande pour de petits arbres phylogénétiques, *c.-à-d.*, quand *n* variait de 8 à 32 (voir la figure 4.13a).

De plus, notre algorithme était de loin la meilleure méthode en termes de temps d'exécution pour les deux paramètres de cette simulation, *c.-à-d.*, le nombre de feuilles dans les arbres phylogénétiques (voir la figure 4.14a) et le nombre d'arbres phylogénétiques fournis en entrée (voir la figure 4.14b).

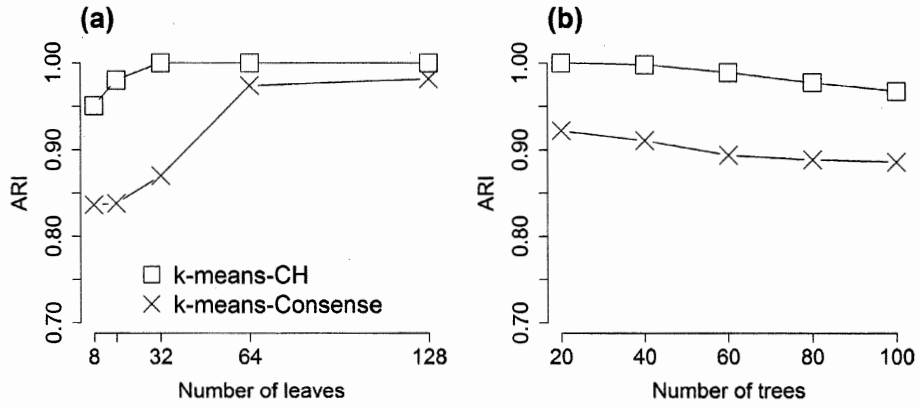


Figure 4.13 Comparaison de notre algorithme, basé sur l'indice de validité des clusters *E-CH* et la distance *RF* non mise au carré (□), avec à l'approche de Stockham *et al.* basée sur l'inférence d'arbre consensus et la distance *RF* quadratique (×) (Stockham, Wang et Warnow, 2002). Cette comparaison a été faite, en utilisant l'ARI moyen, en fonction du nombre de feuilles dans les arbres phylogénétiques (a) et du nombre d'arbres phylogénétiques fournis en entrée (b).

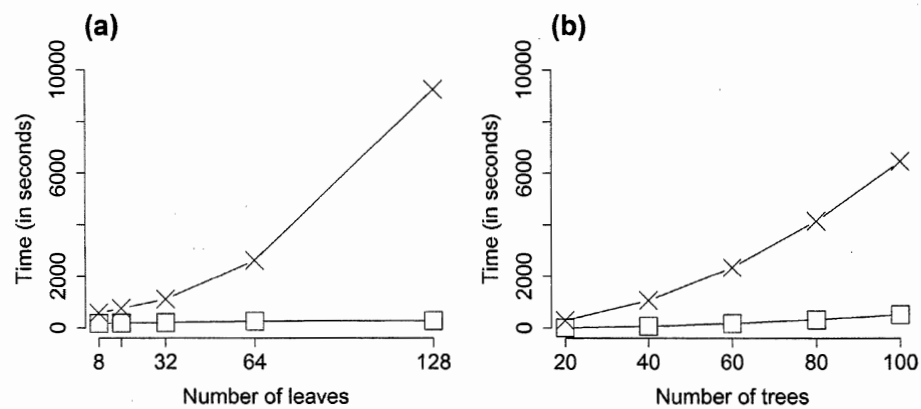


Figure 4.14 Comparaison de notre algorithme, basé sur l'indice de validité des clusters *E-CH* et la distance *RF* non mise au carré (□), avec à l'approche directe de Stockham *et al.* basée sur l'inférence d'arbre consensus et la distance *RF* quadratique

(×) (Stockham, Wang et Warnow, 2002). Cette comparaison a été faite, en mesurant le temps d'exécution des algorithmes, en fonction du nombre de feuilles dans les arbres phylogénétiques (a) et du nombre d'arbres phylogénétiques fournis en entrée (b).

4.7 Conclusion

Dans ce chapitre, nous avons décrit un nouvel algorithme basé sur les *k*-moyennes, qui partitionne un ensemble d'arbres phylogénétiques afin de retrouver un seul ou de multiples arbres consensus. Nous avons présenté un certain nombre de nouvelles formules permettant d'utiliser les indices de validité des clusters de Caliński-Harabasz (*CH*), Silhouette (*SH*), *Gap* et *W*, ainsi que la distance topologique de Robinson et Foulds (*RF*) dans le cadre du regroupement d'arbres. Nous avons constaté que l'utilisation de la distance *RF* non mise carré dans la fonction objective de l'algorithme des *k*-moyennes avec les indices *E-CH* et *W* conduit à des meilleures performances de partitionnement. De plus, nous avons montré que le nouvel algorithme est mieux adapté aux données manquantes et peut également être utilisé pour inférer des super-arbres uniques ou multiples.

Les résultats de nos simulations suggèrent que notre algorithme est beaucoup plus rapide que l'approche de regroupement d'arbres phylogénétiques proposée par Stockham *et al.* (Stockham, Wang et Warnow, 2002), tout en offrant de meilleures solutions de partitionnement dans la plupart des cas. Le nouvel algorithme peut être utilisé pour résoudre un certain nombre de problèmes importants en biologie computationnelle, tels que l'identification de gènes ayant des histoires évolutives similaires (*p.ex.* ceux qui ont subi les mêmes événements de transfert de gènes horizontaux) ou l'inférence de multiples sous-arbres consensus de l'arbre de vie (*Tree of Life*).

Dans les perspectives futures, il pourrait être pertinent de modifier la fonction objective principale de l'algorithme (voir l'équation 4.3) et d'améliorer notre procédure de traitement d'arbres définis sur des ensembles de feuilles différents, mais partiellement chevauchants (*c.-à-d.*, le problème de reconstruction de super-arbres multiples). Ces problématiques seront considérées en détail au chapitre V.

CHAPITRE V

UTILISATION DES BORNES DANS LE CADRE DES K-MOYENNES POUR IDENTIFIER LES ARBRES CONSENSUS ET LES SUPER-ARBRES MULTIPLES

«*Consensus (n,m) : Procédure qui consiste à dégager un accord sans procéder à un vote formel, ce qui évite de faire apparaître les objections et les abstentions.*», dictionnaire Larousse, 2016.

5.1 Préface

Dans ce chapitre, nous introduirons de nouveaux critères de partitionnement d'arbres phylogénétiques et étendrons notre approche aux cas des données manquantes. Au chapitre précédent, nous avons développé et présenté un nouvel algorithme de partitionnement d'arbres phylogénétiques en nous appuyant sur la méthode des *k*-moyennes (Lloyd, 1957; MacQueen, 1967; Bock, 2007; Celebi, Kingravi et Vela, 2013). Nous avons utilisé quatre indices de validité des clusters suivants : le critère de Caliński-Harabasz (*E-CH*) (Caliński et Harabasz, 1974), le critère Silhouette (*SH*) (Rousseeuw, 1987), le critère *Gap* (Tibshirani, Walther et Hastie, 2001) et le critère *W*. Pour chacun des deux indices les plus performants, en termes d'*ARI* moyen, à savoir le critère *E-CH* pour le cas des données hétérogènes et le critère *W* pour le cas des données homogènes, nous avons testé l'algorithme avec deux variantes de la distance topologique de Robinson et Foulds (*RF*) (Robinson et Foulds, 1981) : avec et

sans sa mise au carré. Les résultats de notre algorithme, obtenus en utilisant l'indice *E-CH*, se sont révélés meilleurs que ceux fournis par l'algorithme de Stockham *et al.* (2002) (voir les figures 4.13a et 4.13b). De plus, en termes de complexité temporelle, notre nouvel algorithme était de loin plus rapide que celui de Stockham *et al.* (voir les figures 4.14a et 4.14b). Dans ce chapitre, nous affinerons l'algorithme du chapitre IV en établissant deux bornes, inférieure et supérieure, pour le terme principal de notre fonction objective des *k*-moyennes. Nous introduirons la borne supérieure, la borne inférieure, ainsi que le milieu de l'intervalle entre ces bornes, de la fonction objective des *k*-moyennes, en nous inspirant de la fonction objective considérée au chapitre précédent. Quelques résultats théoriques intéressants seront établis. Nous considérerons également en détails dans ce chapitre la problématique d'inférence de super-arbres multiples.

5.2 Introduction

Au cours des dernières années, les études phylogénétiques comparatives abordent de plus en plus fréquemment l'évolution des traits dans un contexte multivarié, avec un nombre de taxons qui continue d'augmenter de façon spectaculaire (Christenhusz et Byng, 2016; Bennett, 2013). Les méthodes récentes pour les études phylogénétiques comparatives ont fourni des moyens d'intégrer l'erreur de mesure et de résoudre les problèmes de calcul. Cependant, les données manquantes demeurent un problème particulièrement fréquent. Ce problème laisse les chercheurs choisir entre l'omission d'observations et l'utilisation d'approches plus complexes pouvant traiter des données manquantes. C'est pourquoi nous avons choisi d'étendre notre approche et de considérer le cas des arbres phylogénétiques ayant des ensembles de feuilles différents, mais partiellement chevauchants. Nous proposerons donc un nouvel algorithme pour l'identification de super-arbres multiples, basé sur l'indice de validité des clusters - Caliński-Harabasz (*CH*) (Caliński et Harabasz, 1974), soit le meilleur critère selon nos simulations dans le cas des *k*-moyennes (voir le chapitre IV; notons

que nos simulations pour le cas des super-arbres décrites dans le chapitre IV s'appuyaient sur une méthode de reconstruction basique, qui ne prend pas en compte la connectivité des graphes à l'intérieur des clusters d'arbres - comme c'est fait dans le présent chapitre). Dans un premier temps, nous étudierons plus en détails la fonction objective des *k*-moyennes afin d'affiner nos résultats présentés dans le chapitre IV. À la section 5.3, nous démontrerons un théorème mettant en œuvre deux bornes (supérieure et inférieure) de la somme interne de la fonction objective des *k*-moyennes que nous considérons. À la suite de cette démonstration mathématique, nous mettrons en place trois variantes de la fonction objective (*c.-à-d.*, utilisant deux bornes et le milieu de l'intervalle entre ces bornes pour la fonction objective considérée). Dans un deuxième temps, nous comparerons notre approche des *k*-moyennes avec celle des *k*-médoïdes (voir le chapitre III). À la section 5.4, nous présenterons notre nouvel algorithme permettant de partitionner les super-arbres, en y incorporant l'algorithme de recherche en profondeur. À la section 5.5, nous présenterons nos simulations, en décrivant en détail les protocoles de génération d'arbres centroïdes, d'arbres phylogénétiques et d'ajout de bruit aux jeux de données simulées. À la section 5.6, nous présenterons les résultats des tests des différentes variantes de la fonction objective des *k*-moyennes en utilisant nos données simulées. À la section 5.7, nous discuterons des principaux concepts et présenterons quelques pistes pour le travail futur.

5.3 Méthodologie

L'algorithme des *k*-moyennes, comme nous l'avons décrit au chapitre IV, partitionne un ensemble de N objets en K groupes disjoints. Cependant, la fonction objective et les indices de validité des clusters classiques ne peuvent pas être utilisés lors du regroupement d'arbres phylogénétiques puisqu'ils ont été conçus pour fonctionner dans l'espace euclidien. Au chapitre précédent, nous avons montré que la distance topologique de Robinson et Foulds n'est pas une distance euclidienne. Nous avons

également décrit différents changements à apporter à l'algorithme conventionnel des *k*-moyennes afin de l'adapter au problème du regroupement d'arbres. Dans ce chapitre, nous mettrons en place deux bornes, ainsi que le milieu de l'intervalle entre ces bornes, pour la fonction objective des *k*-moyennes et pour le critère de validité des clusters de Caliński-Harabasz (*CH*) (Caliński et Harabasz, 1974).

5.3.1 Théorème définissant les bornes de la fonction objective des *k*-moyennes lors du partitionnement d'arbres phylogénétiques

Théorème 5.1 : *Pour un cluster donné, k, contenant N_k arbres phylogénétiques (c.-à-d., arbres additifs ou X-arbres), les inégalités suivantes sont conservées :*

$$\frac{1}{N_k - 1}\sum_{i=1}^{N_k-1}\sum_{j=i+1}^{N_k} RF(T_{ki},T_{kj}) \le \sum_{i=1}^{N_k} RF(C_k,T_{ki}) \le \frac{2}{N_k}\sum_{i=1}^{N_k-1}\sum_{j=i+1}^{N_k} RF(T_{ki},T_{kj}), \qquad (5.1)$$

où N_k est le nombre d'arbres phylogénétiques dans le cluster k, T_{ki} et T_{kj} sont respectivement les arbres i et j du cluster k et C_k est l'arbre consensus de la règle majoritaire du cluster k.

Preuve

Premièrement, la somme $\sum_{i=1}^{N_k} RF(C_k,T_{ki})$ peut être décomposée en une double somme comme suit :

$$\sum_{i=1}^{N_k} RF(C_k,T_{ki}) = \sum_{i=1}^{N_k-1}\sum_{j=i+1}^{N_k} \frac{1}{(N_k-1)}(RF(C_k,T_{ki}) + RF(C_k,T_{kj})). \qquad (5.2)$$

La distance de Robinson et Foulds est une métrique, qui satisfait donc l'inégalité triangulaire (Robinson et Foulds, 1981). Par conséquent, l'inégalité suivante s'applique à toute paire d'arbres phylogénétiques (i,j) : $RF(C_k,T_{ki}) + RF(C_k,T_{kj}) \ge RF(T_{ki},T_{kj})$.

Ce qui signifie que :

$$\sum_{i=1}^{N_k} RF(C_k,T_{ki}) \geq \frac{1}{(N_k-1)} \sum_{i=1}^{N_k-1} \sum_{j=i+1}^{N_k} RF(T_{ki},T_{kj}) . \qquad (5.3)$$

Deuxièmement, d'après la propriété, prouvée par Barthélemy et McMorris, montrant que l'arbre consensus majoritaire d'un ensemble d'arbres soit un arbre médian de cet ensemble au sens de la distance topologique de Robinson et Foulds (Barthélemy et McMorris, 1986), nous avons : $\sum_{i=1}^{N_k} RF(C_k,T_{ki}) = \underset{T \in T(n)}{Min} \sum_{i=1}^{N_k} RF(T,T_{ki})$, où $T(n)$ est l'ensemble de tous les arbres phylogénétiques possibles avec n feuilles.

Ainsi, nous obtenons :

$$\sum_{i=1}^{N_k} RF(C_k,T_{ki}) \leq \underset{1 \leq j \leq N_k}{Min} (\sum_{i=1}^{N_k} RF(T_{kj},T_{ki})) \leq \frac{1}{N_k} \sum_{i=1}^{N_k} \sum_{j=i}^{N_k} RF(T_{kj},T_{ki}) . \qquad (5.4)$$

Il est facile de voir que cette dernière somme est égale à : $\frac{2}{N_k} \sum_{i=1}^{N_k-1} \sum_{j=i+1}^{N_k} RF(T_{ki},T_{kj})$.

Mentionnons que le terme $\underset{1 \leq j \leq N_k}{Min} (\sum_{i=1}^{N_k} RF(T_{kj},T_{ki}))$ peut également être utilisé comme la borne supérieure de $\sum_{i=1}^{N_k} RF(C_k,T_{ki})$. □

Il convient de noter que les bornes inférieure et supérieure de $\sum_{i=1}^{N_k} RF(C_k,T_{ki})$ établies par le Théorème 5.1 sont égales quand $N_k = 2$.

5.3.2 Borne supérieure de la fonction objective des *k*-moyennes lors du partitionnement d'arbres phylogénétiques

Nous avons déjà constaté que l'inférence fréquente d'arbres consensus lors d'un algorithme de partitionnement d'arbres est très coûteuse en termes de complexité temporelle (voir le chapitre IV). Pour éviter d'inférer l'arbre consensus à chaque itération de l'algorithme des *k*-moyennes, nous avons élaboré une heuristique. En utilisant le Théorème 5.1, nous pourrions donc remplacer dans la fonction objective des *k*-moyennes la distance intragroupe du cluster *k*, à savoir $\sum_{i=1}^{N_k} RF(C_k, T_{ki})$, par sa borne supérieure : $\frac{2}{N_k} \sum_{i=1}^{N_k-1} \sum_{j=i+1}^{N_k} RF(T_{ki}, T_{kj})$.

5.3.3 Borne inférieure de la fonction objective des *k*-moyennes lors du partitionnement d'arbres phylogénétiques

Nous avons également établi une borne inférieure de la distance intragroupe du cluster *k* dans la fonction objective des *k*-moyennes. En utilisant le Théorème 5.1, nous pourrions donc remplacer dans la fonction objective des *k*-moyennes la distance intragroupe du cluster *k* par sa borne inférieure : $\frac{1}{N_k - 1} \sum_{i=1}^{N_k-1} \sum_{j=i+1}^{N_k} RF(T_{ki}, T_{kj})$.

5.3.4 Milieu de l'intervalle entre les bornes inférieure et supérieure de la fonction objective des *k*-moyennes lors du partitionnement d'arbres phylogénétiques

Maintenant que nous avons établi les deux bornes, inférieure et supérieure pour la distance intragroupe, nous pouvons aisément en déduire le milieu de intervalle qu'elles délimitent, par : $\frac{3N_k - 2}{2N_k(N_k - 1)} \sum_{i=1}^{N_k-1} \sum_{j=i+1}^{N_k} RF(T_{ki}, T_{kj})$. Cette expression pourrait aussi être utilisée comme une approximation de la contribution du cluster *k* à la

fonction objective de l'algorithme des *k*-moyennes pour le partitionnement d'arbres (voir l'équation 4.2 pour la fonction objective originale).

5.3.5 Différentes versions de la fonction objective des *k*-moyennes lors du partitionnement d'arbres phylogénétiques

Afin d'accélérer le processus de regroupement d'arbres, nous avons proposé d'utiliser la fonction objective OF_{EA} suivante (voir l'équation 5.5, ou l'équation 4.3 que nous répétons ici à cause de son importance), qui peut être considérée comme une approximation euclidienne de la fonction objective originale *OF*, définie par l'équation 4.2 :

$$OF_{EA} = SS_W = \sum_{k=1}^{K} \frac{1}{N_k} \sum_{i=1}^{N_k-1} \sum_{j=i+1}^{N_k} RF(T_{ki}, T_{kj}) . \tag{5.5}$$

Comme nous l'avons montré dans le chapitre IV, la fonction objective OF_{EA} est seulement une approximation de la fonction objective définie par l'équation 4.2. Comme nous le verrons plus loin dans ce chapitre, cette approximation conduit à de très bons résultats lors du regroupement d'arbres phylogénétiques.

La fonction objective suivante basée sur le milieu de l'intervalle entre les bornes supérieure et inférieure établie dans le Théorème 5.1 peut aussi être utilisée comme une approximation de la fonction d'objective originale *OF*, définie par l'équation 4.2 :

$$OF_{MA} = \sum_{k=1}^{K} \frac{3N_k - 2}{2N_k(N_k - 1)} \sum_{i=1}^{N_k-1} \sum_{j=i+1}^{N_k} RF(T_{ki}, T_{kj}) . \tag{5.6}$$

De manière similaire, nous pouvons définir la fonction objective en utilisant la borne inférieure (*c.-à-d.*, *the lower bound*) de *OF* comme suit :

$$OF_{LA} = \sum_{k=1}^{K} \frac{1}{N_k - 1} \sum_{i=1}^{N_k - 1} \sum_{j=i+1}^{N_k} RF(T_{ki}, T_{kj}) . \quad (5.7)$$

et la borne supérieure (*c.-à-d.*, *the upper bound*) de *OF* comme suit :

$$OF_{UA} = \sum_{k=1}^{K} \frac{2}{N_k} \sum_{i=1}^{N_k - 1} \sum_{j=i+1}^{N_k} RF(T_{ki}, T_{kj}) . \quad (5.8)$$

Nous pouvons donc accélérer le processus de partitionnement d'arbres en utilisant les fonctions objectives approximatives définies dans les équations 5.6 à 5.8 (de même que l'équation 5.5). Le principal avantage d'utiliser les fonctions OF_{MA}, OF_{LA} ou OF_{UA} (de même que la fonction OF_{EA}, comme nous l'avons vu au chapitre IV) est que nous n'avons pas à recalculer les arbres consensus pour les deux groupes impliqués dans chaque opération d'échange d'éléments dans l'algorithme des *k*-moyennes. En effet, si les fonctions approximatives décrites sont utilisées, la complexité temporelle d'une telle opération d'échange sera seulement $O(N)$ parce que toutes les distances *RF* entre tous les arbres dans l'ensemble d'arbres d'entrée Π peuvent être précalculées en $O(n \times N^2)$ (ici N est le nombre d'arbres d'entrée et n est le nombre de feuilles dans ces arbres). Cela mènera à la complexité temporelle globale de $O(n \times N^2 + K \times N \times I)$ pour notre algorithme des *k*-moyennes de partitionnement d'arbres, où I est le nombre d'itérations dans la boucle principale des *k*-moyennes.

5.3.5.1 L'indice de validité des clusters de Caliński-Harabasz adapté au partitionnement d'arbres : cas de la fonction approximative OF_{MA}

L'indice de validité des clusters que nous considérons ici est l'indice de Caliński-Harabasz (Caliński et Harabasz, 1974). Ce critère a été décrit en détail aux sections 3.3.4 et 4.3.5 dans les chapitres III et IV. L'utilisation de la fonction approximative dans le cas du milieu de l'intervalle entre les bornes (voir la section 5.3.5) devrait

impliquer les changements appropriés dans les formules des indices de validité des clusters considérés.

Dans le cas du milieu de l'intervalle, l'approximation de l'indice d'évaluation intragroupe $SS_W(OF_{MA})$, doit être calculée comme suit :

$$SS_W(OF_{MA}) = \sum_{k=1}^{K} \frac{3N_k - 2}{2N_k(N_k - 1)} \sum_{i=1}^{N_k-1} \sum_{j=i+1}^{N_k} RF(T_{ki}, T_{kj}) . \tag{5.9}$$

De plus, la nouvelle variance globale entre tous les groupes, *c.-à-d.*, l'indice d'évaluation intergroupe $SS_B(OF_{MA})$, doit être calculée comme suit :

$$SS_B(OF_{MA}) = \frac{3N - 2}{2N(N-1)} (\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} RF_{norm}(T_i, T_j)) - SS_W(OF_{MA}) , \tag{5.10}$$

où $SS_W(OF_{MA})$ est calculé selon la formule 5.9.

Évidemment, les formules similaires définissant les coefficients $SS_W(OF_{LA})$, $SS_B(OF_{LA})$, $SS_W(OF_{UA})$ et $SS_B(OF_{UA})$ ont aussi été utilisées dans nos simulations.

Il est important de noter que la fonction objective euclidienne, OF_{EA}, et la fonction objective de borne supérieure, OF_{UA}, définies par les équations 5.5 et 5.8, respectivement, ne diffèrent que par un facteur multiplicatif constant. Cela signifie qu'elles atteignent leur minimum au même point (*c.-à-d.*, avec le même regroupement d'arbres). Par conséquent, seule une de ces fonctions (*c.-à-d.*, OF_{EA}) a été utilisée dans nos simulations comparatives avec les fonctions objectives OF_{LA} et OF_{MA}.

5.3.6 L'algorithme des *k*-moyennes étendu pour le cas d'une fonction objective approximative

La procédure des *k*-moyennes adaptée au partitionnement d'arbres phylogénétiques et basée sur une des fonctions objectives approximatives définies dans ce chapitre (les fonctions OF_{MA}, OF_{LA}, OF_{UA} (ou OF_{EA})) est présentée dans l'Algorithme 5.1.

Algorithme 5.1 des *k*-moyennes adapté au partitionnement d'arbres phylogénétiques et basé sur une fonction objective approximative

---

Définir la fonction objective approximative à utiliser: $OF_{appr} = OF_{MA}$, ou OF_{LA}, ou OF_{UA} (ou OF_{EA}).

Étape 1. Partitionner aléatoirement les *N* arbres phylogénétiques donnés en *K* clusters

Calculer OF_{appr}_initiale à l'aide de l'équation (5.5, 5.6, 5.7 ou 5.8)

OF_{appr}_meilleure = OF_{appr}_initiale

Étape 2. Pour chaque arbre phylogénétique *tree* **faire**

Pour chaque cluster *groupe* **faire**

Déplacer *tree* dans *groupe*

Si (OF_{appr} (calculé à l'aide de l'équation (5.5, 5.6, 5.7 ou 5.8)) < OF_{appr}_meilleure) **alors**

Accepter le déplacement de *tree* dans *groupe*

OF_{appr}_meilleure = OF_{appr}

Sinon

Annuler le déplacement de *tree*.

Fin Si

Répéter l'étape 2 jusqu'à la convergence qui sera atteinte soit lorsqu'il n'y aura plus de modifications dans les *k* groupes d'arbres formés, soit lorsqu'on aura atteint le nombre maximum d'itérations *I* (seuil préfixé).

L'algorithme 5.1 sera exécuté plusieurs fois avec les partitionnements de départ aléatoires. Le meilleur partitionnement sera ensuite choisi en fonction de la valeur de

l'indice de validité des clusters $CH = \frac{SS_B}{SS_W} \times \frac{N-K}{K-1}$ (voir, par exemple, les formules pour 5.9 et 5.10 pour SS_W et SS_B, respectivement, pour la fonction objective OF_{MA}).

5.3.7 Classification d'arbres phylogénétiques avec des ensembles de feuilles différents - l'approche de super-arbres

De nombreuses études impliquant des arbres phylogénétiques requièrent la fusion des résultats de différents ensembles de taxons, comme par exemple le projet "*Tree of Life*" (Letunic et Bork, 2006; Letunic et Bork, 2016). La dernière étape de ce projet, consistant en la fusion d'arbres ayant des ensembles de taxons différents mais partiellement chevauchants, est plus complexe que l'étape de construction d'arbres consensus.

Rappelons que la règle majoritaire a été introduite par Margush et McMorris (1981) pour construire un arbre consensus à partir d'arbres définis sur un même ensemble de feuilles. Cette méthode retourne un arbre unique qui correspond aux bipartitions contenues dans plus de 50% des arbres d'entrée. Ainsi, les bipartitions majoritaires de l'arbre consensus peuvent être déterminées simplement en comptant la fréquence des occurrences des bipartitions dans les arbres d'entrée. Goloboff et Pol (2002) ont exprimé des doutes sur la possibilité de développer une méthode équivalente lorsque les arbres d'entrée sont définis sur des ensembles de feuilles différents, mais partiellement chevauchants. En effet, dans ce cas, il n'est pas toujours possible de déterminer combien d'arbres supportent (ou contredisent) une bipartition complète. Il n'est pas difficile de compter combien de fois une bipartition incomplète donnée apparaît dans les arbres d'entrée, mais cette statistique ne peut pas être utilisée directement pour inférer les bipartitions complètes majoritaires. Nous devons donc utiliser un critère différent pour définir *un super-arbre majoritaire.*

Soit Π un ensemble de N arbres phylogénétiques non enracinés contenant des ensembles de feuilles étiquetées qui se chevauchent. Dans ce cas, la fonction objective originale des k-moyennes (voir l'équation 4.2) peut être reformulée comme suit :

$$OF_{ST} = \sum_{k=1}^{K}\sum_{i=1}^{N_k} RF_{norm}(ST_k, T_{ki}) = \sum_{k=1}^{K}\sum_{i=1}^{N_k}\left(\frac{RF(ST_k, T_{ki})}{2n(ST_k, T_{ki}) - 6}\right), \tag{5.11}$$

où K est le nombre de clusters, N_k est le nombre d'arbres phylogénétiques dans le cluster k, $RF_{norm}(ST_k, T_{ki})$ est la distance topologique de Robinson et Foulds normalisée entre le super-arbre de la règle majoritaire ST_k (Cotton et Wilkinson, 2007) du cluster k et l'arbre i de ce même cluster, dénoté par T_{ki}, qui sont réduits en sous-arbres maximums ayant les mêmes ensembles de feuilles. Les versions réduites des arbres ST_k et T_{ki} sont obtenues après la suppression de toutes les feuilles appartenant uniquement à l'un de ces deux arbres. La distance de Robinson et Foulds dans cette formule est normalisée par sa valeur maximale possible, qui est $2n(ST_k, T_{ki}) - 6$, où $n(ST_k, T_{ki})$ est le nombre de feuilles communes dans les arbres ST_k et T_{ki}. L'équation 5.11 est valable si le nombre de feuilles communes dans ST_k et T_{ki} est plus grand que trois.

De même, la fonction d'approximation euclidienne définie par l'équation 5.5 peut être réécrite comme suit dans le cas de super-arbres :

$$OF_{ST\_EA} = \sum_{k=1}^{K}\frac{1}{N_k}\sum_{i=1}^{N_k-1}\sum_{j=i+1}^{N_k} RF_{norm}(T_{ki}, T_{kj}) = \sum_{k=1}^{K}\frac{1}{N_k}\sum_{i=1}^{N_k-1}\sum_{j=i+1}^{N_k}\left(\frac{RF(T_{ki}, T_{kj})}{2n(T_{ki}, T_{kj}) - 6}\right). \tag{5.12}$$

Cette équation implique également que deux arbres appartenant au même cluster doivent avoir au moins quatre feuilles communes, mais un seuil de similarité plus élevé peut être utilisé comme paramètre de la méthode afin d'augmenter l'homogénéité des clusters.

Les autres fonctions objectives considérées (voir les équations 5.6-5.8) et les indices de validité des clusters correspondants doivent être normalisés d'une manière similaire. Une fois que les clusters d'arbres sont formés par notre algorithme, toute méthode de reconstruction de super-arbres existante (Bininda-Emonds, 2004) pourrait être appliquée pour inférer les super-arbres des clusters.

Dans le cas des arbres avec des ensembles chevauchants de feuilles, la somme intragroupe SS_W, de l'équation 4.6 peut être réécrite comme suit :

$$SS_W = \sum_{k=1}^{K} \frac{1}{N_k} \sum_{i=1}^{N_k-1} \sum_{j=i+1}^{N_k} \left(\frac{RF(T_{ki}, T_{kj})}{2n(T_{ki}, T_{kj}) - 6} \right), \tag{5.13}$$

alors que la somme intergroupe SS_B, de l'équation 4.7 peut être réécrite comme suit :

$$SS_B = \frac{1}{N} \left(\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \left(\frac{RF(T_{ki}, T_{kj})}{2n(T_{ki}, T_{kj}) - 6} \right) \right) - SS_W . \tag{5.14}$$

Ces deux formules correspondent bien évidemment à notre approximation euclidienne.

Nous avons utilisé certaines méthodes de la théorie des graphes pour résoudre le problème de partitionnement de super-arbres. En particulier, l'algorithme de recherche en profondeur a été employé pour savoir si un graphe donné est connexe ou non. Nous avons inséré l'algorithme de recherche en profondeur dans trois endroits dans notre algorithme de construction de super-arbres multiples (voir l'Algorithme 5.2).

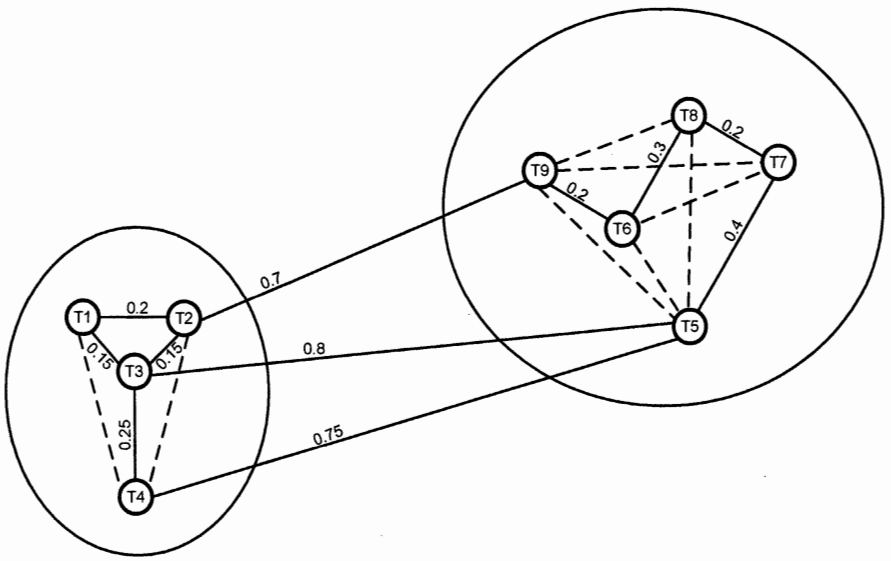


Figure 5.1 Deux clusters de quatre et de cinq arbres phylogénétiques, respectivement, analysés dans l'algorithme de partitionnement de super-arbres. Les cercles représentent les clusters. Chaque nœud représente un arbre phylogénétique binaire. Une arrête pleine entre deux nœuds indique que les arbres phylogénétiques correspondants ont plus de trois taxons communs, alors qu'une arête en pointillé entre deux nœuds indique que les arbres phylogénétiques correspondants ont trois ou moins de taxons communs. Les valeurs sur les arêtes indiquent les distances topologiques de Robinson et Foulds normalisées entre les arbres phylogénétiques reliés. *Notons que certaines arêtes en pointillés n'ont pas été illustrées afin de faciliter la compréhension de la figure.*

Dans notre algorithme de partitionnement de super-arbres, nous avons utilisé des graphes non orientés définis comme suit : chaque nœud d'un graphe représentait un arbre phylogénétique et les arêtes pleines reliaient deux arbres phylogénétiques si et seulement si ces deux arbres avaient au moins quatre taxons en commun (voir la figure 5.1). La vérification que le graphe représentant un cluster donné soit connexe intervenait dans les trois situations suivantes dans notre algorithme :

La *première situation* est le partitionnement initial des N arbres phylogénétiques en K clusters. Cette étape est cruciale puisque nous devons nous assurer que tous les K graphes initiaux, résultant du partitionnement des données en K clusters, sont connexes. Nous ne pouvons pas inférer les super-arbres des graphes non connexes.

La *seconde situation* est de placer correctement chaque arbre phylogénétique dans son propre cluster. Si le cluster source, après le retrait d'un arbre donné, est toujours connexe, alors le déplacement correspondant peut être validé, sinon ce déplacement doit être refusé. De même, si le cluster de destination est toujours connexe après l'ajout d'un nouvel arbre, alors le déplacement est validé et dans le cas contraire, le déplacement est annulé. Il faut que les deux conditions suivantes soient réunies : cluster source et cluster destination connexes, pour pouvoir accepter un déplacement d'un arbre d'un cluster à un autre. Par la suite, nous recalculons la valeur de la fonction objective pour chaque nouveau partitionnement d'arbres. Nous conservons en mémoire le partitionnement avec le meilleur score de la fonction objective choisie.

Enfin, la *troisième situation* est l'étape de fusion des deux clusters. Nous devons fusionner uniquement deux clusters dont la fusion donnera un graphe connexe.

5.4 Algorithme de construction de super-arbres multiples

Dans cette section, nous décrivons l'algorithme de partitionnement d'arbres qui permet d'obtenir des super-arbres multiples. Nous indiquons les différents emplacements où nous avons incorporé l'algorithme de recherche en profondeur comme : Situation 1, Situation 2 et Situation 3 dans l'algorithme 5.2 ci-dessous.

Algorithme 5.2 des *k*-moyennes pour la construction de super-arbres multiples.

Entrée

- Ensemble de N arbres phylogénétiques $\Pi = \{T_1, ..., T_N\}$ définis sur les ensembles chevauchants de feuilles.
- L'indice de validité des clusters *CH* (défini par les équations 5.13 et 5.14)
- Le nombre de partitionnements de départ aléatoires (*c.-à-d.*, *random starts*) R
- Les valeurs minimale et maximale pour le nombre de clusters (K_{MIN} et K_{MAX})

Sortie

- Le nombre optimal de clusters K_{opt}
- Le score de l'indice de validité des clusters *CH*
- La partition optimale trouvée P_{tr}

Procédure

Pour $k = K_{MAX}$ à K_{MIN} **faire**

Répéter R fois

Choisir k arbres centroïdes pris au hasard.

Assigner chaque arbre phylogénétique de Π au centroïde le plus proche.

Initialisation de la partition initiale (P_k).

Situation 1 : Appel de l'algorithme de recherche en profondeur.

Vérifier que tous les graphes formés autour des centroïdes soient connexes (*c.-à-d.*, toutes les paires d'arbres appartenant à ces clusters ont au moins quatre taxons en commun).

Sinon recommencer l'assignation.

$meilleure\_OF_{ST\_EA} = OF_{ST\_EA}$ pour la partition initiale (voir l'équation 5.12)

Four chaque *arbre* dans Π **faire**

Initialiser *cluster_arbre* = numéro du cluster d'*arbre*

Pour $cluster \leq k$ et ($cluster \neq cluster\_arbre$) **faire**

Déplacer *arbre* dans *cluster*

Situation 2 : Appel de l'algorithme de recherche en profondeur.

Si tous les graphes formés sont connexes **alors**

Calculer la valeur de la fonction objective OF_{ST_EA} (voir l'équation 5.12) pour le nouveau partitionnement d'arbres

Si ($OF_{ST\_EA} < meilleure\_OF_{ST\_EA}$) **alors**

P_k = Conserver la nouvelle partition

$meilleure\_OF_{ST\_EA} = OF_{ST\_EA}$

Sinon

Annuler le déplacement

Sinon

Annuler le déplacement

$cluster = cluster + 1$

Fin Pour

Fin Pour chaque

En utilisant les équations 5.13 et 5.14, calculer *CH* pour le nouveau partitionnement d'arbres P_k

Fin Répéter

Fusionner deux clusters dont les arbres forment un graphe connexe (voir la figure 5.1) une fois réunis ensemble.

Situation 3 : Appel de l'algorithme de recherche en profondeur.

Tant que tous les graphes formés ne sont pas connexes **faire**

Fusionner deux autres clusters

Fin Tant que

Fin Pour

Trouver la valeur optimale de l'indice de validité des clusters $CH(k)$ pour k variant entre K_{MIN} et K_{MAX}; K_{opt} va correspondre à la valeur maximale de $CH(k)$; P_{tr} est choisi en fonction de K_{opt}.

L'algorithme 5.2 prend en entrée un ensemble de N arbres phylogénétiques, l'indice de validité des clusters *CH* - adapté au partitionnement des super-arbres, le nombre de partitionnements de départ aléatoires (noté R) (par défaut c'est 100 départs) et enfin les valeurs minimale et maximale du nombre de clusters (notées K_{MIN} et K_{MAX}, respectivement). L'algorithme retourne en sortie le nombre optimal de clusters (noté K_{opt}), le meilleur score de l'indice de validité des clusters *CH* et la solution de partitionnement optimale (notée P_{tr}) pour l'ensemble de données d'entrée.

5.5 Description des données synthétiques utilisées dans les simulations

Dans cette section, nous décrivons notre protocole de simulations, ainsi que les données synthétiques utilisées. Nous avons d'abord comparé notre nouvel algorithme de partitionnement d'arbres phylogénétiques basé sur les k-médoïdes (voir le chapitre III) à l'algorithme basé sur les différentes versions de la fonction objective des k-moyennes décrites dans ce chapitre, en utilisant le protocole de simulations suivant qui comprend trois étapes principales et une étape spécifique aux cas des super-arbres.

5.5.1 Génération d'arbres phylogénétiques représentant les centroïdes

Au cours de la *première étape*, nous avons généré au hasard K arbres phylogénétiques T_1, ..., T_K (correspondant aux K arbres consensus, ou centroïdes, des K clusters) avec n feuilles chacun, où $K = 1, ..., 5$ et $n = 8, 16, 32, 64$ et 128.

5.5.2 Génération d'arbres phylogénétiques de chaque cluster d'arbres

Dans la *deuxième étape*, pour chaque arbre phylogénétique centroïde T_i ($i = 2, ..., K$) obtenu à la première étape, nous avons généré de façon aléatoire un ensemble de 100 arbres correspondant au cluster i en utilisant les quatre intervalles de bruit définis dans la sous-section 5.5.3.

5.5.3 Intervalles de bruit pour les jeux de données simulées

Chaque élément T du cluster i était un arbre phylogénétique, tel que le pourcentage de similitude (mesuré en utilisant la distance RF normalisée) entre T et T_i variait :

- de 0% à 10% (niveau de bruit de 10%),
- de 10% à 25% (niveau de bruit de 25%),
- de 25% à 50% (niveau de bruit de 50%),
- de 50% à 75% (niveau de bruit de 75%).

Les résultats de ces simulations sont présentés sur les figures 5.2 et 5.3.

5.5.4 Simulations spécifiques aux cas des super-arbres

Mentionnons que nos simulations pour le cas des super-arbres décrites dans le chapitre IV s'appuyaient sur une méthode de reconstruction de super-arbres, qui ne prenait pas en compte la connectivité des graphes à l'intérieur des clusters d'arbres (voir la figure 5.1). Afin de tester la robustesse de l'algorithme 5.2 présenté ci-dessus, nous avons supprimé aléatoirement des feuilles dans les arbres phylogénétiques issus de nos données d'entrée (*c.-à-d.*, les arbres obtenus à la suite des étapes précédentes). Le pourcentage de feuilles supprimées était respectivement 10%, 25% et 50% de l'ensemble des feuilles de l'arbre phylogénétique source. À la suite des deux premières étapes de génération de données (voir les sections 5.5.1 et 5.5.2), nous avons ajouté une étape consistant à enlever au hasard un certain pourcentage de feuilles dans les arbres générés. Nous avons alors évalué la capacité de notre algorithme à résoudre le problème des données manquantes, *c.-à-d.*, le problème de construction de super-arbres, qui est une problématique courante en biologie

computationnelle moderne. Les résultats de cette simulation sont présentés sur la figure 5.4.

5.5.5 Les algorithmes testés

Les algorithmes des *k*-moyennes et des *k*-médoïdes ont été exécutés avec 100 départs aléatoires et 100 itérations dans leur boucle interne. Différentes versions de la fonction objective des *k*-moyennes décrites dans ce chapitre ont été testées.

5.6 Résultats

Dans cette section, nous présentons les résultats de nos simulations. Pour ces simulations, le vrai nombre de clusters, *K*, était supposé être connu.

Nous avons testé notre nouvel algorithme de partitionnement d'arbres pour le regroupement : 1) d'arbres consensus lorsque tous les arbres de départ étaient définis sur un même ensemble de feuilles et 2) de super-arbres lorsque les arbres de départ étaient définis sur des ensembles de feuilles différents, mais partiellement chevauchants. Les résultats présentés sont les moyennes calculées pour toutes les combinaisons de nos paramètres (voir la section 5.5), 100 ensembles de données différents ont été générés pour chaque combinaison de paramètres, *c.-à-d.*, le nombre de clusters (voir les figures 5.2a et 5.2d), le nombre de feuilles (voir la figure 5.2b) et le niveau de bruit (voir la figure 5.2c). Nous avons comparé les six fonctions objectives suivantes :

1) la fonction objective euclidienne OF_{EA} (voir l'équation 5.5 ou l'équation 4.3) utilisant le critère de Caliński-Harabasz (*CH*) (*c.-à-d.*, son approximation notée *E-CH*),

2) la fonction objective OF_{LA} (voir l'équation 5.7) utilisant l'approximation par la borne inférieure et le critère de Caliński-Harabasz (noté *LB-CH*),

3) la fonction objective OF_{MA} (voir l'équation 5.6) utilisant l'approximation par le milieu de l'intervalle et le critère de Caliński-Harabasz (noté *MI-CH*),

4) la fonction objective euclidienne OF_{EA} (voir l'équation 5.5 ou l'équation 4.3) utilisant le critère *W*,

5) la fonction objective euclidienne OF_{EA} (voir l'équation 5.5 ou l'équation 4.3) utilisant le critère *Gap*,

6) la fonction objective euclidienne OF_{EA} (voir l'équation 5.5 ou l'équation 4.3) utilisant le critère Silhouette (*SH*).

En observant la figure 5.2, nous pouvons remarquer que les valeurs de l'*ARI* varient beaucoup en fonction du nombre de clusters pour les critères détectant les cas d'ensembles de données homogènes (*c.-à-d.*, les critères *W* et *Gap*, voir la figure 5.2a). Les résultats se stabilisent pour les autres indices de validité des clusters (quand le nombre de clusters $K > 1$). En outre, la stratégie de *E-CH*, basée sur la fonction approximative euclidienne et l'indice *CH*, dépasse clairement les trois autres stratégies concurrentes indépendamment du nombre de clusters pour des ensembles de données hétérogènes, tandis que la stratégie basée sur le critère *W* surpasse le critère *Gap* pour des ensembles de données homogènes (voir la figure 5.2a).

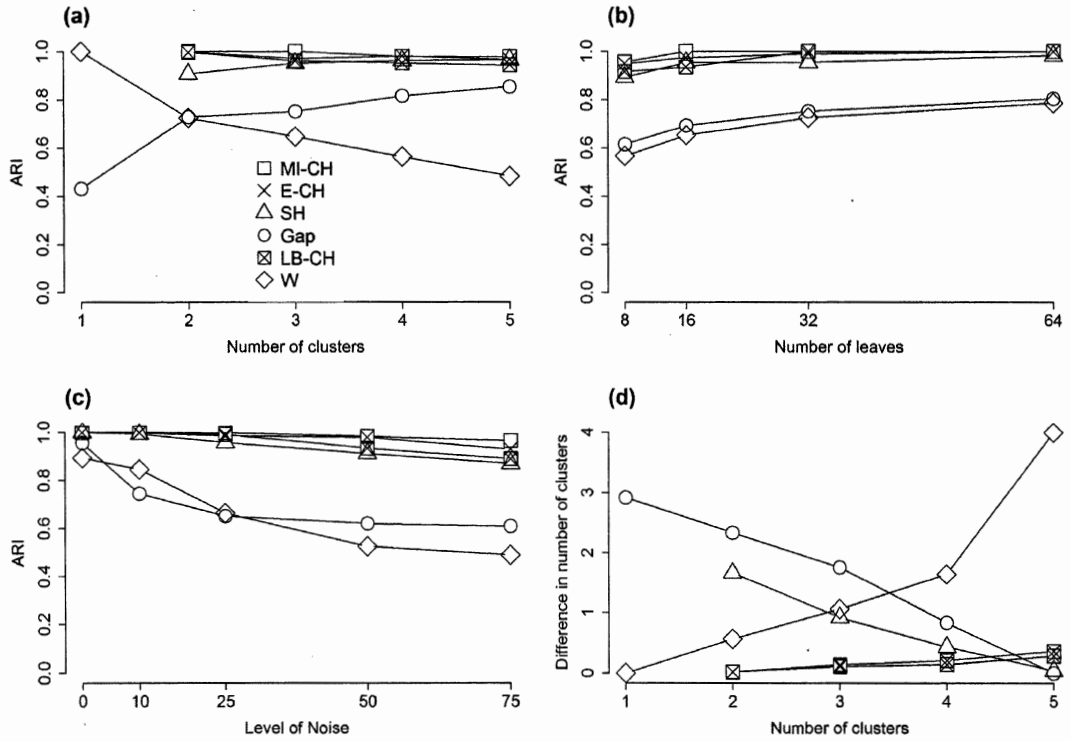


Figure 5.2 Évolution de l'*ARI* moyen en fonction : (a) du nombre de clusters, (b) du nombre de feuilles dans l'arbre phylogénétique, et (c) du niveau de bruit, quand le nombre de clusters, *K*, varie entre 1 et 5. Le graphique (d) représente la moyenne des différences absolues entre le nombre de clusters trouvés par l'algorithme et le nombre de clusters générés, pour *K* variant entre 1 et 5. Les 6 fonctions objective suivantes ont été comparées : 1) la fonction objective euclidienne OF_{EA} (voir l'équation 5.5 ou l'équation 4.3) utilisant le critère de Caliński-Harabasz (*CH*) (*c.-à-d.*, son approximation notée *E-CH*), 2) la fonction objective OF_{LA} (voir l'équation 5.7) utilisant l'approximation par la borne inférieure et le critère de Caliński-Harabasz (noté *LB-CH*), 3) la fonction objective OF_{MA} (voir l'équation 5.6) utilisant l'approximation par le milieu de l'intervalle et le critère de Caliński-Harabasz (noté *MI-CH*), 4) la fonction objective euclidienne OF_{EA} (voir l'équation 5.5 ou l'équation 4.3) utilisant le critère *W*, 5) la fonction objective euclidienne OF_{EA} (voir l'équation

5.5 ou l'équation 4.3) utilisant le critère *Gap* et 6) la fonction objective euclidienne OF_{EA} (voir l'équation 5.5 ou l'équation 4.3) utilisant le critère Silhouette (*SH*).

La figure 5.2b montre une légère augmentation des valeurs de l'*ARI* moyen quand le nombre de feuilles augmente. Encore une fois, les valeurs de l'*ARI* moyen pour le critère *E-CH* sont plus élevées que pour les autres critères. Les valeurs du critère *W* sont supérieures à celles du critère *Gap*.

La figure 5.2c montre que l'indice *W* est plus affecté par le bruit que le critère *Gap* lors du regroupement d'arbres et que le critère *E-CH* est plus stable que tous les autres critères. Par conséquent, notre simulation suivante (voir la figure 5.3) a été réalisée avec l'algorithme basé sur l'indice *E-CH* et la distance de Robinson et Foulds non mise au carré.

Les résultats de cette simulation comparative sont présentés sur la figure 5.3. Les courbes illustrées sur les figures 5.3 (a et b) indiquent que le nouvel algorithme basé sur la fonction objective euclidienne, OF_{EA}, qui est équivalente à la fonction objective de la borne supérieure, OF_{UA}, (voir les équations 5.5 et 5.8) dépasse largement l'approche directe de partitionnement d'arbres par les *k*-moyennes de Stockham *et al.* (Stockham, Wang et Warnow, 2002), ainsi que notre algorithme de partitionnement d'arbres basé sur les *k*-médoïdes (voir le chapitre III), en termes de qualité des regroupements obtenus. De plus, nos algorithmes des *k*-moyennes et des *k*-médoïdes étaient de loin les meilleures méthodes en termes de temps d'exécution pour les deux paramètres de simulation considérés, *c.-à-d.*, le nombre de feuilles (voir la figure 5.2c) et le nombre d'arbres (voir la figure 5.2d).

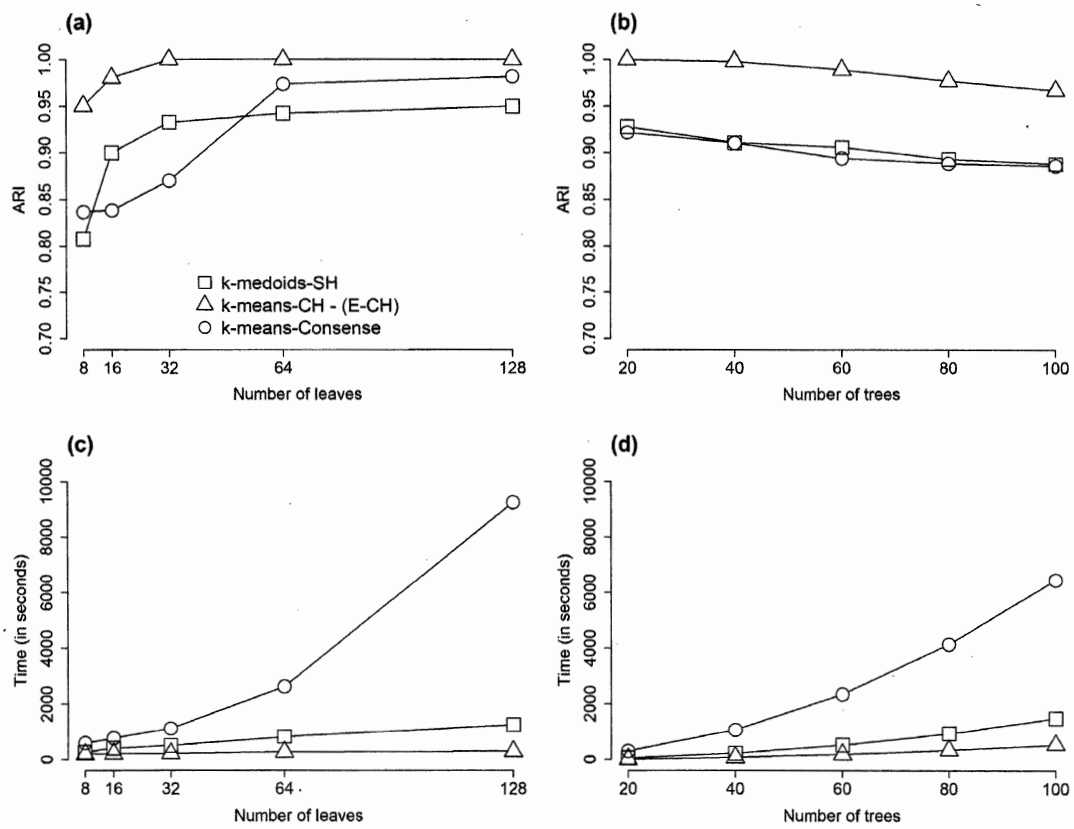


Figure 5.3 Comparaison des algorithmes de partitionnement : (Δ) notre algorithme basé sur le partitionnement par les *k*-moyennes avec la distance *RF* normalisée (sans la mettre au carré), (○) l'algorithme des *k*-moyennes (avec la distance *RF* mise au carré) réalisant l'inférence des arbres consensus majoritaires (à l'aide du programme Consense) à chaque étape dans l'algorithme des *k*-moyennes, implémenté selon Stockham *et al.* (Stockham, Wang et Warnow, 2002), et (□) notre algorithme basé sur les *k*-médoïdes et l'indice de validité des clusters *SH* (voir le chapitre III) La comparaison a été faite en termes d'*ARI* moyen (panneaux a et b) et de temps d'exécution (mesuré en secondes) fournis par les algorithmes (panneaux c et d) en fonction du nombre de feuilles et du nombre d'arbres.

La figure 5.4 montre les performances de notre algorithme des *k*-moyennes basé sur le critère *CH* adapté au problème de partitionnement de super-arbres (voir

l'algorithme 5.2) appliqué aux données affectées par l'élimination aléatoire des feuilles dans les arbres phylogénétiques d'entrée (voir la section 5.5.4).

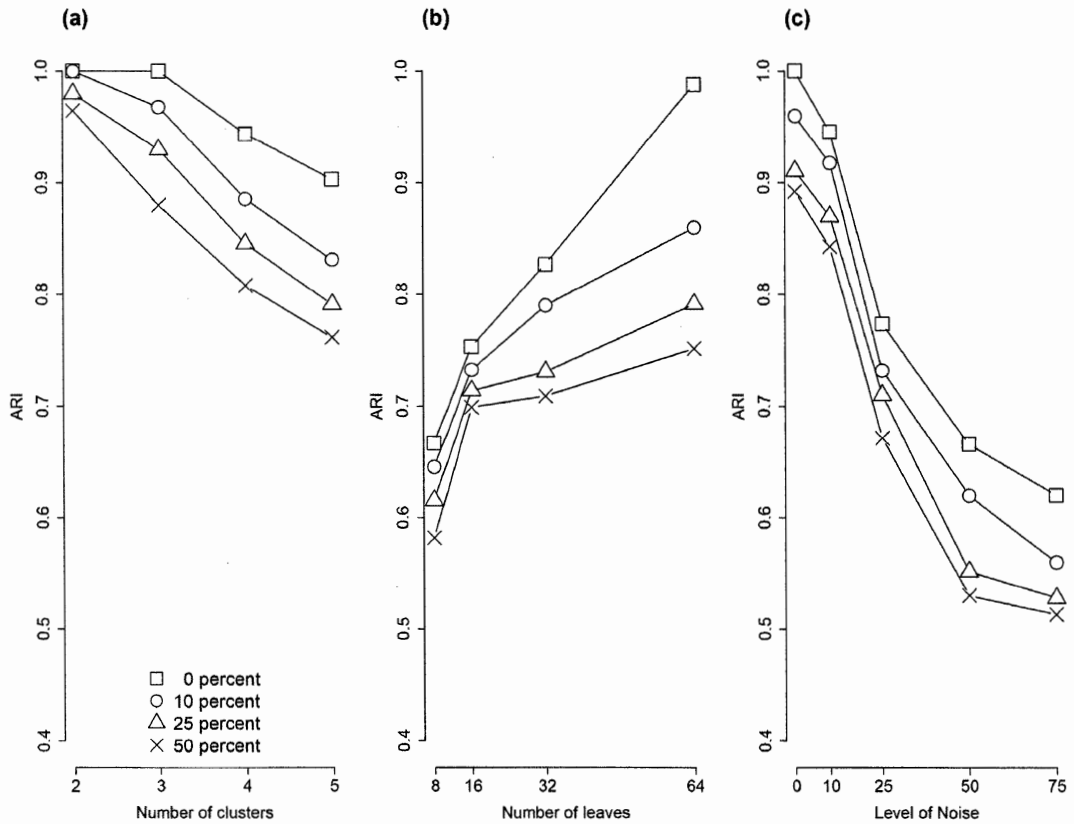


Figure 5.4 Les performances de notre algorithme de construction de super-arbres multiples basé sur l'indice de validité des clusters *CH*, adapté à l'approche de super-arbres et utilisant la distance *RF* non mise carré (voir l'algorithme 5.2), en fonction du : (a) nombre de clusters, (b) nombre de feuilles dans l'arbre phylogénétique, et (c) niveau de bruit, quand le nombre de clusters, *K*, varie entre 2 et 5. Pour chaque cas, nous avons fait varier le pourcentage de feuilles enlevées de l'arbre phylogénétique, avec 0% (□), 10% (○), 25% (Δ) et 50% (×) des feuilles qui ont été retirées des arbres.

La figure 5.4a présente la variation de l'*ARI* moyen en fonction du nombre de clusters. Ces résultats montrent que la performance en termes d'*ARI* moyen baisse

considérablement quand le nombre de clusters augmente. L'*ARI* moyen baisse également avec l'augmentation du pourcentage des feuilles enlevées.

Par exemple, le score de l'*ARI* moyen pour 5 clusters est égal à 0.51 lorsque 50% des feuilles ont été enlevées des arbres phylogénétiques, alors qu'il est égal à 0.66 quand aucune feuille n'a été enlevée des arbres (voir la figure 5.4a).

La figure 5.4b montre que l'impact de l'élimination des feuilles est particulièrement perceptible pour les grandes phylogénies (*c.-à-d.*, pour $n > 16$).

En fonction du bruit ajouté aux arbres phylogénétiques (voir la figure 5.4c), le score de l'*ARI* moyen baisse de 1.0 à 0.63 pour 0% de feuilles enlevées et de 0.89 à 0.51 pour 50% de feuilles enlevées.

5.7 Conclusion

Dans ce chapitre, nous avons présenté différentes versions de l'algorithme des *k*-moyennes de partitionnement d'arbres phylogénétiques permettant d'inférer des arbres consensus et des super-arbres multiples. Nous avons considéré plusieurs approximations de la fonction objective des *k*-moyennes et de l'indice de validité des clusters de Caliński-Harabasz (*CH*), qui était l'indice le plus performant (dans le cadre des *k*-moyennes) selon nos simulations. Quelques résultats théoriques intéressants ont été obtenus (voir *p.ex.* le Théorème 5.1). La distance utilisée dans ce chapitre était toujours la distance topologique de Robinson et Foulds (*RF*).

Les résultats de nos simulations démontrent que le nouvel algorithme est beaucoup plus rapide que l'approche de partitionnement d'arbres de Stockham *et al.* (Stockham, Wang et Warnow, 2002), tout en offrant de meilleures solutions de partitionnement dans la plupart des cas.

Pour les cas des super-arbres, nous avons développé une extension de la fonction objective des k-moyennes et du critère de Caliński-Harabasz, prenant en compte les ensembles de feuilles communes entre les paires d'arbres phylogénétiques.

Dans le chapitre VI, nous utiliserons des données biologiques et linguistiques réelles pour tester nos nouveaux algorithmes et comparerons nos résultats avec ceux discutés dans la littérature.

CHAPITRE VI

APPLICATION DES ALGORITHMES DE PARTITIONNEMENT D'ARBRES AUX JEUX DE DONNÉES RÉELLES

«*Consensus trees and supertrees are regularly used in systematic biology in order to obtain a summary for the common agreement of the evolutionary relationships among a collection of phylogenetic trees (hierarchies).*», McMorris et Powers, 2009.

6.1 Préface

Dans ce chapitre, nous appliquerons aux données réelles (biologiques et linguistiques) les algorithmes de partitionnement d'arbres phylogénétiques décrits dans les chapitres précédents. Nous analyserons trois ensembles de données : les données d'archées (Matte-Tailliez *et al.*, 2002), les données linguistiques (Dyen, Kruskal et Black, 1992; Gray et Atkinson, 2003) et les données phylogénétiques considérées par Stockham *et al.* (2002). Ces trois ensembles de données sont disponibles dans le répertoire Github suivant : https://github.com/TahiriNadia/dataset. Nous étudierons les données issues de l'étude de Stockham *et al.* (2002) afin de valider notre approche et de comparer nos résultats avec ceux obtenus par ces auteurs.

6.2 Introduction

Nous testerons nos approches de partitionnement d'arbres en nous basant dans un premier temps sur l'algorithme des *k*-médoïdes (voir le chapitre III) et les deux indices de validité des clusters associés : le critère de Caliński-Harabasz (*CH*) (Caliński et Harabasz, 1974) et le critère Silhouette (*SH*) (Rousseeuw, 1987). Dans un deuxième temps, nous appliquerons aux données réelles notre nouvel algorithme basé sur les *k*-moyennes (voir les chapitres IV et V) en utilisant quatre indices de validité des clusters étudiés : l'indice *CH*, l'indice *SH*, le nouvel indice *W* et l'indice *Gap* (Tibshirani, Walther et Hastie, 2001). Enfin, dans un troisième temps, nous considérerons la nouvelle fonction objective de l'algorithme des *k*-moyennes permettant de construire des super-arbres multiples (voir le chapitre V). Le reste du chapitre s'articule autour de trois jeux de données réelles, à savoir : les données d'archées (Matte-Tailliez *et al.*, 2002), les données linguistiques (Dyen, Kruskal et Black, 1992; Gray et Atkinson, 2003) et les données phylogénétiques analysées par Stockham *et al.* (2002). Nous décrirons en détails ces trois jeux de données dans la section 6.3, puis discuterons des résultats d'application de nos méthodes dans la section 6.4. Enfin, dans la section 6.5, nous ferons quelques conclusions en rapport avec l'application de nos méthodes à des données réelles.

6.3 Description des jeux de données réelles

Nous avons testé nos nouveaux algorithmes de partitionnement d'arbres (voir les chapitres III à V) sur des ensembles de données biologiques et linguistiques présentés ci-dessous.

6.3.1 Données d'archées

Le groupe des archées (ou archaebactéries) constitue l'un des trois groupes majeurs de l'Arbre de Vie avec les Bactéries et les Eucaryotes (voir la figure 6.1). Dans cette étude, nous avons examiné les 14 organismes d'archées considérés dans l'article de Matte-Tailliez *et al.* (2002). Nous avons étudié l'évolution de 52 protéines ribosomales de 7175 acides aminés (aa) de ces organismes. Plusieurs gènes codant pour ces 52 protéines sont affectés par des transferts horizontaux de gènes (HGT) (Matte-Tailliez *et al.*, 2002). De nombreuses études (Matte-Tailliez *et al.*, 2002; Boc, Philippe et Makarenkov, 2010; Boc, Legendre et Makarenkov, 2013) ont confirmé que les archées utilisent les transferts horizontaux de gènes pour s'adapter aux changements de l'environnement. Par exemple, Boc *et al.* (Boc, Philippe et Makarenkov, 2010) ont montré que cinq transferts étaient nécessaires pour réconcilier l'arbre d'espèces et l'arbre du gène *rpl*12*e*. Nous avons testé notre algorithme sur cet ensemble de données et identifié des sous-ensembles de gènes ayant des antécédents évolutionnaires communs (*c.-à-d.*, les gènes qui ont subi les mêmes transferts horizontaux).

Les archées forment une collection de procaryotes diversifiés phénotypiquement, qui sont reconnus comme une unité taxonomique majeure (*c.-à-d.*, un des trois domaines de l'arbre de vie, voir la figure 6.1). Certaines archées présentent des caractéristiques uniques dans leurs modes de vies, telles que la méthanogenèse ou la capacité de croître à des températures au-dessus de 95°C ou dans des milieux très acides. La recherche de nouvelles archées par PCR (ou réaction en chaîne par polymérase) en différents milieux a relevé que la lignée des archées est un élément important de la biosphère (Pace, 1997). Les études moléculaires et celles de la génomique comparative ont démontré que les archées se caractérisent par une combinaison de caractéristiques uniques, telles que des isoprénoïdes gauchers contenant des glycéryolides et des caractéristiques de bactéries et d'eucaryotes en mosaïque. En particulier, malgré une organisation bactérienne de leur chromosome, *c.-à-d.*, la

présence de gènes assemblés en opérons, une origine commune de réplication bidirectionnelle et la séquence Shine-Dalgarno en amont du codon d'initiation de l'ARN (Acide RiboNucléique) messagers (ou ARNm), la plupart de leurs protéines informatives ressemblent à celles des eucaryotes (Forterre, 1997; Olsen et Woese, 1997; Rivera *et al.*, 1998; Makarova *et al.*, 1999). L'étude des archées est essentielle pour comprendre l'histoire des mécanismes moléculaires et la diversité des métabolismes sur notre planète, ainsi que pour démêler les mécanismes par lesquels la vie peut prospérer dans des environnements extrêmes.

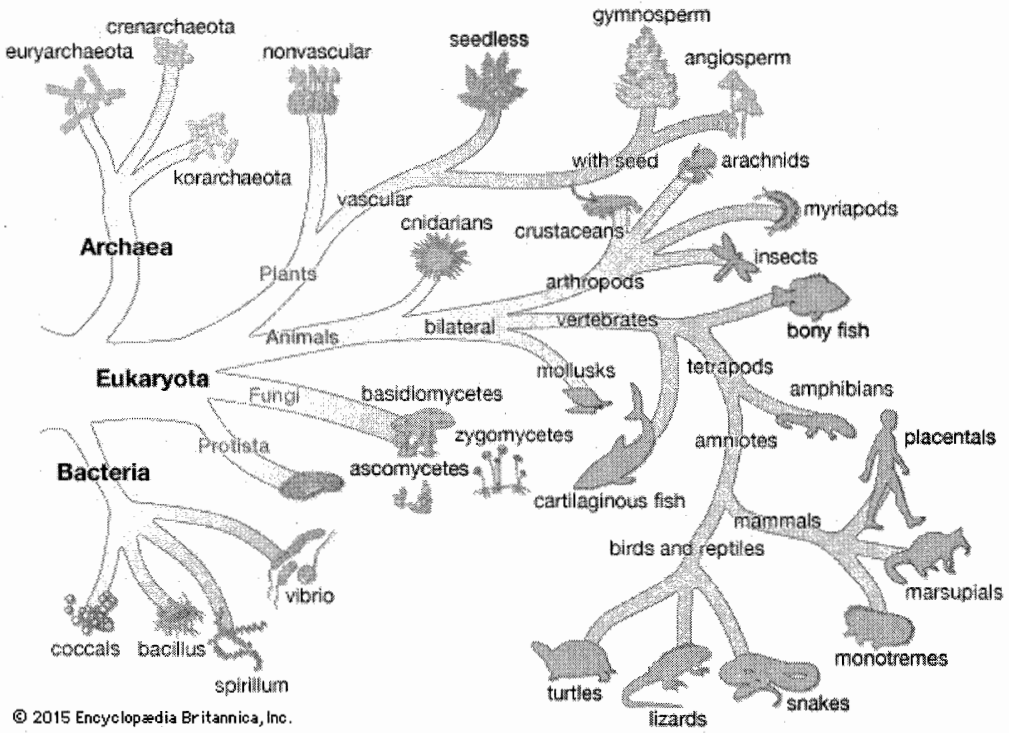


Figure 6.1 Représentation actuelle de l'arbre de vie (reproduction de l'encyclopédie en ligne Britannica[1]).

[1] https://www.britannica.com/science/archaea

Dans cette étude, nous avons sélectionné l'ensemble des espèces considérées par Matte-Tailliez *et al.* (2002), qui se divise en deux clades (voir le Glossaire pour la définition), dont 11 espèces appartiennent au clade d'Euryarchaeota et 3 espèces appartiennent au clade de Crenarchaeota (voir la figure 6.2).

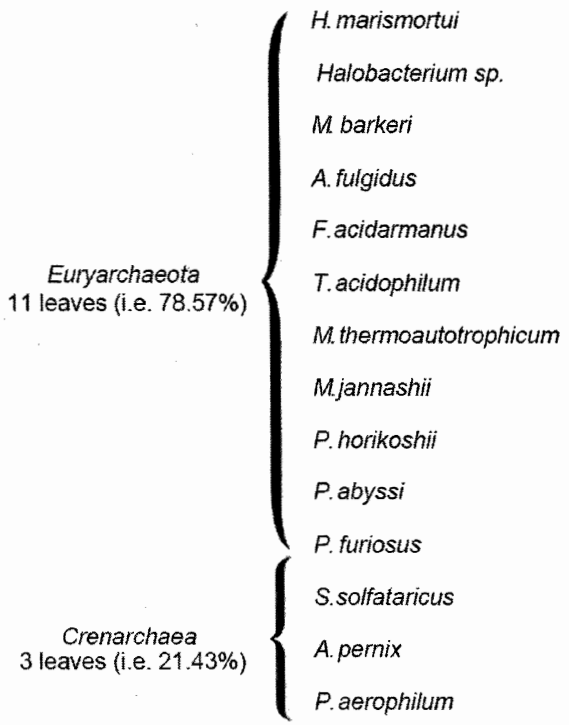


Figure 6.2 Description des données de l'arbre d'espèces des archées pour les données de Matte-Tailliez *et al.* (2002).

6.3.2 Données linguistiques

Les biologistes ne sont pas seuls à utiliser le concept d'arbre évolutif de Darwin pour représenter l'histoire évolutive des espèces. Il existe de nombreuses recherches portant sur les arbres pour étudier l'évolution linguistique en associant les concepts de phylogénie et d'arbre d'évolution des langues naturelles (Atkinson et Gray, 2005).

Cette ressemblance curieuse entre les processus d'évolution des espèces et des langues a été observée même avant « *The Descent of Man* » de Darwin (Darwin, 1871). En août 1863, August Schleicher (Schleicher, 1863) a envoyé une lettre à Ernst Haeckel dans laquelle il discutait de certaines de ces similitudes en comparant, par exemple, des langages mixtes à des plantes hybrides en botanique. Plus récemment, cette ressemblance a été soulignée par Atkinson et Gray (2005). Ces chercheurs ont présenté un tableau mettant en évidence les parallèles conceptuels les plus importants pouvant être établis entre ces phénomènes évolutifs. En particulier, cette dernière étude compare le processus de sélection sociale en linguistique à la sélection naturelle des espèces, l'emprunt de mots à travers les langues au transfert horizontal des gènes, les langues créoles aux plantes hybrides, les textes anciens aux fossiles et les cognats aux homologies. Il existe également quelques différences entre ces processus (Geisler et List, 2013). Par exemple, l'alphabet biologique (*p. ex.* ADN) est universel pour tous les gènes, alors que l'ensemble de sons utilisés pour former les mots est spécifique à chaque langue. De plus, les données de séquences sont généralement beaucoup plus longues en biologie moléculaire qu'en linguistique. La sélection d'une liste parfaite de significations de base pour l'application des méthodes phylogénétiques dans le contexte de l'évolution des langues naturelles reste une tâche difficile (Gray et Atkinson, 2003). Néanmoins, les similitudes et les parallèles entre les deux disciplines permettent aux chercheurs d'utiliser des méthodes de biologie computationnelle, bien développées pour étudier l'évolution des espèces et en particulier l'évolution réticulée, dans le domaine de la linguistique. Évidemment, il n'est pas possible d'appliquer ces méthodes de biologie computationnelle directement, sans une adaptation appropriée, ce qui est essentiel dans la recherche interdisciplinaire. Ainsi, les algorithmes phylogénétiques existants doivent être modifiés, et les flux de travaux adaptés, afin d'obtenir des résultats et interprétations linguistiques significatifs (Willems *et al.*, 2016).

Deux séquences nucléotidiques observées chez deux espèces distinctes sont dites *homologues* si elles ont évolué à partir d'une séquence ancestrale commune (Fitch, 2000). De même, en linguistique, un groupe de mots apparentés est un ensemble de formes de mots dans différentes langues qui ont été héritées d'une forme de mot ancestral commune (Trask, 2000), indiqué par le terme de *cognat*. La principale différence entre ces concepts est que le concept d'homologie inclut la possibilité de transferts latéraux, alors que le concept relatif aux cognats exclut tous les processus d'emprunt potentiels. Les arbres de cognats et les arbres phylogénétiques jouent un rôle fondamental dans l'étude de l'évolution des langues naturelles à l'aide de méthodes phylogénétiques (Gray et Atkinson, 2003). Par exemple, un arbre phylogénétique représentant les principaux traits de l'évolution lexicale équivaut à une phylogénie des espèces représentant les principaux événements de spéciation (Atkinson et Gray, 2005).

L'origine des langues Indo-Européennes (IE) est un des problèmes clefs de l'histoire des langues. Deux théories principales s'affrontent. Elles sont les suivantes : la première remonte l'origine des langues IE à 5000 ou 6000 ans av. J.-C. au sein de la culture Kourgane, alors que la deuxième est basée essentiellement sur des études archéologiques et place l'origine des langues IE entre 8000 et 9500 ans av. J.-C. en Anatolie (Gray et Atkinson, 2003; Renfrew, 1988; Renfrew, 1990; Bouckaert *et al.*, 2012). Gray et Atkinson (Gray et Atkinson, 2003, voir la figure C.1 de l'Appendice C) ont retrouvé un arbre conforme à l'hypothèse Anatolienne en utilisant des méthodes phylogénétiques bayésiennes (voir Appendice C, la figure C.1).

Dans notre étude, nous avons d'abord considéré un ensemble de 95 langues IE étudiées par Dyen *et al.* (Dyen, Kruskal et Black, 1992). La base de données de Dyen *et al.* contient les traductions en 95 langues des 200 mots de la liste de Swadesh (1952). Cette liste est une des diverses listes de mots de signification de base établies par M. Swadesh dans l'intervalle entre 1940 et 1950. Ces données sont souvent

utilisées en lexicostatistique (qui étudie l'évaluation quantitative de la parenté des langues) et en glottochronologie (qui étudie la datation des divergences entre les langues). Dyen *et al.* ont regroupé les traductions des mots en des ensembles de cognats. Un ensemble de cognats est un groupe de mots connexes (ou de traductions en différentes langues), ayant une racine et donc une origine commune. Nous étudierons en détail la version de cette base de données qui a été établie par Boc *et al.* (Boc, Di Sciullo et Makarenkov, 2010). Premièrement, nous ne considérerons que les 84 langues IE les plus répandues parmi les 95 langues originales (*p. ex.* les langues manquantes de traductions, telles que Hittite, ne seront pas considérées). Deuxièmement, certains cognats douteux seront supprimés et de nouveaux ensembles de cognats, composés de cognats douteux, seront créés, comme suggéré dans Boc *et al.* (Boc, Di Sciullo et Makarenkov, 2010). Les données considérées seront ainsi regroupées en 1315 ensembles de cognats de façon à ce que les mots empruntés soient ajoutés aux ensembles de cognats originaux (les traductions qui étaient reliées par des emprunts ou des similarités accidentelles ont été mises dans une classe séparée par Dyen *et al.* (Dyen, Kruskal et Black, 1992)). Les données linguistiques analysées sont disponibles sur : https://github.com/TahiriNadia/traitement_data_linguistic.

6.3.3 Données de Stockham *et al.*

Nous avons utilisé quatre jeux de données biologiques, considérés originellement par Stockham, Wang et Warnow (2002), pour pouvoir comparer nos solutions à celles obtenues par ces auteurs.

Le premier jeu de données est appelé Camp (*c.-à-d.*, *Campanulaceae*) (Stockham, Wang et Warnow, 2002; Cosner *et al.*, 2000). Il concerne une famille des plantes. Cet ensemble de données contient les séquences d'acides désoxyribonucléiques de chloroplaste. Les séquences de Camp sont hautement conservées (*p.ex.* ordre des

gènes, taille de génomes). Pour cette raison elles sont fréquemment utilisées comme marqueurs phylogénétiques d'intérêt (Downie et Palner, 1992).

Le second jeu de données est appelé Caesal (*c.-à-d.*, *Caesalplinia*) (Stockham, Wang et Warnow, 2002). Il concerne la plante incluse dans le groupe des Angiospermes. Cet ensemble de données concerne l'intron trnL-trnF et les régions d'espacements de génomes de chloroplastes.

Le troisième et le quatrième ensembles de données sont appelés PEVCCA1 et PEVCCA2, respectivement (*c.-à-d.*, *Porifera, Echinodermata, Vertebrata, Cnidaria, Crustacea* et *Annelida*). Ces jeux de données incluent des petites séquences d'acides ribonucléiques ribosomaux de 129 espèces (Stockham, Wang et Warnow, 2002).

6.4 Résultats obtenus pour les jeux de données réelles

6.4.1 Application aux données d'archées

Nous considérons d'abord l'évolution du gène *rpl12e* pour les 14 organismes d'archées étudiés par Matte-Tailliez *et al.* (2002) (voir la figure 6.3). L'arbre présenté à la figure 6.3 montre les espèces qui ont été considérées dans notre étude. Matte-Tailliez *et al.* ont inféré cet arbre phylogénétique d'espèces après la concaténation de toutes leurs séquences protéiques disponibles. Cependant, l'évolution de chacune de ces protéines peut être représentée par son propre arbre phylogénétique. L'analyse de ces données par le partitionnement nous renseignera sur le nombre de scénarios évolutifs différents caractérisant l'histoire évolutive de ces séquences (*c.-à-d.*, combien existe-t-il de clusters d'arbres dans l'ensemble d'arbres phylogénétiques relatifs aux protéines considérées). Nous avons testé notre algorithme sur deux jeux de données d'archées différents. Le premier jeu de données est constitué de 52 alignements définis sur différents ensembles de feuilles, qui se chevauchent partiellement (ce qui permet de tester sur ces données les algorithmes de construction

de super-arbres multiples). Il s'agit du jeu de données original de l'étude de Matte-Tailliez *et al.* (2002). Le deuxième jeu de données est constitué de 47 alignements construits pour le même ensemble des 14 espèces initiales (parmi les 52 alignements originaux). Les cinq alignements restants sont incomplets (*c.-à-d.*, les alignements incluant seulement 12 et 13 organismes parmi les 14 organismes d'origine). Nous avons considéré le premier et le deuxième jeux de données pour tester nos algorithmes dans le cas des super-arbres multiples et des arbres consensus multiples, respectivement (voir le chapitre V). Nous avons d'abord inféré 52 arbres phylogénétiques (pour le premier jeu de données) et 47 arbres phylogénétiques (pour le deuxième jeu de données) au moyen de la méthode PhyML (Guindon et Gascuel, 2003). Les 52 alignements initiaux, ainsi que leurs arbres phylogénétiques inférés sont disponibles sur le site web suivant : https://github.com/TahiriNadia/dataset/tree/master/data_archeae. Nous avons ensuite déterminé l'arbre consensus majoritaire de chaque cluster trouvé par notre algorithme, en utilisant les logiciels : 1) *Consense* du Package PHYLIP (Felsenstein, 1985), dans le cas des arbres consensus, *c.-à-d.*, pour les arbres définis sur un même ensemble de feuilles, et 2) CLANN (Creevey et McInerney, 2004), dans le cas de super-arbres, *c.-à-d.*, pour les arbres n'ayant pas les mêmes ensembles de feuilles. Ensuite, nous avons utilisé le programme T-Rex (Boc, Diallo et Makarenkov, 2012), disponible sur le site web suivant http://www.trex.uqam.ca, pour calculer le scénario unique des transferts horizontaux de gènes pour chaque arbre consensus ou chaque super-arbre trouvé. Nous avons considéré l'arbre phylogénétique de Matte-Tailliez *et al.* (voir la figure 6.3) comme l'arbre d'espèces et l'arbre consensus associé à chacun des clusters retrouvés comme l'arbre de gènes dans l'algorithme de recherche des transferts horizontaux de gènes de Boc *et al.* (2010) que nous avons appliqué.

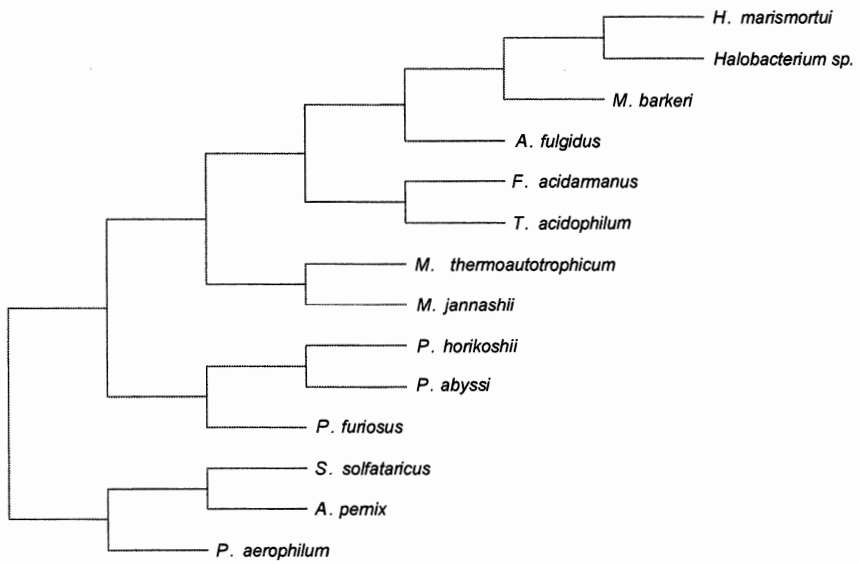


Figure 6.3 L'arbre d'espèces (Matte-Tailliez *et al.* 2002, figure 1a dans cet article). Cet arbre phylogénétique a été obtenu par concaténation des 53 protéines ribosomales (7 175 positions) de 14 organismes d'archées. Le programme PUZZLE (Strimmer et von Haeseler, 1996) avec la correction Γ-law a été utilisé pour trouver le meilleur choix d'arbre et les longueurs de branches optimales.

6.4.2 Résultats obtenus par l'algorithme des *k*-médoïdes pour les données des archées

Tout d'abord, nous avons testé la version de notre algorithme de partitionnement d'arbres basé sur les *k*-médoïdes (voir le chapitre III), l'indice *SH* (voir les formules 3.7 à 3.10) et la distance *RF* non mise au carré, afin d'inférer un partitionnement de l'ensemble de 47 arbres (*c.-à-d.*, notre deuxième jeu de données) d'archées. Le maximum du critère *SH* a été atteint pour 5 clusters (*c.-à-d.*, $K = 5$), ce qui correspond à 5 scénarios de transferts horizontaux de gènes différents représentés sur la figure 6.4 (panneaux b à f). Les résultats détaillés obtenus pour ce jeu de données sont disponibles au répertoire Github suivant : https://github.com/TahiriNadia/CKMedoidsTreeClustering/blob/master/output.

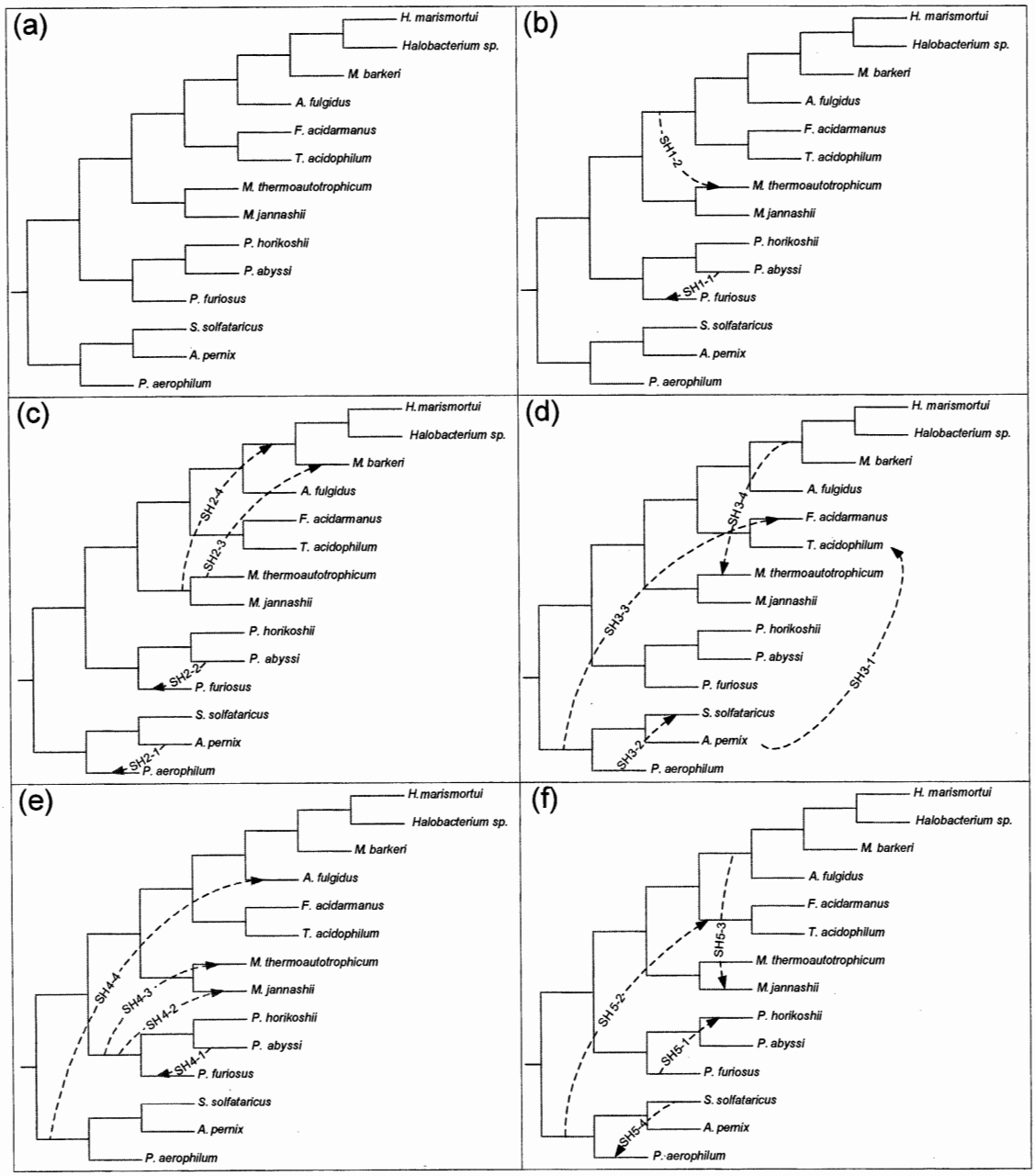


Figure 6.4 (a) L'arbre d'espèces pour le jeu de données d'archées et les 5 scénarios de transferts horizontaux de gènes (panneaux b à f) obtenus pour les 47 arbres protéiques, étudiés originalement par Matte-Tailliez *et al.* (2002), en utilisant l'indice de validité des clusters *SH* et la distance topologique de Robinson et Foulds non mise

au carré dans l'algorithme des *k*-médoïdes de partitionnement d'arbres phylogénétiques.

Le premier cluster obtenu contient 11 arbres, le deuxième cluster obtenu contient 4 arbres, le troisième cluster obtenu contient 20 arbres, le quatrième cluster obtenu contient 11 arbres et le cinquième cluster obtenu contient 1 arbre. Nous avons ensuite inféré les arbres consensus majoritaires, SH1, SH2, SH3, SH4 et SH5, pour ces 5 clusters d'arbres. Puis, en utilisant l'algorithme de détection des transferts horizontaux de gènes de Boc *et al.* (2010), nous avons identifié les scénarios d'événements de transferts qui réconcilient l'arbre d'espèces (voir la figure 6.4a) et chacun des arbres consensus obtenus, de SH1 à SH5 (voir les figures de 6.4b à 6.4f). Ces transferts tiennent compte de différentes histoires qui caractérisent l'évolution des 47 protéines ribosomales considérées. Deux transferts prédits par Boc *et al.* (voir la figure 6 dans Boc, Philippe et Makarenkov, 2010), qui sont également en accord avec les résultats de Matte-Tailliez *et al.* (2002) et de Boc *et al.* (2013), peuvent être trouvés dans nos scénarios. Précisément, les transferts SH3-2 (ou son transfert équivalent SH5-4) et SH5-1 ont été prédits par Boc *et al.* 2010 (voir la figure 6 dans Boc, Philippe et Makarenkov, 2010). De plus, les transferts SH3-2 (ou son transfert équivalent SH5-4) et SH3-1 ont été prédits par Boc *et al.* 2013 (voir la figure 3 dans Boc, Legendre et Makarenkov, 2013) comme transferts horizontaux de gènes partiels (*c.-à-d.*, les transferts conduisant à la formation d'un gène chimère composé de portions de deux ou plusieurs séquences génétiques).

Le tableau 6.1 montre la similarité entre les arbres consensus obtenus pour chacun des cinq clusters (correspondant aux arbres de gènes) et l'arbre de référence (correspondant à l'arbre d'espèces). L'arbre de référence pour ce jeu de données est illustré à la figure 6.4(a) dont l'encodage en format Newick est comme suit :

(((*A.pernix*:1.0,*S.solfataricus*:1.0):1.0,*P.aerophilum*:1.0):10.0,(((*P.abyssi*:1.0,*P.horikoshii*:1.0):1.0,*P.fuiosus*:1.0):1.0,((*M.jannaschii*:1.0,*M.thermoautrophicum*:1.0):1.0,((*T.acidarinanus*:1.0,*F.acidarinanus*:1.0):1.0,(((*Halobacterium.sp.*:1.0,*H.marismortui*:1.0):1.0,*M.barkeri*:1.0):1.0,*A.fulgidus*:1.0):1.0):1.0):1.0):1.0);

Tableau 6.1 La distance *RF* entre l'arbre d'espèces et l'arbre consensus du cluster *k* (*k* = 1, ..., 5), puis cette même distance normalisée par 2*n*-6 (soit 2*14-6=22) et le nombre d'arbres pour chacun des cinq clusters obtenus en utilisant l'algorithme des *k*-médoïdes et le critère Silhouette pour partitionner les 47 arbres d'entrée construits pour les 14 espèces d'archées (Matte-Tailliez *et al.*, 2002).

k	Nombre d'arbres	Distance *RF* entre l'arbre consensus du cluster *k* et l'arbre d'espèces	Distance *RF* normalisée entre l'arbre consensus du cluster *k* et l'arbre d'espèces
1	11	4	0.182
2	4	10	0.455
3	20	2	0.091
4	11	10	0.455
5	1	16	0.727

La valeur de similarité entre chaque paire d'arbres est mesurée en utilisant la distance topologique de Robinson et Foulds (Robinson et Foulds, 1981). Nous avons normalisé cette distance par 2*n*-6 (soit 2*14-6=22) pour que toutes les valeurs obtenues soient comprises entre les bornes 0 et 1. Les résultats comparatifs obtenus pour ce jeu de données d'archées en utilisant notre algorithme des *k*-médoïdes et le critère de validité des clusters Silhouette sont présentés au tableau 6.1. La distance *RF* normalisée entre l'arbre consensus du cluster 1 et l'arbre d'espèces est égale à 0.182, pour le cluster 2 cette distance est égale à 0.455, pour le cluster 3 cette distance est égale à 0.091, pour le cluster 4 cette distance est égale à 0.455 et pour le cluster 5 cette distance est égale à 0.727. L'arbre consensus du cluster 3 est donc le plus proche de l'arbre d'espèces que les autres arbres consensus (voir le tableau 6.1).

Nous avons également utilisé le programme MCT (*Multiple Consensus Trees*) de Guénoche pour analyser ce jeu de données d'archées. La stratégie de lien moyen hiérarchique (*Average linkage hierarchical strategy*) et la distance de Robinson et Foulds étaient les paramètres que nous avons sélectionnés dans le programme MCT. Nous avons comparé nos arbres consensus trouvés par les *k*-médoïdes avec les arbres consensus trouvés par l'algorithme de Guénoche (2013). Par exemple pour $K = 5$ (c'était le nombre optimal de classes trouvées à l'aide de notre algorithme pour l'indice *SH*), le programme MCT a retourné les arbres consensus de gènes dont les topologies ont conduit aux transferts horizontaux suivants : SH2-1, SH2-3, SH4-2 et SH4-3 (voir la figure 6.4).

Deuxièmement, nous avons utilisé la version de notre algorithme de regroupement d'arbres basé toujours sur les *k*-médoïdes (voir le chapitre III), mais en utilisant l'indice *CH* (voir les formules 3.2, 3.4 et 3.6) et la distance *RF* non mise au carré, pour classer le même ensemble de 47 arbres phylogénétiques d'archées. Le maximum de l'indice *CH* a été obtenu pour le cas de trois clusters ($K = 3$). Le premier cluster contient 25 arbres, le second contient 14 arbres et le troisième contient 8 arbres. Nous avons ensuite déterminé les arbres consensus majoritaires étendus, CH1, CH2 et CH3, pour chacun de ces trois clusters. Comme pour le cas de l'indice *SH*, nous avons identifié les scénarios d'événements de transferts horizontaux de gènes qui réconcilient l'arbre d'espèces (voir la figure 6.4a) et chacun des arbres consensus CH1, CH2 et CH3 (voir la figure 6.5, panneaux a à c). Le transfert CH2-1 que nous avons retrouvé a été également prédit par Boc *et al.* 2010 (voir la figure 6 dans Boc, Philippe et Makarenkov, 2010), et finalement les transferts CH1-1 (ou son transfert équivalent CH3-1), CH1-2, CH1-3, CH2-1, CH3-2 et CH4-3 ont été prédits par Boc *et al.*, 2013 comme des transferts horizontaux de gènes partiels (voir la figure 3 dans Boc, Legendre et Makarenkov, 2013). De plus, tous les transferts qui se trouvent dans les scénarios de transferts horizontaux de gènes représentés dans la figure 6.5 (a-c)

peuvent être retrouvés dans les scénarios de transferts horizontaux de gènes représentés dans la figure 6.4 (b-f).

Enfin, nous avons exécuté le programme MCT pour $K = 3$ (il s'agissait du nombre optimal de clusters trouvés en utilisant notre algorithme avec l'indice *CH*) et comparé les arbres consensus obtenus par MCT avec ceux trouvés par notre méthode. Les arbres consensus obtenus par MCT dans ce cas ont permis de retrouver quatre transferts horizontaux de gènes équivalents aux transferts CH1-2, CH1-3, CH3-2 et CH3-3, (voir la figure 6.5) trouvés par notre algorithme des *k*-médoïdes de partitionnement d'arbres basé sur l'indice *CH*.

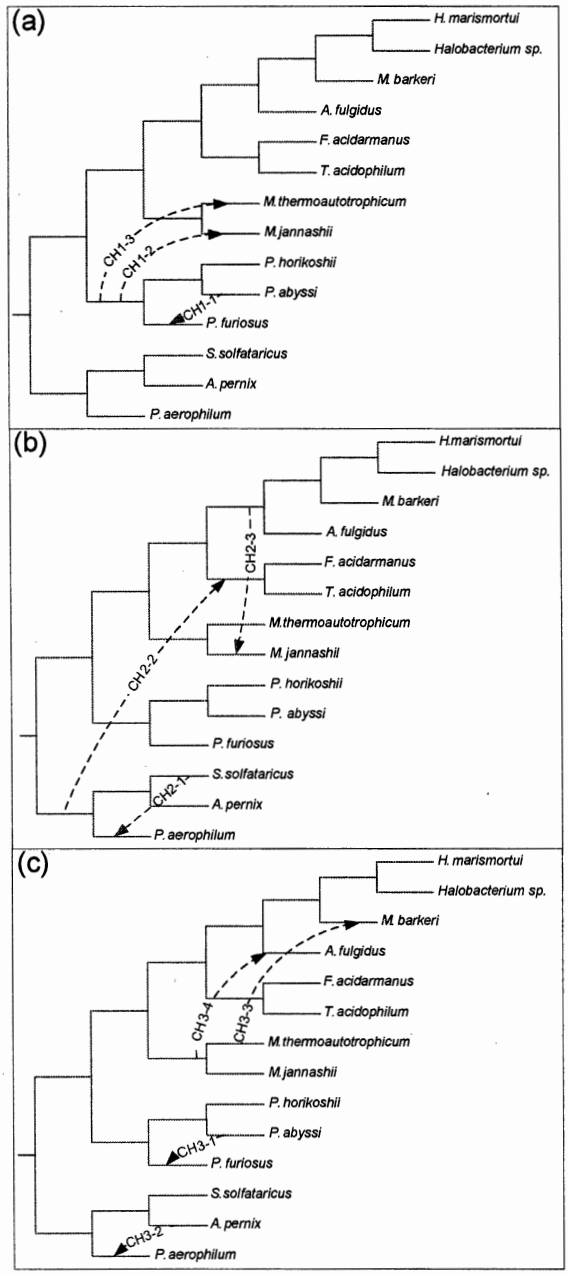


Figure 6.5 Trois arbres consensus majoritaires et trois scénarios de consensus de transferts horizontaux de gènes (panneaux a à c) obtenus pour les 47 arbres protéiques étudiés originalement par Matte-Tailliez *et al.* (2002). Pour obtenir ces résultats, nous avons utilisé le critère de validité des clusters *CH* et la distance topologique de

Robinson et Foulds non mise au carré comme paramètres de notre algorithme des *k*-médoïdes de partitionnement d'arbres phylogénétiques.

Le tableau 6.2 montre la similarité entre les arbres consensus obtenus pour chacun des trois clusters (correspondant aux arbres consensus différents) et l'arbre de référence (correspondant à l'arbre d'espèces). Les résultats obtenus pour ce jeu de données d'archées de 47 arbres phylogénétiques et 14 espèces, en utilisant l'algorithme des *k*-médoïdes et le critère de validité des clusters de Caliński-Harabasz sont indiqués dans ce tableau. La distance *RF* entre l'arbre consensus du cluster 1 et l'arbre d'espèces est égale à 6, alors que pour les clusters 2 et 3, cette distance est égale à 10. L'arbre consensus du cluster 1 est donc plus proche de l'arbre d'espèces que les arbres consensus des clusters 2 et 3.

Tableau 6.2 La distance *RF* entre l'arbre d'espèces et l'arbre consensus du cluster *k*, ($k = 1, \ldots, 3$), cette même distance normalisée par $2n$-6 (soit 2*14-6=22) et le nombre d'arbres pour chacun des trois clusters obtenus en utilisant l'algorithme des *k*-médoïdes et le critère de Caliński-Harabasz utilisés pour partitionner les 47 arbres d'entrée construits pour 14 espèces d'archées (Matte-Tailliez *et al.*, 2002).

k	Nombre d'arbres	Distance *RF* entre l'arbre consensus du cluster *k* et l'arbre d'espèces	Distance *RF* normalisée entre l'arbre consensus du cluster *k* et l'arbre d'espèces
1	25	6	0.273
2	14	10	0.455
3	8	10	0.455

Les résultats globaux, dont les fréquences des transferts de gènes intragroupe et intergroupe trouvés par l'algorithme des *k*-médoïdes de partitionnement d'arbres en utilisant les indices *SH* et *CH*, sont indiqués dans le tableau 6.3. Ils suggèrent que les

transferts de gènes ont été plus fréquents entre les espèces du même phylum qu'entre les espèces de phyla différents (Crenarchaeota et Euryarchaeota).

Tableau 6.3 Les statistiques de transferts horizontaux de gènes pour les 47 arbres protéiques ribosomaux pour les 14 espèces d'archées, obtenues en utilisant les indices de validité des clusters *SH* et *CH* et la distance topologique *RF* non mise au carré dans l'algorithme des *k*-médoïdes de partitionnement d'arbres. Le groupe Crenarchaea est composé des 3 espèces suivantes : *S. solfactaricus*, *A. pernix* et *P. aerophilum* et le groupe d'Euryarchaeota est composé des 11 espèces suivantes : *P. furiosus*, *P. abyssi*, *P. horikoshii*, *M. jannashii*, *M. thermoautotrophicum*, *T. acidophilum*, *F. acidamanus*, *A. fulgidus*, *M. barkeri*, *Halobacterium sp.* et *H. marismortui.*

Indice	Type de transferts de gènes	Nombre de transferts détectés	Pourcentage de transferts détectés
SH	Intragroupe	14	77.78%
	Intergroupe	4	22.22%
CH	Intragroupe	9	90%
	Intergroupe	1	10%

6.4.3 Résultats obtenus par l'algorithme des *k*-moyennes pour les données des archées

Nous avons également utilisé notre algorithme des *k*-moyennes et, plus précisément sa version de super-arbre (voir le chapitre V), pour inférer un partitionnement optimal de l'ensemble complet de 52 arbres d'archées (*c.-à-d.*, notre premier jeu de données). Le score maximal de *CH* (ou *E-CH*) a été obtenu pour cinq clusters ($K = 5$). Le premier cluster contient 12 arbres, le deuxième cluster contient 1 arbre, le troisième cluster contient 12 arbres, le quatrième cluster contient 8 arbres et le cinquième cluster contient 19 arbres. Nous avons ensuite inféré les super-arbres, CH1, CH2, CH3, CH4 et CH5, pour les cinq clusters d'arbres obtenus en utilisant le logiciel CLANN (Creevey et McInerney, 2004) combiné avec le logiciel PAUP* (Swofford, 2002). L'algorithme de détection des transferts horizontaux de gènes de Boc *et al.*

(2010) a été utilisé pour identifier les scénarios de transferts horizontaux qui réconcilient l'arbre d'espèces (voir la figure 6.6a) et chaque super-arbre consensus obtenus, CH1 à CH5. Ces transferts comptent pour cinq histoires évolutives différentes qui caractérisent l'évolution des 52 protéines ribosomales considérées dans notre étude. Deux transferts prédits par Boc *et al.* (voir la figure 6 dans Boc, Philippe et Makarenkov, 2010), qui sont également en accord avec les résultats de Matte-Tailliez *et al.* (2002) et de Boc *et al.* (2013), peuvent être retrouvés dans les cinq scénarios obtenus. Précisément, les transferts CH2-2 et CH3-3 ont été originellement prédits par Boc *et al.* 2010 (voir la figure 6 dans Boc, Philippe et Makarenkov, 2010) et par Boc *et al.* 2013 (voir la figure 2b dans Boc, Legendre et Makarenkov, 2013) comme transferts horizontaux de gènes partiels (*c.-à-d.*, transferts conduisant à la formation de gènes chimériques composés de portions de deux ou plusieurs séquences codantes génétiquement distinctes). Nous pouvons constater que les cinq scénarios obtenus sont différents. Cela signifie que chacun des cinq super-arbres représente sa propre histoire évolutive (voir la figure 6.6).

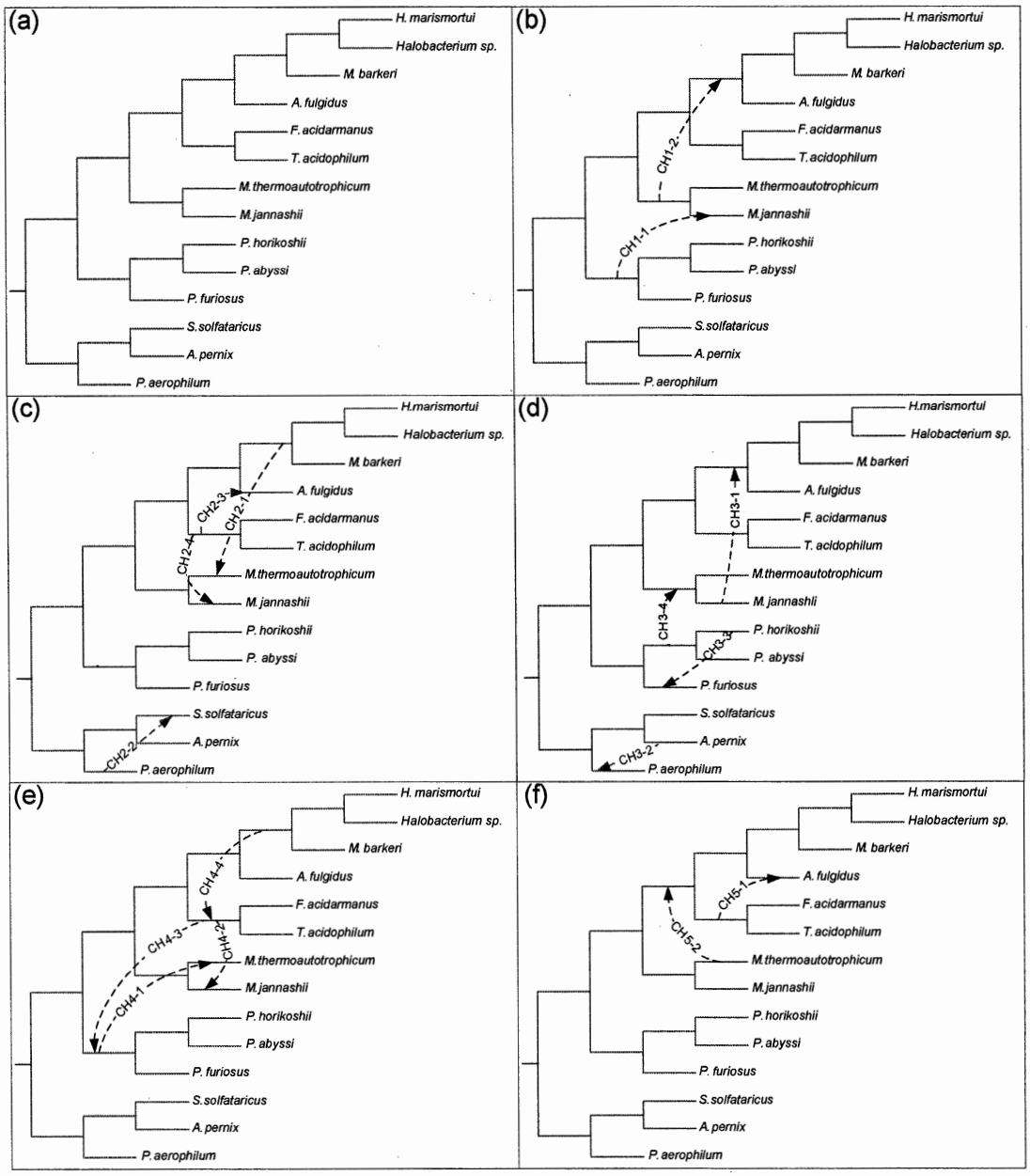


Figure 6.6 (a) L'arbre d'espèces pour le jeu de données d'archées et les 5 scénarios de transferts horizontaux de gènes (panneaux de b à f) obtenus pour les 52 arbres protéiques originalement étudiés par Matte-Taïlliez *et al.* (2002) en utilisant l'approche de super-arbres avec le critère de validité des clusters *E-CH* et la distance

topologique *RF* non mise au carré dans l'algorithme des *k*-moyennes de partitionnement d'arbres phylogénétiques.

La figure 6.6(b) (les figures 6.6c, 6.6d, 6.6e et 6.6f, respectivement) illustre un scénario de transferts permettant de transformer l'arbre d'espèces en arbre consensus du cluster 1 (les arbres consensus des clusters 2 à 5, respectivement). Le premier et le dernier scénarios requièrent deux transferts horizontaux de gènes (flèches de CH1-1 à CH1-2 et de CH5-1 à CH5-2), alors que le deuxième, le troisième et le quatrième scénarios requièrent quatre transferts horizontaux de gènes (flèches CH2-1 à CH2-4, CH3-1 à CH3-4 et CH4-1 à CH4-4) pour cette transformation.

6.4.4 Application des méthodes de partitionnement d'arbres aux données linguistiques

Dans cette section, nous montrons les résultats d'application des méthodes de partitionnement d'arbres basées sur les *k*-médoïdes et les *k*-moyennes sur les données linguistiques de Dyen *et al.* (1992).

6.4.4.1 Résultats obtenus par les algorithmes des *k*-médoïdes et des *k*-moyennes pour les données linguistiques du groupe Germanique du Nord

Dans notre étude, nous nous sommes d'abord concentrés sur le groupe de sept langues Germaniques du Nord (*c.-à-d.*, Islandais ST, Féroïen, Suédois Up, Suédois VL, Suédois List, Danois et Riksmål). Nous avons d'abord calculé les matrices de distances entre ces langues pour les cognats contenant les traductions des langues Germaniques du Nord. La distance $d(L_1,L_2)$ entre deux langues L_1 et L_2 dans un cognat donné a été calculée comme suit. Si les traductions de L_1 et L_2 étaient présentes dans ce cognat, alors $d(L_1,L_2)$ était égale à la distance de Levenshtein normalisée (Yujian et Bo, 2007) entre ces traductions, sinon, $d(L_1,L_2)$ était égale à 1. Nous avons inféré les arbres à partir de ces matrices de distances en utilisant l'algorithme de

Neighbor-Joining (Saitou et Nei, 1987). Nous avons filtré les données originales de Dyen *et al.* pour conserver uniquement les arbres contenant des langues du groupe Germanique du Nord. Notre version de l'algorithme des *k*-médoïdes est sensible aux données manquantes et nous devrions donc avoir pour chaque arbre linguistique (arbre de cognat) le même nombre de feuilles *n* (ici, $n = 7$). Nous avons obtenu 188 arbres avec sept feuilles après l'élimination des arbres contenant de 1 à 6 langues Germaniques du Nord. Les données (matrices et arbres) et les scripts (en Perl et Python) que nous avons utilisés se trouvent dans le répertoire suivant : https://github.com/TahiriNadia/traitement_data_linguistic.

Premièrement, nous avons appliqué la méthode des *k*-médoïdes d'inférence d'arbres consensus, décrite dans le chapitre III, pour reconstruire un ou plusieurs arbres consensus pour l'ensemble des langues Germaniques du Nord. Puis, nous avons appliqué à ces données l'algorithme des *k*-moyennes, en utilisant les différents critères étudiés au chapitre V.

Les résultats obtenus sont résumés dans le tableau 6.4 suivant.

Tableau 6.4 Résultats obtenus par les algorithmes des *k*-médoïdes et des *k*-moyennes de partitionnement d'arbres en utilisant les indices : *CH*, *SH*, *E-CH*, *W*, *Gap*, *LB-CH* et *MI-CH*.

Caractéristiques			Algorithmes / Indices							
Données	Nombre de feuilles	Nombre d'arbres	*k*-médoïdes		*k*-moyennes					
			CH	*SH*	*E-CH*	*SH*	*W*	*Gap*	*LB-CH*	*MI-CH*
Langues Germaniques du Nord	7	188	2	2	2	2	1	2	2	2

Seulement le critère *W* a retourné comme solution optimale la solution avec un seul cluster. Les autres critères de partitionnement suggèrent la présence de deux clusters différents dans ces données (voir le tableau 6.4).

La figure 6.7 présente les résultats obtenus par l'algorithme des *k*-moyennes basé sur le critère *E-CH* (*Euclidean-CH*).

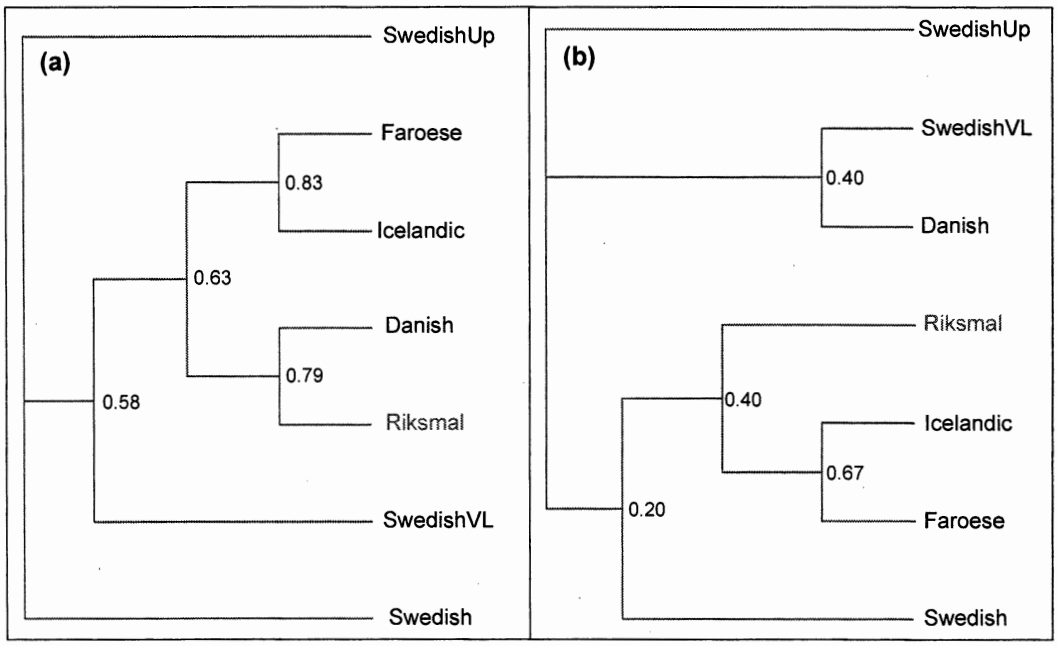


Figure 6.7 Les deux arbres consensus pour un groupe de 188 arbres phylogénétiques appartenant au groupe Germanique du Nord; (a) l'arbre consensus obtenu pour le premier cluster trouvé; et (b) l'arbre consensus obtenu pour le deuxième cluster trouvé.

Ici, nous avons découvert un hybride lexical, la langue Riksmål (en rouge sur la figure 6.7). Le Riksmål est la norme écrite norvégienne contemporaine la plus largement utilisée. Ce résultat a été également observé dans l'étude de Willems *et al.* (2016), voir la figure 6.8b. La langue Riksmål, en tant que destinataire du matériel lexique du Danois (par emprunt de mots) et de l'Islandais (par emprunt de mots

également) a été mise en évidence dans cette dernière étude. Historiquement, les langues Germaniques du Nord ont été divisées en trois branches principales : la Scandinavie Orientale (Danois et Suédois), la Scandinavie de l'Ouest (Islandais, Féroïen et Norvégien) et l'ancien Gotlandais. Le Gotlandais est une langue parlée par les habitants de l'île de Gotland appartenant à la Suède. Il s'agit d'un dialecte du Suédois.

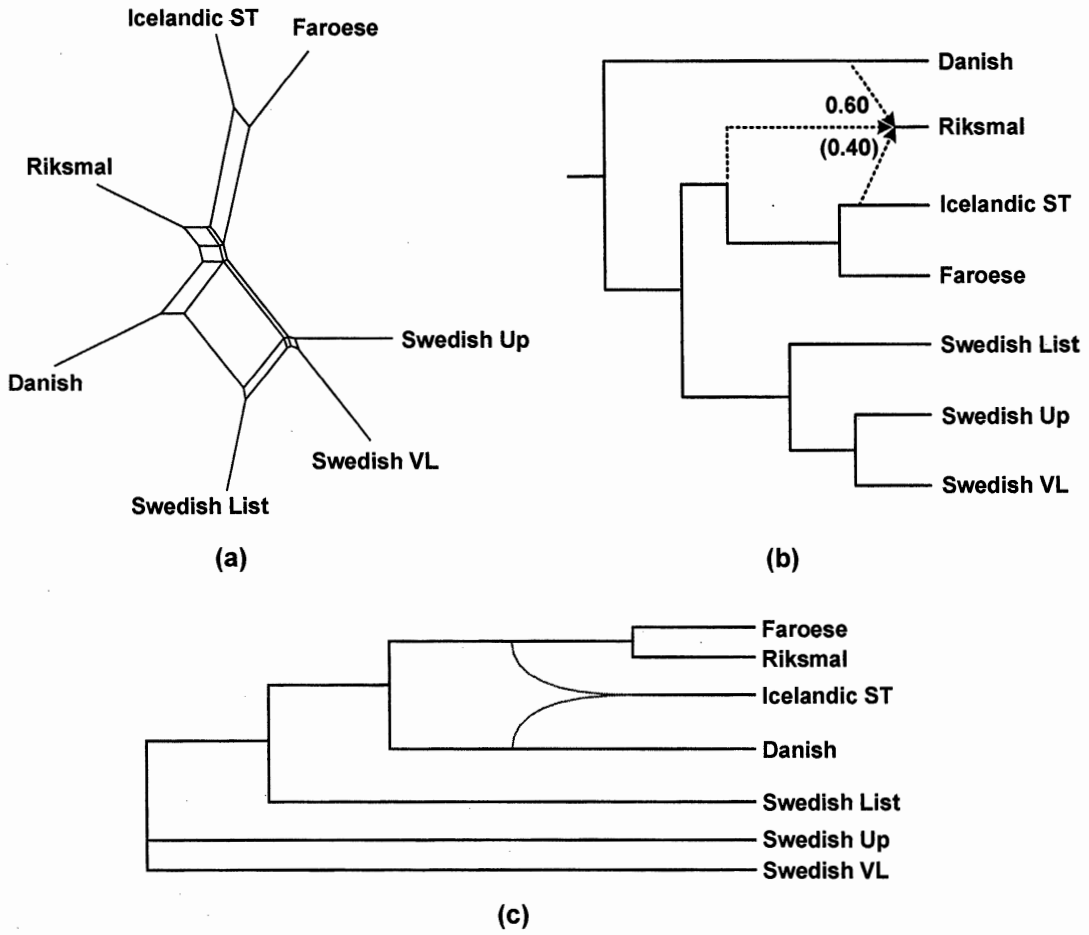


Figure 6.8 (a) Split-graphe, (b) réseau d'hybridation explicite et (c) *galled network* obtenus pour les sept langues du groupe Germanique du Nord (reproduction de l'article de Willems *et al.*, 2016).

D'un côté, l'arbre consensus de la figure 6.7a met en évidence l'influence du Danois sur le Riksmål. Cette influence est due à la domination politique du Danemark sur la Norvège entre la fin du 14^{e} siècle et le début du 19^{e} siècle[1] (Willems *et al.*, 2016). D'un autre côté, l'arbre consensus de la figure 6.7b montre que le Riksmål pourrait également être influencé par l'ancien Islandais.

6.4.4.2 Résultats obtenus par l'algorithme des *k*-moyennes pour les données linguistiques des groupes Germaniques du Nord et de l'Ouest

Nous avons également appliqué notre approche de partitionnement d'arbres à un autre sous-ensemble des données linguistiques (voir la description des données à la section 6.3.2). Ici, nous nous sommes concentrés exclusivement sur un ensemble de 12 langues appartenant aux groupes Germaniques du Nord et de l'Ouest. Ces langues sont les suivantes : Islandais, Féroïen (ou Farose), Suédois, Danois et Riksmål du groupe Germanique du Nord et Néerlandais, Flamand, Germanique ST, Frisian, PennDutch, Sranan et Anglais du groupe Germanique de l'Ouest. Parmi les arbres de 1315 cognats prétraités (*c.-à-d.*, corrigés par Boc *et al.* (Boc, Di Sciullo et Makarenkov, 2010)), disponibles sur le site web http://www.trex.uqam.ca/bioling_interactive ou à partir du répertoire de Github suivant : https://github.com/TahiriNadia/CKMeansTreeClustering, nous avons conservé uniquement les arbres de cognats ayant plus de trois langues parmi les 12 langues Germaniques indiquées ci-dessus. Cette étape a été réalisée par un script écrit en Python version 3.2 (voir l'Appendice D, la section D.1). Ainsi, nous avons obtenu 248 arbres de cognats. Notons que plusieurs mots Indo-Européens avaient plus qu'une traduction en certaines langues. Par exemple, le mot « tree » en anglais peut être traduit par « tre » ou « tra » en Suédois. La première variante, « tre », est proche du

[1] http://www.worldstatesmen.org/Denmark.html

Riksmål et de l'Islandais, tandis que la deuxième variante, « tra », est proche du Danois et du Féroïen. Pour conserver toutes ces hypothèses d'évolution, nous avons dupliqué l'arbre d'évolution du cognat donné pour toutes les hypothèses différentes possibles relatives aux 12 langues étudiées (*c.-à-d.*, aux groupes Germaniques du Nord et de l'Ouest). Nous avons ainsi obtenu 264 arbres, ayant de 4 à 12 feuilles chacun.

Nous avons appliqué à ces données notre algorithme de construction de super-arbres multiples (voir l'Algorithme 5.2 du chapitre V), qui a retourné comme solution trois groupes et trois super-arbres correspondants (voir le tableau 6.5). Le premier groupe contenait 76 arbres, le deuxième groupe contenait 82 arbres et le troisième groupe contenait 106 arbres.

Tableau 6.5 Résultat d'application de notre algorithme de construction de super-arbres multiples à des arbres de langues Germaniques du Nord et de l'Ouest.

Groupe Linguistique	Caractéristiques		Nombre optimal de clusters selon *E-CH*
	Nombre de langues	Nombre d'arbres	L'approche de super-arbre
Germanique du Nord et Germanique de l'Ouest	12	264	3

Pour chacun des trois groupes retrouvés, nous avons utilisé le logiciel CLANN (Creevey et McInerney, 2004) pour inférer le super-arbre correspondant (voir la figure 6.9). Cette figure montre, par exemple, que le Sranan est d'un côté proche de l'Anglais (voir les figures 6.9a et 6.9c) et d'un autre côté proche du clade (Hollandais et Flamand) (voir la figure 6.9b). Les résultats semblables ont été obtenus par Bryant *et al.* (2005), Gray *et al.* (2010) et Willems *et al.* (2016), indiquant que le Sranan est une langue hybride de l'Anglais et de l'ancienne langue Néerlandaise.

Nous avons également trouvé que la langue PennDutch (l'Hollandais de la Pennsylvanie) est une langue hybride du clade composé de l'Anglais et du Sranan (voir les figures 6.9a et 6.9c) et de l'Allemand (voir la figure 6.9b). Les résultats très semblables ont été obtenus par Willems *et al.* (2016), montrant que le PennDutch est une langue hybride du groupe {Anglais et Sranan} et de l'Allemand.

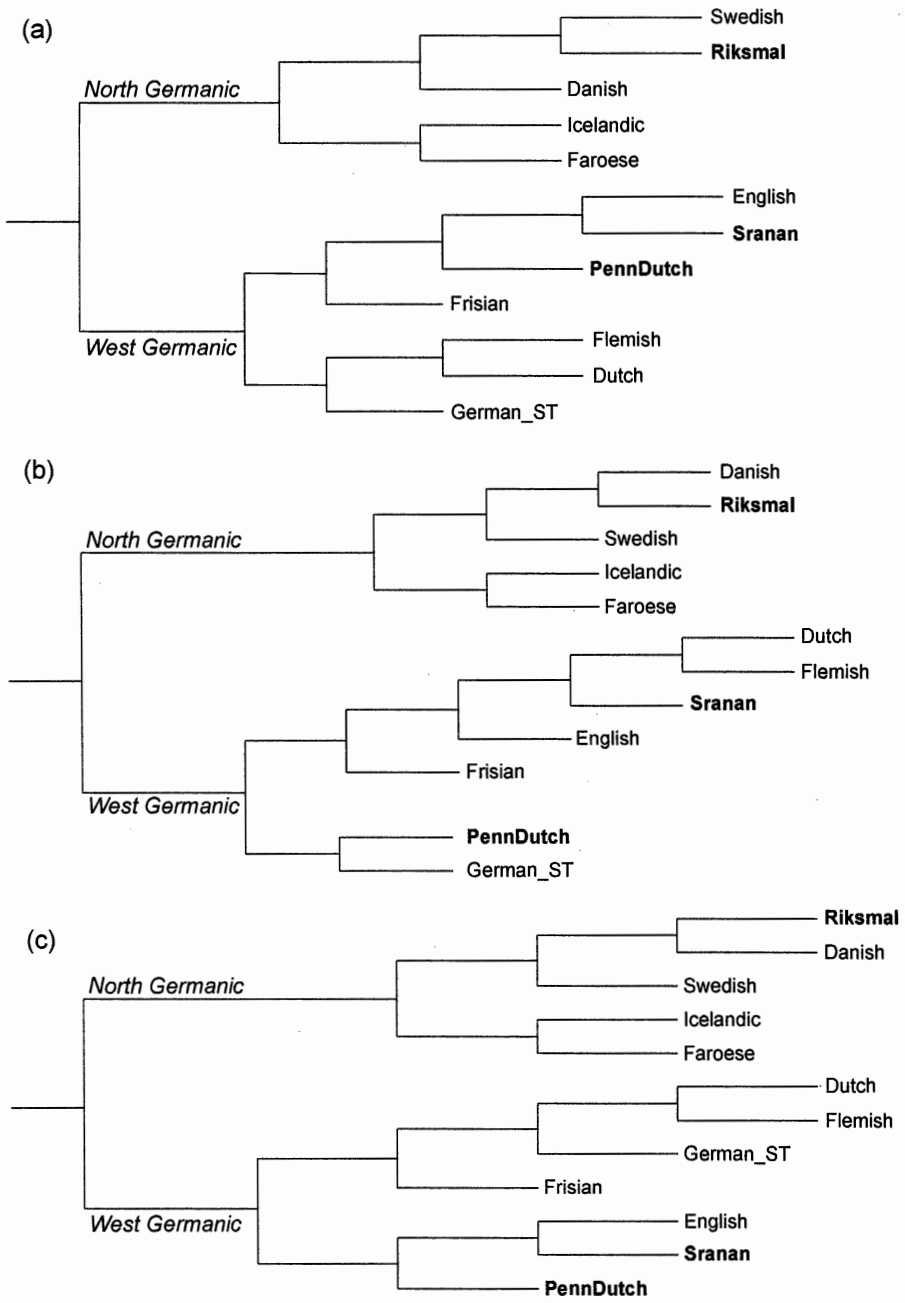


Figure 6.9 Les trois super-arbres obtenus pour 12 langues Indo-Européennes des groupes Germaniques du Nord et de l'Ouest. (a) le super-arbre obtenu à partir des arbres appartenant au premier cluster d'arbres de cognats retrouvé par notre algorithme; (b) le super-arbre obtenu à partir des arbres appartenant au deuxième

cluster d'arbres de cognats retrouvé par notre algorithme; et (c) le super-arbre obtenu à partir des arbres appartenant au troisième cluster d'arbres de cognats retrouvé par notre algorithme.

La méthode de décomposition de partitions (*Split Decomposition*) introduite par Bandelt et Dress en 1992 décompose la matrice de distances donnée en composantes simples, basées sur des divisions pondérées (*c.-à-d.*, des bipartitions de taxons, d'espèces ou de langues). Ces divisions peuvent alors être représentées à l'aide d'un graphe divisé ou un *split-graphe*. La méthode Neighbor-Net, introduite par Bryant et Moulton (2004) et implémentée dans le programme SplitsTree (Huson et Bryant, 2006), fonctionne de manière similaire à *Split Decomposition*, mais construit des réseaux phylogénétiques beaucoup plus résolus que ceux donnés par la méthode de Bandelt et Dress (1992). Les split-graphes ont été largement utilisés dans les études biologiques pour décrire les relations phylogénétiques réticulées entre les espèces, mais plusieurs travaux ont également considéré leurs applications en linguistique historique (Atkinson et Gray, 2005). Willems *et al.* (2016), par exemple, ont utilisé le logiciel SplitsTree pour inférer les split-graphes correspondant aux groupes Germanique du Nord (voir la figure 6.10a) et Germanique de l'Ouest (voir la figure 6.10b).

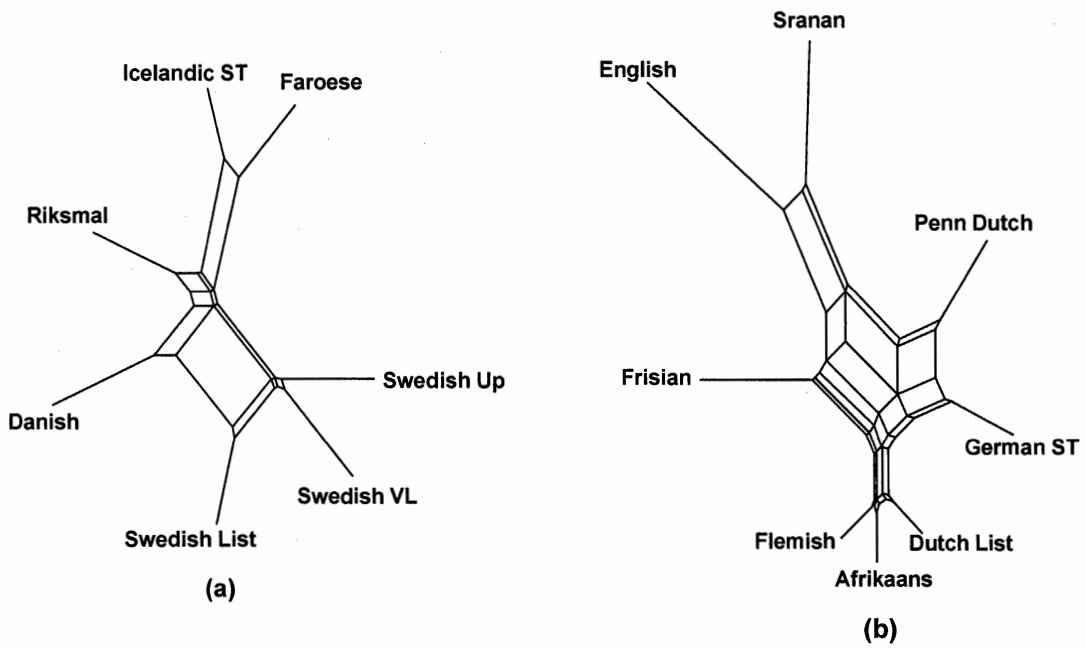


Figure 6.10 (a) Split-graphe pour le cluster Germanique du Nord constitué de sept langues (Willems *et al.*, 2016) et (b) split-graphe pour le cluster Germanique de l'Ouest constitué de huit langues (Willems *et al.*, 2016).

Nous avons également utilisé le logiciel SplitsTree (Huson et Bryant, 2006), pour visualiser l'évolution de l'ensemble d'arbres phylogénétiques appartenant au groupe Germanique du Nord en utilisant nos distances entre les langues. Le premier split-graphe (voir la figure 6.11a) a été obtenu à partir de l'ensemble d'arbres de cognats (pour les groupes Germaniques du Nord et de l'Ouest) que nous avons considérés. Le deuxième split-graphe (voir la figure 6.11b) a été inféré en utilisant les trois super-arbres obtenus par notre algorithme des *k*-moyennes basé sur le critère *E-CH* (voir le tableau 6.5 et la figure 6.9). Nous pouvons observer le changement de la position de la langue PennDutch (voir les figures 6.11a et 6.11b) ce qui confirme l'incertitude quant à sa position dans l'arbre d'évolution des langues Indo-Européennes.

Nous pouvons également observer plusieurs similarités entre les split-graphes obtenus par Willems *et al.* (2016, voir la figure 6.10), et ceux que nous avons retrouvés (voir la figure 6.11).

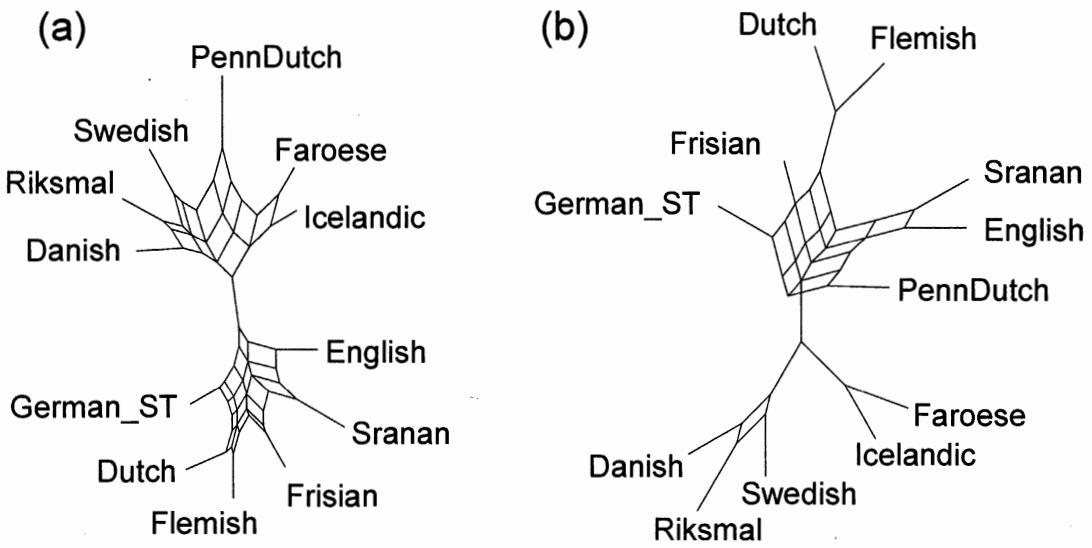


Figure 6.11 Deux split-graphes pour 12 langues Indo-Européennes des groupes Germaniques du Nord et de l'Ouest obtenus en utilisant le logiciel SplitsTree (Huson et Bryant, 2006); (a) le split-graphe obtenu à partir de l'ensemble des données linguistiques considérées et (b) le split-graphe obtenu à partir des trois super-arbres retournés par notre algorithme.

6.4.5 Application de notre algorithme des *k*-moyennes aux données de Stockham *et al.*

Dans cette section nous présentons les différents résultats obtenus en appliquant notre approche de partitionnement d'arbres par les *k*-moyennes aux quatre jeux de données examinés originellement par Stockham *et al.* (2002). Dans un premier temps, nous avons construit les heatmaps pour chacun de ces quatre jeux de données (voir la figure 6.12). Pour ce faire, nous avons calculé la distance topologique de Robinson et Foulds (*RF*) normalisée pour chaque paire d'arbres phylogénétiques donnés et pour

chaque jeu de données de Stockham *et al.* étudié. Le heatmap fait correspondre la distance *RF* normalisée à l'intensité d'une palette de couleurs sur une matrice à deux dimensions. La palette de couleurs s'étend des couleurs les plus chaudes aux couleurs les plus froides (voir le cercle chromatique[1]) selon la valeur de la distance *RF*.

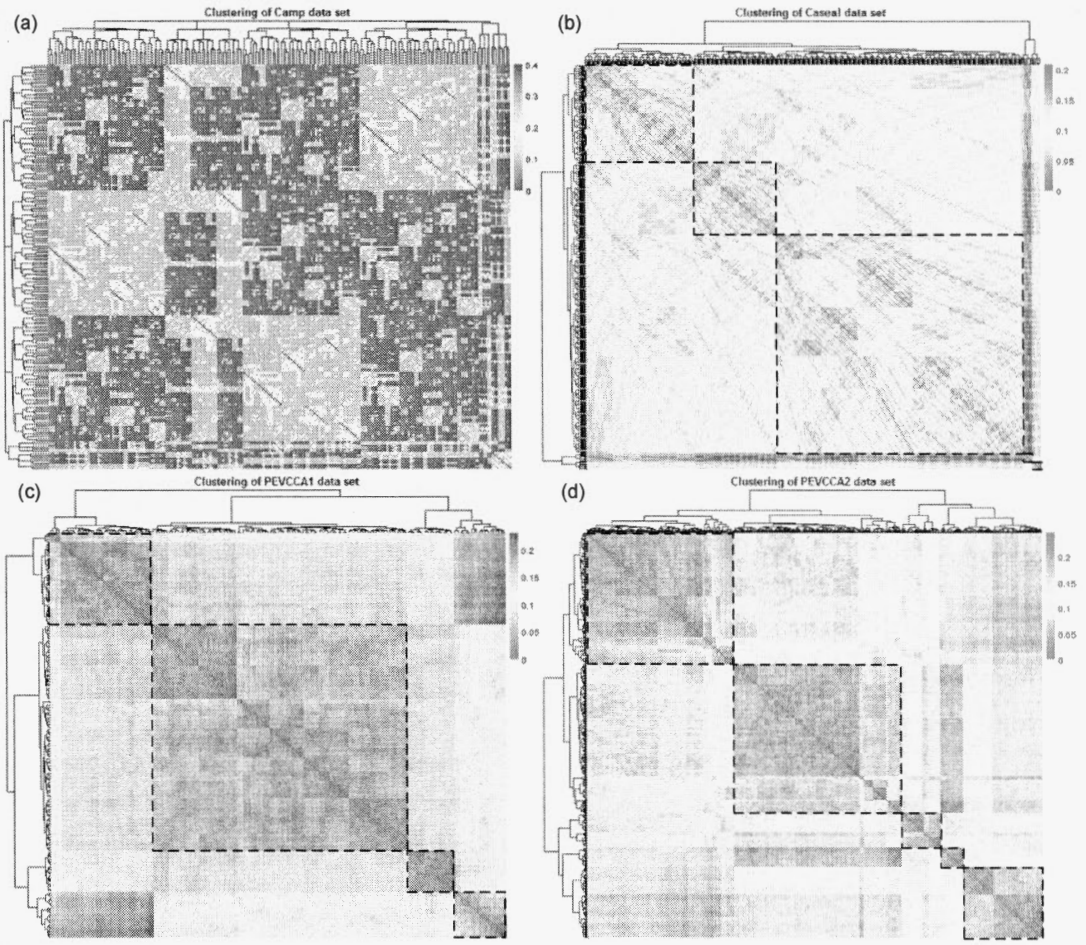


Figure 6.12 Les heatmaps pour les données de (a) Camp, (b) Caesal, (c) PEVCCA1 et (d) PEVCCA2, initialement examinées par Stockham *et al.* (2002). Ces heatmaps ont été obtenus à l'aide du package *pheatmap*[2] du langage R, version 3.2.2.

---

[1] https://fr.wikipedia.org/wiki/Cercle_chromatique

[2] https://cran.r-project.org/web/packages/pheatmap/pheatmap.pdf

Puis, dans un deuxième temps, nous avons appliqué nos algorithmes de partitionnement d'arbres avec les critères *E-CH* et *W* aux quatre jeux de données considérés : Camp, Caesal, PEVCCA1 et PEVCCA2.

6.4.5.1 Camp

Le premier jeu de données, Camp, contient 216 arbres avec 13 feuilles chacun. Stockham *et al.* (2002) ont suggéré qu'il existe seulement un seul cluster dans ce jeu de données parce que les densités calculées varient de façon irrégulière (Stockham, Wang et Warnow, 2002). Notre algorithme des *k*-moyennes de partitionnement d'arbres phylogénétiques, basé sur les critères *E-CH* et *W*, a été capable de retrouver la bonne solution dans ce cas. En effet, le critère *W* a fourni la solution avec un cluster unique et le critère *E-CH* a donné la solution avec deux clusters, ce qui est le minimum pour ce critère (voir le tableau 6.6). Ces résultats ont été confirmés par le heatmap correspondant (voir la figure 6.12a). Nous avons utilisé la notion de la spécificité introduite par Stockham *et al.* (2002) pour connaître la robustesse de l'arbre phylogénétique et le nombre de polytomies (*c.-à-d.*, nœuds non résolus dans l'arbre phylogénétique). La spécificité de l'arbre consensus général (obtenu avec le critère *W*) que nous avons inféré pour les données Camp était de 60%.

Tableau 6.6 Les spécificités pour chaque arbre consensus trouvé en utilisant les critères *W* et *E-CH* dans l'algorithme des *k*-moyennes pour le jeu de données Camp.

Jeu de données Camp : 216 arbres avec 13 feuilles		
Indice (cluster)	Nombre d'arbres phylogénétiques	Spécificité de consensus (Stockham, Wang et Warnow, 2002) en %
E-CH(1)	1	100.00
E-CH(2)	215	60.00
W(1)	216	60.00

6.4.5.2 Caesal

Le second jeu de données, Caesal, est constitué de 450 arbres phylogénétiques avec 51 feuilles chacun. L'algorithme de Stockham *et al.* (2002) et notre algorithme des *k*-moyennes basé sur le critère *E-CH* a trouvé la solution avec trois clusters. Le premier cluster contient 312 arbres phylogénétiques, le deuxième cluster contient 27 arbres phylogénétiques et le troisième cluster contient 111 arbres phylogénétiques. La figure 6.12b confirme la plausibilité de ces résultats. La spécificité du consensus général est de 77.08% dans ce cas, du premier cluster est de 87.50%, du deuxième cluster est de 79.17% et du dernier cluster est de 83.33% (voir le tableau 6.7).

Tableau 6.7 Les spécificités pour chaque arbre consensus trouvé en utilisant le critère *E-CH* dans l'algorithme des *k*-moyennes pour le jeu de données Caesal.

Jeu de données Caesal : 450 arbres avec 51 feuilles		
Indice (cluster)	Nombre d'arbres phylogénétiques	Spécificité de consensus (Stockham, Wang et Warnow, 2002) en %
E-CH(1)	312	87.50
E-CH(2)	27	79.19
E-CH(3)	111	83.33
Consensus général	450	77.08

6.4.5.3 PEVCCA1 et PEVCCA2

Le troisième et le quatrième jeux de données, PEVCCA1 et PEVCCA2, contiennent 168 et 654 arbres phylogénétiques, respectivement, avec 129 feuilles chacun.

PEVCCA1

Stockham *et al.* (2002) ont trouvé trois clusters pour ce jeu de données, tout comme notre algorithme des *k*-moyennes de partitionnement d'arbres basé sur le critère *E-CH* (voir le tableau 6.8). Le premier cluster trouvé contient 36 arbres phylogénétiques, le deuxième cluster trouvé contient 38 arbres phylogénétiques et le troisième cluster trouvé contient 94 arbres phylogénétiques. La spécificité du premier cluster est de 89.68% et celle du deuxième et du troisième clusters est de 92.06% (voir le tableau 6.8). La spécificité de l'arbre consensus général pour les données PEVCCA1 est de 76.98%.

Tableau 6.8 Les spécificités pour chaque arbre consensus trouvé en utilisant le critère *E-CH* dans l'algorithme des *k*-moyennes pour le jeu de données PEVCCA1.

Jeu de données PEVCCA1 : 168 arbres avec 129 feuilles		
Indice (cluster)	Nombre d'arbres phylogénétiques	Spécificité de consensus (Stockham, Wang et Warnow, 2002) en %
E-CH(1)	36	89.68
E-CH(2)	38	92.06
E-CH(3)	94	92.06
Consensus général	168	76.98

PEVCCA2

Dans l'étude de Stockham et *al.* (2002), les chercheurs ont trouvé cinq clusters pour PEVCCA2 (voir le tableau 6.8). Notre algorithme des *k*-moyennes a également trouvé cinq clusters en utilisant le critère *E-CH*. De plus, ces résultats peuvent être confirmés par le heatmap correspondant (voir la figure 6.12d). La spécificité du consensus général pour ce jeu de données est de 72.22%. Le critère *E-CH* a donné une solution avec 29 arbres phylogénétiques pour le premier cluster, 200 arbres phylogénétiques

pour le deuxième cluster, 241 arbres phylogénétiques pour le troisième cluster, 70 arbres phylogénétiques pour le quatrième cluster et 114 arbres phylogénétiques pour le cinquième cluster. La spécificité du premier et du deuxième cluster est de 87.30%, du troisième cluster est de 84.13%, du quatrième cluster est de 86.51% et du cinquième cluster est de 92.06% (voir le tableau 6.9). La spécificité de l'arbre consensus général pour les données PEVCCA2 est de 72.22%.

Tableau 6.9 Les spécificités pour chaque arbre consensus trouvé en utilisant le critère *E-CH* dans l'algorithme des *k*-moyennes pour le jeu de données PEVCCA2.

Jeu de données PEVCCA2 : 654 arbres avec 129 feuilles		
Indice (cluster)	Nombre d'arbres phylogénétiques	Spécificité de consensus (Stockham, Wang et Warnow, 2002) en %
E-CH(1)	29	87.30
E-CH(2)	200	87.30
E-CH(3)	241	84.13
E-CH(4)	70	86.51
E-CH(5)	114	92.06
Consensus général	654	72.22

6.5 Conclusion

Dans ce chapitre, nous avons présenté les résultats pour un ensemble de jeux de données réelles composées des données biologiques et linguistiques. Entre autres, nous avons pu valider nos nouveaux algorithmes sur des jeux de données tests avec des résultats connus (*p.ex.* les données de Stockham *et al.* 2002).

Deux phénomènes d'évolution réticulée majeurs influencent l'histoire des espèces : le transfert horizontal de gènes et l'hybridation. La compréhension que ces deux phénomènes pourraient jouer un rôle clef dans l'évolution des espèces est l'un des changements les plus fondamentaux dans notre perception générale de la biologie moléculaire (Makarova *et al.* 1999). Dans ce chapitre, nous avons montré que nos

algorithmes de partitionnement d'arbres peuvent être utilisés pour la détection des transferts horizontaux de gènes (voir les résultats pour les données d'archées), de même que pour la recherche d'hybrides (voir les résultats pour les données linguistiques et la détection des langues hybrides).

Dans le futur, il serait intéressant d'ajouter aux différentes méthodes de partitionnement d'arbres un vecteur de poids durant le calcul des distances pour prendre ainsi en compte la robustesse statistique de chaque arbre phylogénétique fourni en entrée.

CHAPITRE VII

CONCLUSIONS ET PERSPECTIVES

«Cette nuit, en regardant le ciel, je suis arrivé à la conclusion qu'il y a beaucoup plus d'étoiles qu'on en a besoin.», Quino Cabrera, 1964-1973 (Mafalda).

7.1 Conclusions

Les arbres phylogénétiques consensus apportent des informations importantes sur l'évolution conjointe des familles de gènes considérées. Les types d'arbres consensus les plus connus sont l'arbre consensus strict, l'arbre consensus majoritaire et l'arbre consensus majoritaire étendu. Un arbre consensus d'espèces fiable ne peut pas être inféré d'un alignement de séquences multiples d'une famille de gènes unique ou de la concaténation d'alignements correspondant à des familles de gènes ayant des histoires évolutives différentes. Ces histoires évolutives peuvent différer en raison de transferts horizontaux de gènes ou d'anciennes duplications de gènes qui provoquent l'émergence de paralogues dans un génome. De nombreuses méthodes ont été proposées pour inférer un arbre consensus unique à partir d'une collection d'arbres de gènes. Pourtant, l'application de ces méthodes de fusion d'arbres peut conduire à la perte d'informations évolutives spécifiques qui caractérisent certaines familles de gènes ou certains groupes de familles de gènes. Ainsi, la problématique d'inférence d'arbres consensus multiples devient pertinente.

Les super-arbres sont des arbres consensus d'espèces qui sont généralement assemblés à partir d'un ensemble d'arbres phylogénétiques de tailles distinctes, qui ont été inférés en utilisant différentes familles de gènes ou différents ensembles de données (*p.ex.* morphologiques ou moléculaires). Les arbres phylogénétiques d'entrée pour un algorithme d'inférence de super-arbres ont généralement des ensembles de feuilles (*c.-à-d.*, espèces) différents, mais chevauchants.

Dans notre thèse, la problématique de la fusion d'arbres a été subdivisée en deux cas d'analyse. Le premier cas d'analyse concerne les arbres définis sur un même ensemble de feuilles (le cas d'arbres consensus uniques ou multiples). Le deuxième cas considère les arbres définis sur des ensembles de feuilles différents, mais chevauchants (le cas de super-arbres multiples). L'avantage principal de notre approche par rapport à l'approche classique, qui retourne toujours un arbre consensus unique comme solution, est qu'elle propose comme solution un ou plusieurs arbres consensus, dépendamment des topologies d'arbres phylogénétiques fournies en entrée.

Au chapitre III, nous avons présenté une nouvelle heuristique pour inférer des arbres consensus multiples en utilisant l'algorithme des *k*-médoïdes et en proposant de nouveaux critères de validité des clusters spécifiques à notre problématique. La métrique de Robinson et Foulds (*RF*) a été utilisée pour mesurer la distance entre les arbres. Nous avons adapté à l'algorithme des *k*-médoïdes de partitionnement d'un ensemble d'arbres les indices de validité des clusters populaires de Caliński-Harabasz (*CH*) et de Silhouette (*SH*). L'indice *SH* s'est montré plus performant que l'indice *CH* dans le cadre des *k*-médoïdes adaptés au partitionnement d'arbres. Nous avons également testé deux variantes de la distance topologique de Robinson et Foulds, avec et sans sa mise au carré. Nous avons trouvé que la variante de l'algorithme utilisant la distance *RF* non mise au carré était plus performante que la variante utilisant sa version quadratique.

Au chapitre IV, nous avons repris la même problématique que celle analysée au chapitre III, mais en employant l'algorithme des *k*-moyennes pour partitionner un ensemble d'arbres en un ou en plusieurs clusters. Nous avons également proposé un nouveau critère de validité des clusters, le critère *W*, et adapté au problème de partitionnement d'arbres le critère *Gap*. Ces deux critères permettent de comparer des solutions avec des arbres consensus uniques et multiples. Nous avons trouvé que l'utilisation de l'approximation euclidienne de la fonction objective des *k*-moyennes conduit à de très bons résultats.

Au chapitre V, nous avons présenté une nouvelle version spécifique de l'algorithme des *k*-moyennes afin de l'étendre aux cas des super-arbres. Nous avons également obtenu quelques résultats théoriques importants concernant les bornes, inférieure et supérieure, de la fonction objective des *k*-moyennes. L'un des principaux avantages de notre algorithme est qu'il est beaucoup plus rapide que les approches de partitionnement d'arbres existantes. Cela le rend particulièrement bien adapté à l'analyse de grands ensembles de données génomiques.

Au chapitre VI, nous avons appliqué les différents algorithmes décrits dans les chapitres précédents à des jeux de données réelles. Nous avons varié nos types de données, allant des données biologiques aux données linguistiques, afin de montrer tous les champs d'application de nos méthodes.

7.2 Les contributions principales

- ✓ Analyse, conception et implémentation de nouveaux algorithmes, basés les *k*-médoïdes et *k*-moyennes, pour le partitionnement d'arbres phylogénétiques et la construction d'arbres consensus uniques ou multiples.

- ✓ Analyse, conception et implémentation d'un nouvel algorithme pour le partitionnement d'arbres phylogénétiques de tailles différentes et la construction de super-arbres multiples.

- ✓ Adaptation de trois critères de validité des clusters (*c.-à-d.*, l'indice Silhouette, l'indice de Caliński-Harabasz et la statistique *Gap*) au cas de partitionnement d'arbres phylogénétiques.

- ✓ Conception de nouveaux critères de validité des clusters : 1) le critère *W* dans le cas des données homogènes et 2) les bornes de l'indice de Caliński-Harabasz dans le cas des données hétérogènes.

- ✓ Présentation de quelques résultats théoriques intéressants concernant la fonction objective des *k*-moyennes adaptée au cas de partitionnement d'arbres et la distance topologique de Robinson et Foulds.

- ✓ Mise en place d'un protocole de simulations permettant de tester des algorithmes de partitionnement d'arbres phylogénétiques.

7.3 Perspectives

Dans cette thèse, nous avons décrit de nouveaux algorithmes de partitionnement d'arbres phylogénétiques. Cette problématique a été subdivisée en deux axes principaux, à savoir le cas des arbres consensus et le cas des super-arbres. Nous avons également développé des critères spécifiques concernant les données homogènes et hétérogènes. Cependant, il serait aussi intéressant d'ajouter aux différentes méthodes de partitionnement d'arbres un vecteur de poids durant le calcul des distances pour prendre ainsi en compte la robustesse statistique de chaque arbre phylogénétique fourni en entrée. Dans une autre perspective, nous pourrions développer un algorithme qui cherche le vrai centroïde d'un ensemble d'arbres, dans le sens de la

distance topologique de Robinson et Foulds (*RF*). Ce centroïde ne sera pas nécessairement associé à un arbre phylogénétique.

Dans cette thèse, nous avons utilisé la distance topologique de Robinson et Foulds pour comparer les arbres phylogénétiques. Plusieurs autres mesures ont été également proposées pour estimer la distance entre les arbres phylogénétiques. Les plus populaires d'entre elles sont la distance de quartets (Bryant *et al.*, 2000), la distance SPR (*Subtree Prune and Regraph*, Bruen et Bryant, 2008), la distance MAST (*Maximum Agreement SubTree*) et la dissimilarité de bipartitions (Makarenkov *et al.* 2007). Malheureusement, la distance SPR, qui est souvent utilisée pour identifier les événements de transferts horizontaux de gènes, prend un temps exponentiel pour être calculée. C'est aussi le cas de la distance MAST, qui ne peut être calculée en temps polynomial. La distance *RF* que nous avons utilisée et celle de quartets sont des distances topologiques parmi les plus rapides à calculer. En effet, ces deux distances peuvent être calculées en $O(n^2)$ lorsque deux chaînes Newick, représentant deux arbres phylogénétiques avec n feuilles, sont considérées. La distance de quartets (Bryant *et al.*, 2000) pourrait donc également être utilisée, à la place de la distance de Robinson et Foulds, dans les algorithmes de partitionnement d'arbres phylogénétiques.

Finalement, il serait aussi intéressant de développer un algorithme de prédiction des classes d'arbres en utilisant l'approche de réseaux neuronaux convolutifs (*Convolutional Neural Network* ou CNN). La supériorité de l'apprentissage profond (*Deep Learning* ou DL) est maintenant reconnue. Il serait donc important d'étudier les relations phylogénétiques (via la distance entre les arbres) en considérant un CNN dans l'architecture DL. Une telle méthode pourrait être basée sur la distance topologique de Robinson et Foulds entre les paires d'arbres phylogénétiques donnés en entrée.

APPENDICE A

ARTICLES PUBLIÉS OU ACCEPTÉS POUR PUBLICATION DANS LE CADRE DU PROJET DOCTORAL

Tahiri, N., Willems, M. Boc, A., et Makarenkov, V. (2012). Classification des langues Indo-Européennes basée sur un modèle d'identification de transferts horizontaux de gènes. Actes des XIXèmes Rencontre de la Société Francophone de Classification, ECM et LIF de Marseille, France.

Tahiri, N., Willems, M. et Makarenkov, V. (2014). Classification d'arbres phylogénétiques basée sur l'algorithme des *k*-moyennes. Actes des XXIèmes Rencontre de la Société Francophone de Classification, CNRST et ENSA de Rabat, Maroc, pages 49-54.

Willems, M., **Tahiri, N.** et Makarenkov, V. (2014). A new efficient algorithm for inferring explicit hybridization networks following the Neighbor-Joining principle, Journal of Bioinformatics and Computational Biology, 12(5), DOI: 10.1142/S0219720014500243.

Tahiri, N., Willems, M. et Makarenkov, V. (2015). A new fast method for building multiple consensus trees using k-means. Actes de la Première Rencontre de la Société Marocaine de Classification, ENSA de Tanger, Maroc.

Badescu, D., **Tahiri, N.** et Makarenkov, V. (2015). A new fast method for detecting and validating horizontal gene transfer events using phylogenetic trees and

aggregation functions, Pattern Recognition in Computational Molecular Biology: Techniques and Approaches, 483-504, John Wiley & Sons, Inc.

Tahiri, N., Willems, M. et Makarenkov, V. (2017). Inférence de super-arbres phylogénétiques multiples en utilisant l'algorithme des *k*-moyennes. Actes des XXIIèmes Rencontre de la Société Francophone de Classification, MSH Guépin de Nantes, France, pages 110-114.

De Amorim, R.C., **Tahiri, N.**, Mirkin, B. et Makarenkov, V. (2017). A median-based consensus rule for distance exponent selection in the framework of intelligent and weighted Minkowski clustering. Weihs (Eds) proceedings of IFCS 2015, Data Science - Innovative Developments in Data Analysis and Clustering, Springer Verlag, Cham, pages 97-110.

Tahiri, N., Willems, M., et Makarenkov, V. (2018). A new fast method for inferring multiple consensus trees using *k*-medoids. Accepté pour publication dans BMC Evolutionary Biology.

Classification des langues Indo-Européennes basée sur un modèle d'identification de transferts horizontaux de gènes

Nadia Tahiri*, Matthieu Willems*,
Alix Boc* et Vladimir Makarenkov*

* Département d'informatique
Université du Québec à Montréal
Case postale 8888, Succursale Centre-ville
Montréal (Québec), H3C 3P8, Canada
makarenkov.vladimir@uqam.ca

Résumé. Dans cet article, nous présentons une nouvelle approche permettant de modéliser l'évolution des langues Indo-Européennes à l'aide d'un modèle phylogénétique en réseau. L'avantage de notre approche par rapport à l'approche classique, qui utilise la topologie d'un arbre phylogénétique (i.e., additif ou X-arbre) pour représenter l'évolution des langues naturelles, est que le modèle en réseau, basé sur un algorithme de détection de transferts horizontaux de gènes, permet aussi d'identifier des emprunts lexicaux survenus en cours de l'évolution et d'estimer le taux d'échanges horizontaux entre différents groupes des langues.

1 Introduction

De nombreux parallèles entre le processus d'évolution en linguistique et l'évolution biologique selon Darwin ont été observés. Atkinson et Gray (2005) ont présenté un tableau des parallèles conceptuels les plus importants caractérisant l'évolution biologique et linguistique. D'importantes études ont considéré des méthodes phylogénétiques et leurs applications aux données linguistiques (Gray et Atkinson, 2003; Rexová *et al.*, 2003; Atkinson et Gray, 2005). Un des domaines largement étudié en biolinguistique demeure l'évolution des langues Indo-Européennes (IE) (Gray et Atkinson, 2003; Rexová *et al.*, 2003).

Malheureusement, ni les arbres phylogénétiques, ni les split-graphes (Bandelt et Dress, 1992) ne peuvent être utilisés pour prédire et représenter le phénomène d'emprunt lexical. Par exemple, les split-graphes peuvent être utilisés pour visualiser les caractéristiques hybrides de la langue créole (Atkinson et Gray, 2005), mais ne peuvent pas dresser explicitement un portait des évènements d'emprunts de mots, qui sont équivalents à des transferts horizontaux de gènes (THG) en biologie. Le THG est un des mécanismes majeurs contribuant à la diversification des génomes microbiens (Doolittle, 1999).

Les principales suppositions pour la représentation de l'évolution des langues naturelles en arbre sont les suivantes : l'évolution des langues a été strictement divergente, chaque langue a été transmise en entier et la fréquence d'emprunts lexicaux (i.e., transmission horizontale de mots individuels) entre les langues a été faible. L'arbre d'évolution des langues résume l'histoire de mots individuels, mais souligne seulement la tendance verticale de l'évolution des langues bien que, pour certaines langues, le niveau d'emprunts puisse être très significatif. Par exemple, l'anglais est une langue germanique, mais il a emprunté environ 50% de son lexique du français et du latin (Pagel, 2000).

Dans cet article nous raffinons le travail commencé dans Boc et al. (2010b) en utilisant la base de données de Dyen *et al.* (1997) modifiée et en appliquant une version améliorée de l'algorithme de détection des échanges lexicaux.

2 Description des données

La base de données constituée par Dyen *et al.* (1997) inclut 200 mots de la liste Swadesh (1952). La liste Swadesh est une des diverses listes de mots de signification basique, constituée par M. Swadesh dans les années 1940-50, qui est couramment utilisée en lexicostatistique (i.e., évaluation quantitative de la parenté des langues) et en glottochronologie (i.e., datation des divergences entre les langues). Pour chacun des 200 mots de la liste Swadesh, la base de données originale contient les traductions utilisées dans 95 variétés de langues (87 d'entre elles ont été considérées dans notre étude) regroupées par Dyen *et al.* (1997) en ensembles de cognats (i.e., groupes de mots apparentés ayant une racine commune). Le groupement par parenté a été faite seulement pour les traductions ayant le même sens (Dyen *et al.*, 1997). Deux traductions dans deux langues différentes étaient identifiées comme apparentées si pour ces deux langues, elles avaient une histoire évolutive ininterrompue depuis une forme ancestrale commune. Les traductions connues pour être reliées par des emprunts ou par des similarités accidentelles étaient mises dans une classe séparée. Dans un petit nombre de cas, il était difficile de différencier les cognats des emprunts ou des similarités accidentelles ; de telles traductions ont été classifiées comme cognats douteux. Cette base de données a été utilisée par Gray et Atkinson (2003) pour inférer l'arbre d'évolution des langues IE. Nous avons modifié des ensembles originaux de cognats en leur ajoutant les mots empruntés connexes. Dans certains cas, des mauvais cognats ont été supprimés. Aussi, quand c'était approprié, de nouveaux ensembles, composés de cognats douteux, ont été créés.

3 Méthode et résultats

Une version adaptée de l'algorithme de détection des THG (Boc *et al.*, 2010a) a été appliquée dans cette étude pour inférer un réseau phylogénétique des langues IE. Un nombre important de nouvelles caractéristiques a été ajouté à l'algorithme de base pour le rendre applicable à l'identification des emprunts de mots. Quand l'algorithme de base est exécuté dans un contexte biologique, il identifie les THG d'un gène donné pour un ensemble d'espèces considérées, réconciliant ainsi les arbres phylogénétiques d'espèces et de gène. À chaque étape du processus de réconciliation, un ensemble de THG compatibles est inféré. En établissant les parallèles entre les processus d'identification des THG et des emprunts de mots, l'arbre des langues IE (Gray et Atkinson, 2003) joue le rôle de l'arbre d'espèces et l'arbre de mot, représentant l'évolution d'un cognat particulier, joue le rôle de l'arbre de gène. La procédure algorithmique incluait les trois principales étapes décrites ci-dessous :

Étape 1. Soit L l'arbre enraciné de 87 langues IE inféré par Gray et Atkinson (2003, Fig. 1). Nous avons considéré les 200 mots de la liste Swadesh. Pour chacun de ces mots nous disposons les traductions en 87 langues. Ces traductions ont été regroupées par les linguistes en 1484 cognats (Dyen et al., 1997); les mots du même cognat sont supposés avoir une racine commune. Toutes les traductions ont été présentées par les lettres de l'alphabet latin. Pour chaque cognat, c, nous avons calculé la matrice de distances, $\mathbf{W}_c$, incluant les distances entre les traductions incluses dans c. La distance entre les traductions i et j dans c était calculée à l'aide de la formule suivante :

$$Normalized\_LD = \frac{Modified\_Levenshtein\_distance(i,j)}{length(i)+length(j)}. \quad (1)$$

Cette formule utilise une version modifiée de la distance de Levenshtein (1966), qui est normalisée par les longueurs des traductions i et j.

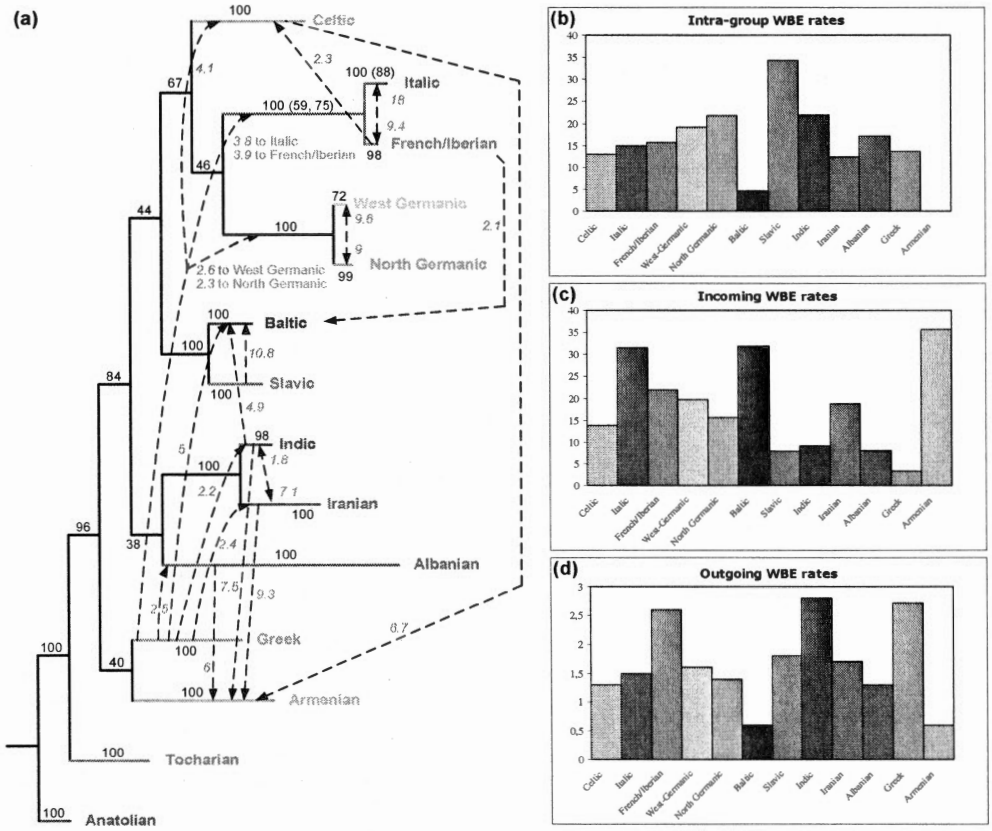


FIG. 1 – *(a) Résultats obtenus pour le total de 200 mots de la liste Swadesh. Par exemple, 18% des mots du groupe Italic ont une origine French/Iberian. Trois autres diagrammes montrent : (b) le taux d'emprunts à l'intérieur des groupes, (c) le taux d'emprunts pour des mots provenant d'autres groupes et (d) le taux d'emprunts pour des mots sortant vers d'autres groupes.*

L'arbre de langues réduit *L_c*, correspondant à l'ensemble de cognats *c*, a été obtenu en supprimant d'abord toutes les arêtes de *L* relatives aux langues absentes dans *c* et puis en éliminant tous les nœuds de degré 2. Les arbres *W_c* et *L_c* avaient donc le même nombre de feuilles, ce qui était nécessaire pour l'exécution de l'algorithme de détection des THG. Un arbre de langues *L_c* pouvait contenir des multifurcations (i.e., des nœuds internes de degré supérieur à 3) dans le cas où une ou plusieurs langues avaient de multiples traductions incluses dans l'ensemble de cognats *c*.

Étape 2. Nous avons appliqué l'algorithme de détection de THG (Boc *et al.*, 2010a) pour inférer les emprunts de mots, en considérant l'arbre de langues réduit *L_c* comme l'arbre d'espèces et l'arbre de mot *W_c* comme l'arbre de gène. L'algorithme a été exécuté 1484 fois et 1484 scénarios d'emprunts lexicaux (un scénario par cognat) ont été identifiés. L'algorithme appliqué reposait sur le critère de dissimilarité de bipartitions (Boc *et al.*, 2010a) et recherchait un scénario de coût minimal d'opérations SPR (*Subtree Prune and Regraft*), qui sont nécessaires pour transformer dans chaque cas l'arbre de langues en l'arbre de mot. Une version spécifique de l'algorithme, permettant d'utiliser des arbres de mots non-enracinés a été développée.

Étape 3. Ici, nous avons combiné tous les résultats obtenus provenant des 1484 scénarios d'emprunts lexicaux pour calculer les statistiques d'emprunt. Si plusieurs groupes de langues étaient impliqués dans l'emprunt d'un mot, les transferts obtenus étaient pondérés tenant compte de toutes les traductions impliquées. Nous avons donc estimé, les taux d'emprunts lexicaux pour les 12 groupes principaux de

langues IE, puis les pourcentages de mots affectés par un emprunt dans chaque groupe. Les taux de mots empruntés entrants et sortants ont été aussi calculés.

Les emprunts de mots les plus fréquents identifiés par une version spécifique de notre algorithme de détection de THG ont été ajoutés à l'arbre de langues pour représenter les échanges lexicaux les plus importants qui se sont produits durant l'évolution des langues IE (Fig. 1a). Si la proximité géographique peut expliquer la plupart des échanges fréquents (e.g., entre les groupes Germaniques du nord et de l'ouest ou entre les groupes Français/Iberian et Italique), certains parmi eux arrivent entre des groupes éloignés (e.g., entre les groupes Grec et Baltique ou Indic et Arménien). Les diagrammes (Fig. 1b, c et d) illustrent respectivement les taux d'emprunt lexicaux intra-groupe, entrant et sortant. Selon notre estimation, 35,4% des traductions IE considérées ont été affectées par des emprunts, incluant 15,5% découlant de groupes distincts.

Références

Atkinson, Q.D. et R.D. Gray. (2005). Curious parallels and curious connections - phylogenetic thinking in biology and historical linguistics. *Systematic Biology*, 54:513-526.

Bandelt, H-J et A.W.M. Dress. (1992). Split decomposition: a new and useful approach to phylogenetic analysis of distance data. *Molecular Phylogenetics and Evoution*, 1:242-252.

Boc, A., H. Philippe et V. Makarenkov. (2010a). Inferring and validating horizontal gene transfer events using bipartition dissimilarity, *Systematic Biology*, 59:195-211.

Boc, A., A-M. Di Sciullo et V. Makarenkov. (2010b). Classification of the Indo-European languages using a phylogenetic network approach. *Classification as a Tool for Research*, H. Locarek-Junge, C. Weihs (Eds), IFCS 2009, Springer, Berlin-Heidelberg-New York, 647-655.

Doolittle, W.F. (1999). Phylogenetic classification and the universal tree. *Science*, 284:2124-29.

Dyen, I., J.B. Kruskal, et P. Black. (1997). Fichier sur les langues Indo-Européennes (IE-DATA1) disponible à : http://www.ntu.edu.au/education/langs/ielex/IE-DATA1l.

Gray, R.D. et Q.D. Atkinson. (2003). Language-tree divergence times support the Anatolian theory of Indo-European origin. *Nature*, 426:435-439.

Levenshtein, V.I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10:707-710.

Rexová, K., D. Frynta et J. Zrzavý. (2003). Cladistic analysis of languages: Indo-European classification based on lexicostatistical data. *Cladistics*, 19:120-127.

Swadesh, M. (1952). Lexico-statistic dating of prehistoric ethnic contacts. *Proceedings of the American Philosophical Society*, 96:452-463.

Classification d'arbres phylogénétiques basée sur l'algorithme des *k*-moyennes

Nadia Tahiri*, Matthieu Willems*, Vladimir Makarenkov*

* Département d'informatique
Université du Québec à Montréal
Case postale 8888, Succursale Centre-ville
Montréal (Québec), H3C 3P8, Canada
makarenkov.vladimir@uqam.ca

Résumé. Nous présentons un nouvel algorithme permettant d'affiner l'inférence d'arbres consensus obtenus par regroupement d'arbres phylogénétiques (i.e., arbres additifs ou *X*-arbres). L'avantage de notre approche par rapport à l'approche classique, qui retourne toujours un arbre consensus unique, est que notre algorithme propose comme solution un ou plusieurs arbres consensus dépendamment des topologies d'arbres fournies en entrée. Nous utilisons l'algorithme des *k*-moyennes pour obtenir le partitionnement optimal de l'ensemble des arbres considérés.

1 Introduction

L'avènement et la mise en place du projet « Tree of Life » (ToL) [1] visant la reconstruction de l'arbre phylogénétique du vivant a nécessité la collaboration de plusieurs équipes de recherche à travers le monde. L'approche adoptée par ToL consiste à réduire récursivement le problème de reconstruction de l'Arbre de Vie en plusieurs sous-problèmes puis à fusionner les résultats. Cette stratégie implique deux difficultés majeures lors de la fusion des sous ensembles d'arbres phylogénétiques : 1) l'inférence d'un arbre phylogénétique consensus et 2) la vérification de la compatibilité évolutive des arbres phylogénétiques obtenus.

Dans cette étude nous nous sommes intéressés à la première problématique. De nombreux algorithmes ont été proposés pour construire un arbre consensus [2] à partir d'un ensemble d'arbres phylogénétiques. Trois algorithmes prédominent : l'algorithme du consensus strict, l'algorithme du consensus majoritaire et l'algorithme du consensus majoritaire étendu. L'arbre consensus strict contient seulement les bipartitions qui sont communes à tous les arbres donnés. L'arbre consensus majoritaire contient les arêtes qui sont présentes dans plus que 50% des arbres donnés, bien que d'autres pourcentages puissent aussi être considérés. L'arbre consensus majoritaire étendu contient toutes les bipartitions majoritaires auxquelles sont ajoutées à tour de rôle les bipartitions compatibles restantes en commençant par les bipartitions les plus fréquentes pour l'ensemble des arbres donnés. L'arbre consensus majoritaire étendu est le plus fréquemment utilisé en classification car il est toujours le mieux résolu parmi les trois types d'arbres consensus mentionnés.

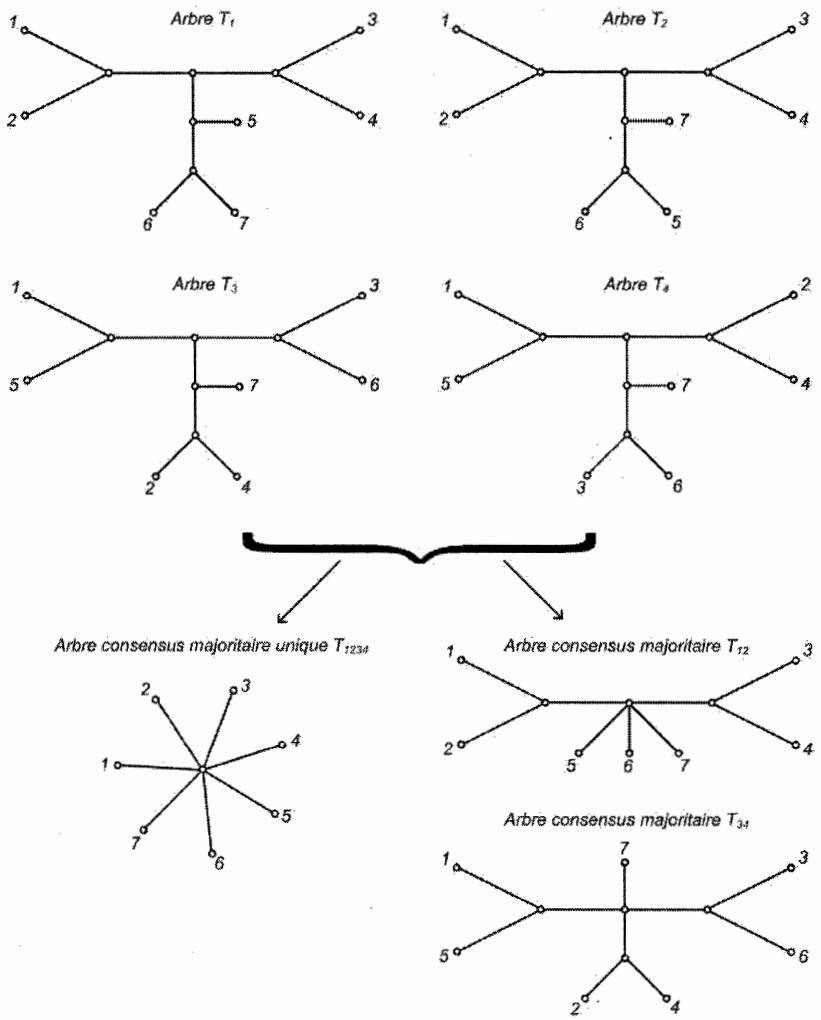


FIG. 1 – Quatre arbres phylogénétiques T_1, T_2, T_3 et T_4 définis sur un ensemble de sept feuilles ; leur arbre consensus majoritaire unique T_{1234} ; la solution à deux arbres consensus majoritaires T_{12} et T_{34}.

Ce qui est commun à la plupart des algorithmes de consensus classiques est l'unicité de l'arbre consensus fourni en sortie de l'algorithme. Cependant, dans plusieurs situations pratiques il est plus judicieux de prévoir le cas où plusieurs arbres consensus sont fournis par l'algorithme de classification. En biologie, il s'avère souvent hasardeux de regrouper des arbres phylogénétiques construits à partir de plusieurs gènes différents. Chaque gène traduit une histoire évolutive qui lui est propre et qui peut être très différente des autres histoires. Par exemple, certains gènes peuvent subir des transferts horizontaux (qui sont souvent les mêmes) et leur histoire évolutive sera représentée par des arbres phylogénétiques distincts des arbres de gènes non affectés par des transferts. Dans cet article, nous proposons une approche alternative d'inférence d'arbres consensus permettant d'obtenir un ou plusieurs arbres consensus en sortie de l'algorithme. La figure (1) illustre le cas de quatre arbres phylogénétiques T_1, T_2, T_3 et T_4 définis sur un ensemble de sept feuilles. Ici, la solution à deux arbres consensus majoritaires, T_{12} et T_{34}, semble être plus appropriée que celle à l'arbre consensus majoritaire unique T_{1234} proposée par l'approche classique.

L'idée d'inférer plusieurs arbres consensus a été initialement formulée par Maddison [3], qui a trouvé que les arbres consensus de certains sous-ensembles d'arbres sont souvent très différents et ont une meilleure résolution que l'arbre consensus de l'ensemble des arbres donnés. Plus récemment, Guénoche

[4] a introduit une nouvelle méthode pour déterminer si la représentation d'un ensemble d'arbres phylogénétiques doit se faire par un seul ou plusieurs arbres consensus majoritaires. Après avoir testé sa méthode sur des données réelles (la bactérie *E.Coli*) et artificielles, Guénoche a conclu qu'avec un ensemble d'arbres phylogénétiques homogènes (i.e., le cas des données réelles), l'arbre consensus unique était la solution la plus adéquate. Par contre, pour un ensemble d'arbres phylogénétiques hétérogènes, qui est caractéristique des arbres de gènes ayant des histoires évolutives différentes, le meilleur score n'est presque jamais atteint par un arbre consensus unique.

Nous proposons ici un nouvel algorithme permettant de regrouper les arbres phylogénétiques en se basant sur deux principaux critères : le critère de Calinski-Harabasz (*CH*) [5, 6] et le critère de Ball-Hall (*BH*) [7, 8] incorporés dans l'algorithme des *k*-moyennes.

2 Méthodologie

Le nouvel algorithme prend en entrée un ensemble d'arbres phylogénétiques définis sur le même ensemble de feuilles et retourne en sortie le partitionnement optimal de ces arbres en un ou plusieurs groupes. Pour chaque groupe retrouvé, l'algorithme retourne la liste de ses éléments (i.e., arbres phylogénétiques) et l'arbre-consensus qui s'y réfère. La sortie s'accompagne également des résultats statistiques (les indices :*W*, *CH* et *BH*). L'algorithme proposé utilise la méthode des k-moyennes adaptée à des données d'arbres.

La classification selon la méthode des k-moyennes procède par le partitionnement des données en *k* groupes selon un critère de similarité choisi [9]. Le problème de la recherche d'un partitionnement optimal selon le critère des *k*-moyennes est NP-difficile, ce qui explique l'existence de nombreuses variantes heuristiques de cette méthode. Nous avons utilisé la version des *k*-moyennes implémentée par Makarenkov et Legendre [10].

Soit Π l'ensemble de N arbres phylogénétiques définis sur un même ensemble de feuilles (i.e., éléments ou espèces). Nous recherchons le nombre optimal K ($1 \leq K \leq N - 1$) et le partitionnement correspondant d'arbres de Π en K groupes qui minimise (ou maximise) la fonction objective choisie. La fonction de coût que nous minimisons à chaque pas de l'algorithme est la suivante (Équation 1) :

$$W(\Pi) = \sum_{k=1}^{K} \sum_{i=1}^{N_k - 1} RF(C_k, T_{ik}) \rightarrow Min, \quad (1)$$

où N_k est le nombre d'arbres dans la classe *k*, RF est la distance topologique de Robinson et Foulds [11, 12] entre deux arbres phylogénétiques, C_k est l'arbre consensus de la classe *k* (i.e., le centroïde de la classe *k*) et T_{ik} est l'arbre *i* de la classe *k*. Évidemment, dans le cas des *k*-moyennes, les classes des singletons ne doivent pas être considérées. Rappelons que l'arbre consensus majoritaire est l'arbre médian pour l'ensemble des arbres donnés dans le sens de la distance de Robinson et Foulds [13].

Pour déterminer le nombre optimal de groupes, nous utilisons le critère de Calinski-Harabasz. Ce critère, parfois appelé le critère de ratio de variance, est défini comme suit (Équation 2) :

$$CH = \frac{SSB}{SSW} \times \frac{(N-K)}{(K-1)}, \quad (2)$$

où SSB est l'indice d'évaluation intergroupe et SSW est l'indice d'évaluation intragroupe. Ce critère peut être utilisé pour les valeurs de K telles que : $2 \leq K \leq N - 1$. La variance globale entre les groupes (SSB) se calcule comme suit (Équation 3) :

$$SSB = \sum_{k=1}^{K} N_i \times RF(C_k, C), \tag{3}$$

où C est l'arbre consensus de tous les arbres appartenant à Π. La variance globale intragroupe (SSW) se calcule comme suit (Équation 4) :

$$SSW = \sum_{k=1}^{K} \sqrt{\sum_{i=1}^{N_k} RF^2(C_k, T_{ik})}. \tag{4}$$

Plus la valeur du critère de Calinski-Harabasz est grande, meilleur est le partitionnement des données. Le seul désavantage de *CH* est qu'il ne permet pas de comparer la solution consistant en un arbre consensus unique (cas où $K = 1$) avec la solution admettant les arbres consensus multiples (cas où $K \geq 2$). Pour pouvoir effectuer cette comparaison, nous avons également considéré le critère de Ball-Hall. Ce critère est défini comme suit (Équation 5) :

$$BH = \frac{1}{K} \sum_{k=1}^{K} \frac{1}{N_k} \sqrt{\sum_{i=1}^{N_k} RF^2(C_k, T_{ik})}. \tag{5}$$

Cependant, ce dernier critère ne tient pas compte de la distance intergroupe. Notons que la fonction objective principale *W*, utilisée par l'algorithme des *k*-moyennes (Équation 1), peut aussi être utilisée pour comparer la solution avec un arbre consensus unique à celles proposant des arbres consensus multiples.

3 Méthode et résultats

Dans cet article nous avons présenté une approche permettant de choisir un partitionnement optimal d'arbres phylogénétiques en vue de l'inférence d'un ou de plusieurs arbres consensus. Notons que la méthode proposée est générique car elle pourrait être appliquée dans le cadre du consensus strict, majoritaire ou majoritaire étendu. Cette méthode peut être utilisée dans plusieurs situations pratiques. Par exemple, pour regrouper les gènes ayant des histoires évolutives similaires, e.g., ceux qui ont subi les mêmes transferts horizontaux ou les mêmes recombinaisons génétiques. La méthode discutée pourrait aussi être utile pour distinguer les familles de gènes orthologues – un problème commun en bioinformatique. Les simulations statistiques doivent être effectuées pour tester les performances de la méthode proposée dans différents cas pratiques. Une autre question ouverte reste l'inférence de plusieurs arbres consensus à partir des arbres phylogénétiques partiels, i.e., définis sur des ensembles différents de feuilles – problématique des super-arbres.

Références

[1] Ivica Letunic and Peer Bork. Interactive tree of life (itol) : an online tool for phylogenetic tree display and annotation. Bioinformatics, 23(1) :127–128, 2007.

[2] David Bryant. A classification of consensus methods for phylogenetics. DIMACS series in discrete mathematics and theoretical computer science, 61 :163–184, 2003.

[3] David R Maddison. The discovery and importance of multiple islands of mostparsimonious trees. Systematic Biology, 40(3) :315–328, 1991.

[4] Alain Guénoche. Multiple consensus trees : a method to separate divergent genes. BMC bioinformatics, 14(1) :46, 2013.

[5] Tadeusz Cali´nski and Jerzy Harabasz. A dendrite method for cluster analysis. Communications in Statistics-theory and Methods, 3(1) :1–27, 1974.

[6] Hong He and Yonghong Tan. A two-stage genetic algorithm for automatic clustering. Neurocomputing, 81 :49–59, 2012.

[7] Geoffrey H Ball and David J Hall. Isodata, a novel method of data analysis and pattern classification. Technical report, DTIC Document, 1965.

[8] Maria Giulia Di Giuseppe, Antonio Troiano, Claudia Troise, and Giuseppe De Natale. k-means clustering as tool for multivariate geophysical data analysis. an application to shallow fault zone imaging. Journal of Applied Geophysics, 101 :108–115, 2014.

[9] James MacQueen. Some methods for classification and analysis of multivariate observations. In Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, volume 1, page 14. California, USA, 1967.

[10] Vladimir Makarenkov and Pierre Legendre. Optimal variable weighting for ultrametric and additive trees and k-means partitioning : Methods and software. Journal of Classification, 18(2) :245–271, 2001.

[11] DF Robinson and Leslie R Foulds. Comparison of phylogenetic trees. Mathematical Biosciences, 53(1) :131–147, 1981.

[12] Vladimir Makarenkov and Bruno Leclerc. Comparison of additive trees using circular orders. Journal of Computational Biology, 7(5) :731–744, 2000.

[13] Jean-Pierre Barthélemy and FR McMorris. The median procedure for n-trees. Journal of Classification, 3(2) :329–334, 1986.

Summary

Here we present a new algorithm allowing one to refine the consensus tree inferring procedure. The main advantage of our approach, compared to the traditional one which always provides as a solution a unique phylogenetic tree (i.e., additive tree, X-tree), is that is capable of inferring one or multiple consensus trees depending on the given set of phylogenetic trees. We carry out the k-means algorithm to obtain the optimal partitioning of the given set of trees.

Inférence de super-arbres phylogénétiques multiples en utilisant l'algorithme des *k*-moyennes

Nadia Tahiri*, Matthieu Willems*
Vladimir Makarenkov*

*Département d'informatique
Université du Québec à Montréal
Case postale 8888, Succursale Centre-ville
Montréal (Québec), H3C 3P8, Canada
makarenkov.vladimir@uqam.ca

Résumé. Nous présentons un nouvel algorithme permettant d'affiner l'inférence de super-arbres obtenus par regroupement d'arbres phylogénétiques (*i.e.*, arbres additifs ou *X*-arbres). Ces arbres sont définis sur des ensembles de feuilles différents mais partiellement chevauchants. Nous utilisons l'algorithme des *k*-moyennes pour obtenir le partitionnement optimal de l'ensemble des arbres considérés.

4 Introduction

Un des problèmes majeurs en biologie comparative est de combiner des arbres phylogénétiques définis sur différents ensembles de feuilles. L'arbre phylogénétique (*i.e.*, arbre additif) résultant est nommé « *super-arbre* ». Il peut être construit à l'aide de différents algorithmes (Sanderson *et al.*, 1998). La règle majoritaire a été introduite par Margush et McMorris (1981) pour construire un arbre consensus à partir d'arbres définis sur un même ensemble de feuilles. Cette dernière méthode retourne un arbre unique qui correspond aux bipartitions contenues dans 50% (ou plus selon le choix de l'utilisateur) des arbres d'entrée. Ainsi, les bipartitions majoritaires de l'arbre consensus peuvent être déterminées simplement en comptant la fréquence des occurrences des bipartitions dans les arbres d'entrée. Goloboff et Pol (2002) ont exprimé des doutes sur la possibilité de développer une méthode équivalente lorsque les arbres d'entrée sont définis sur des ensembles différents de feuilles. En effet, dans ce cas, il n'est pas toujours possible de déterminer combien d'arbres supportent (ou contredisent) une bipartition complète. Il n'est pas difficile de compter combien de fois une bipartition incomplète donnée apparait dans les arbres d'entrée, mais cette statistique ne peut pas être utilisée directement pour inférer les bipartitions complètes majoritaires. Nous devons donc utiliser un critère différent pour définir *un super-arbre majoritaire.*

Nous proposons ici un nouvel algorithme permettant de regrouper des arbres phylogénétiques en se basant sur deux principaux critères : notre nouvelle fonction objectif *W* et le critère de Caliński-Harabasz (*CH*) (Caliński et Harabasz, 1974), incorporés dans l'algorithme des *k*-moyennes.

5 Méthodologie

La méthode des *k*-moyennes est une méthode de partitionnement d'un ensemble de données en *k* groupes en fonction d'un critère de similarité déterminé (MacQueeen, 1967). Le problème de la recherche d'un partitionnement optimal est NP-difficile, ce qui explique l'existence de nombreuses variantes heuristiques de cette méthode. Dans notre étude, nous avons utilisé la version des *k*-moyennes implémentée par Makarenkov et Legendre (Makarenkov et Legendre, 2001).

Le nouvel algorithme que nous proposons dans ce papier est une généralisation de l'algorithme de Tahiri *et al.* (2014). Dans ce dernier article, l'algorithme prenait en entrée uniquement des arbres phylogénétiques définis sur un même ensemble de feuilles.

Le nouvel algorithme prend en entrée un ensemble d'arbres phylogénétiques définis sur des ensembles de feuilles différents mais partiellement chevauchants et retourne en sortie le partitionnement optimal de ces arbres en un ou plusieurs groupes. Pour chaque groupe retrouvé, l'algorithme retourne également la liste de ses éléments (*i.e.*, arbres phylogénétiques). Cette sortie s'accompagne de résultats statistiques (les indices *W* et *CH*). L'algorithme proposé utilise la méthode des *k*-moyennes adaptée à des données d'arbres.

Soit Π un ensemble de *N* arbres phylogénétiques définis sur des ensembles de feuilles (*i.e.*, espèces ou taxa) différents mais partiellement chevauchants. Nous recherchons le nombre optimal *K* ($1 \leq K \leq N-1$) et le partitionnement correspondant de Π en *K* groupes, qui minimisent (ou maximisent) la fonction objectif choisie. La fonction objectif que nous proposons de minimiser à chaque pas de l'algorithme est la suivante (Équation 1) :

$$W(\Pi) = \frac{1}{N-K} \sum_{k=1}^{K} \frac{2}{N_k \times (N_k - 1)} \sum_{i=1}^{N_k - 1} \sum_{j=i+1}^{N_k} w(T_{ki}, T_{kj}) \times \left(\frac{RF(T_{ki}, T_{kj})}{2n(T_{ki}, T_{kj}) - 6} \right)^2 \rightarrow Min\,, \quad (1)$$

où N_k est le nombre d'arbres dans la classe *k*, *RF* est la distance topologique de Robinson et Foulds (Robinson et Foulds, 1981 et Makarenkov et Leclerc, 2000) entre deux arbres phylogénétiques, T_{ki} est l'arbre phylogénétique *i* de la classe *k*, T_{kj} est l'arbre phylogénétique *j* de la classe *k*, $n(T_{ki}, T_{kj})$ est le nombre d'espèces identiques entre les arbres phylogénétiques T_{ki} et T_{kj} et $w(T_{ki},T_{kj})$ est un poids défini comme suit : $w(T_{ki},T_{kj}) = 0$ si $n(T_{ki},T_{kj}) < 4$, et $w(T_{ki},T_{kj}) = 1$, sinon. Rappelons qu'il n'existe qu'une seule topologie d'arbre avec 2 ou 3 feuilles. Nous normalisons la distance $RF(T_{ki},T_{kj})$ par $2n(T_{ki},T_{kj}) - 6$, qui est sa valeur maximum possible, pour obtenir une valeur située entre 0 et 1. Notre fonction *W* suppose qu'il n'y a pas de classe singleton ($N_k > 1$). Nous avons utilisé la distance *RF* à l'entrée de l'algorithme des *k*-moyennes car l'arbre consensus majoritaire d'un ensemble d'arbres est l'arbre médian de cet ensemble dans le sens de la distance de Robinson et Foulds (Barthélemy et McMorris, 1986).

Notre deuxième fonction objectif se base sur le critère de Caliński-Harabasz. Ce critère, parfois appelé critère de ratio de variance, est défini comme suit (Équation 2) :

$$CH = \frac{SS_B}{SS_W} \times \frac{N-K}{K-1}, \quad (2)$$

où SS_B est l'indice d'évaluation intergroupe et SS_W est l'indice d'évaluation intragroupe. Ce critère peut être utilisé pour les valeurs de *K* telles que : $2 \leq K \leq N-1$.
La variance globale intragroupe (SS_W) se calcule comme suit (Équation 3) :

$$SS_W = \sum_{k=1}^{K} \frac{1}{N_k} \sum_{i=1}^{N_k - 1} \sum_{j=i+1}^{N_k} w(T_{ki}, T_{kj}) \times \left(\frac{RF(T_{ki}, T_{kj})}{2n(T_{ki}, T_{kj}) - 6} \right)^2. \quad (3)$$

La variance globale entre les groupes (SS_B) se calcule comme suit (Équation 4) :

$$SS_B = \frac{1}{N} \left(\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} w(T_i, T_j) \times \left(\frac{RF(T_i, T_j)}{2n(T_i, T_j) - 6} \right)^2 \right) - SS_W\,, \quad (4)$$

où $n(T_i,T_j)$ est le nombre d'espèces identiques entre les arbres phylogénétiques T_i et T_j de l'ensemble d'arbres donnés Π. Les poids $w(T_i,T_j)$ sont définis comme précédemment.
La valeur maximum du critère de Caliński-Harabasz indique le nombre optimal de classes. Le seul désavantage de *CH* est qu'il ne permet pas de comparer la solution consistant en un arbre consensus unique (cas où $K = 1$) avec la solution admettant des arbres consensus multiples (cas où $K \geq 2$).

6 Résultats des simulations

Nous avons effectué des simulations avec notre nouvel algorithme afin d'évaluer la performance des deux critères employés (*i.e.*, *W* et *CH*). Nous avons également testé une deuxième variante pour chacun des critères lors de nos simulations, sans mettre la distance *RF* au carré dans les Équations 1, 3 et 4 ci-dessus. Nous avons généré aléatoirement K arbres phylogénétiques binaires T_1, ..., T_K avec n feuilles chacun (*i.e.*, arbres centroïdes des classes), pour K compris entre 1 et 5, et n compris dans l'ensemble {8, 16, 32, 64}. Pour chaque arbre phylogénétique T_i (où $i = 1, ..., K$), nous avons généré un ensemble de 100 arbres correspondant à la classe i pour chacun des intervalles de bruit indiqués ci-dessous. Chaque élément T de la classe i était un arbre phylogénétique tel que le pourcentage de similitude (mesuré à l'aide de la distance *RF*) entre T et T_i était : de 0 à 10%, de 10 à 25%, de 25 à 50% ou de 50 à 75%. Ces intervalles correspondent au niveau de bruit qu'on note B (B = 10%, 25%, 50%, 75%). Nous avons par la suite enlevé aléatoirement un nombre aléatoire de feuilles (variant de 0 à $n/2$) des arbres générés pour créer des arbres partiels. Nous avons exécuté notre programme sur les ensembles d'arbres, en générant 100 jeux de données pour chaque jeu de paramètres (K, n, B). Les tendances obtenues sont les suivantes. Tout d'abord, les résultats s'améliorent quand le nombre de feuilles de l'arbre augmente. De plus, avec des niveaux de bruit élevés, la valeur de l'indice Rand diminue de 1 jusqu'à des valeurs proches de 0,7. En effet, plus il y a de bruit dans les clusters, plus il est difficile à l'algorithme de placer correctement les arbres phylogénétiques dans les bons groupes. Nous avons également constaté que dans la plupart des cas, le critère *CH* a donné des résultats supérieurs au critère *W* et que la variante des Équations 1, 3 et 4 présentée ci-dessus a fourni des meilleurs résultats que celle considérant la distance *RF* sans le carré.

7 Conclusion

Dans cet article nous avons présenté une approche permettant de choisir un partitionnement optimal d'un ensemble d'arbres phylogénétiques en vue de l'inférence d'un ou de plusieurs super-arbres. Mentionnons que la méthode discutée pourrait se généraliser au cas du consensus majoritaire étendu. Notre algorithme peut être utilisé dans plusieurs situations pratiques (*e.g.*, le regroupement de gènes ayant des histoires évolutives similaires ou la vérification de la stabilité d'un arbre inféré en utilisant le bootstrap). Le principal avantage de la méthode présentée est sa capacité à traiter les arbres phylogénétiques partiels (*i.e.*, définis sur des ensembles de feuilles différents mais partiellement chevauchants).

Références

Barthélemy, J. P. et McMorris, F. R. (1986). The median procedure for n-trees. Journal of Classification, 3, 329-334.

Bryant, D. (2003). A Classification of Consensus Methods for Phylogenetics. DIMACS series in discrete mathematics and theoretical computer science, 61, 163-184.

Caliński, T. et Harabasz, J. (1974). A dendrite method for cluster analysis. Communications in Statistics-theory and Methods, 3, 1-27.

Goloboff, P. A. et Pol, D. (2002). Semi - strict supertrees. Cladistics, 18, 514-525.

Guénoche, A. (2013). Multiple consensus trees: a method to separate divergent genes. BMC Bioinformatics, 14, 46.

MacQueen, J. B. (1967). Some Methods for classification and Analysis of Multivariate Observations. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability 1. University of California Press, pp. 281–297.

Makarenkov, V. et Legendre, P. (2001). Optimal variable weighting for ultrametric and additive trees and K-means partitioning: Methods and software. Journal of Classification, 18, 245-271.

Makarenkov, V. et Leclerc, B. (2000). Comparison of additive trees using circular orders, Journal of Computational Biology, 7, 731-744.

Margush, T. et McMorris, F. R. (1981). Consensusn-trees. Bulletin of Mathematical Biology, 43, 239-244.

Rand, W. M. (1971) Objective criteria for the evaluation of clustering methods. Journal of the American Statistical Association, 66 (336), pp. 846-850.

Robinson, D. F. et Foulds, L. R. (1981). Comparison of phylogenetic trees. Mathematical Biosciences, 53, 131-147.

Sanderson, M. J., Purvis, A. et Henze, C. (1998). Phylogenetic supertrees: assembling the trees of life. Trends in Ecology & Evolution, 13, 105-109.

Tahiri, N., Willems, M. et Makarenkov, V. (2014). Classification d'arbres phylogénétiques basée sur l'algorithme des k-moyennes, Actes de SFC 2014, Rabat, Maroc, pp. 49-54.

A new fast method for building multiple consensus trees using *k*-means

Nadia Tahiri[1] and Vladimir Makarenkov[1]

[1]Département d'informatique, Université du Québec à Montréal, Case postale 8888, Succursale Centre-ville, Montréal, (Québec), H3C 3P8, Canada
tahiri.nadia@courrier.uqam.ca
makarenkov.vladimir@uqam.ca

Abstract. We describe a new fast method for inferring multiple consensus trees from a given set of phylogenetic trees (i.e. additive trees or *X*-trees) defined on the same set of species (i.e. object or taxa). The traditional consensus approach yields a single consensus tree. We use the popular *k*-means partitioning algorithm to divide a given set of trees into several clusters. We propose a new version of the well-known Caliński-Harabasz cluster validity index, adapted for tree clustering. The main advantage of our method is that it is much faster than the existing tree clustering approaches, while providing similar or better clustering results in most cases. This makes our method particularly appropriate for the analysis of large genomic and phylogenetic datasets.

Keywords. Bioinformatics, consensus tree, phylogenetic tree, *k*-means clustering, cluster validity index

1 INTRODUCTION

The implementation of the famous project "Tree of Life" (ToL) intended for reconstruction of the largest possible phylogenetic tree was possible due to a collaborative effort of biologists and nature enthusiasts from around the world [1]. The approach adopted by the project organizers allows for a gradual reduction of the complex tree reconstruction problem into several sub-problems and then for merging the obtained results. Indeed, such an approach produces thousands of small trees which should be combined in order to infer Tree of Life. The problem is twofold: first, we infer small sub-trees of ToL (i.e., often gene trees) defined on different sets of taxa (i.e. species or objects), second, we merge these small trees into one or multiple large consensus phylogenies using a supertree reconstruction algorithm, which allows for combining trees inferred for different, but mutually overlapping, sets of taxa [2]. Here we present a new method which can be applied to solve the problem of inferring multiple consensus trees.

There are many methods for defining a consensus tree for a given set of phylogenetic trees [3]. The most known types of consensus tree are the strict consensus tree, the majority consensus tree and the extended majority consensus tree [3,4]. The strict consensus tree contains only the edges that are common to all input trees. The majority consensus tree contains the edges that are present in more than 50% of the input trees, although higher percentages may also be considered. According to the extended majority rule, the consensus tree includes all the majority edges to which compatible residual edges are added gradually, starting with the most frequent ones. Extended majority consensus trees are the most frequently used consensus trees in bioinformatics because they are usually much better resolved (i.e. has lower total degree of internal nodes) than strict and majority consensus trees [4].

The output of most conventional consensus algorithms is a single consensus tree [3]. However, in many practical situations it is much more appropriate to infer several consensus trees. In biology, it is often risky to group phylogenetic trees constructed for different genes. Each gene has its own evolutionary history, which can be very different from the evolutionary history of other genes. For example, some genes or groups of genes can be affected by specific horizontal transfer events and thus display different evolutionary patterns from the rest of genes under study [5,6]. Their evolutionary history will be represented by phylogenetic trees having different tree topologies from that of the genes not affected by any gene transfer.

In this paper we describe a new algorithm for determining clusters of homogeneous trees which can be combined within their clusters in order to infer multiple consensus trees. The idea of building several consensus trees was originally formulated by Maddison [7], who found that consensus trees of some subsets of a given set of trees are often very different and have better resolution than the consensus tree of the whole set. Then, Stockham *et al.* [8] proposed two variants of the popular *k*-means algorithm [9] to infer a set of strict consensus trees (called characteristic trees) that

minimize the information loss. However, these methods are very expensive in terms of the running time because the consensus trees must be determined for each set of clusters in all intermediate partitioning solutions tested by *k*-means. Our algorithm makes use of a specific version of the Caliński-Harabasz (CH) index [10] adapted for tree clustering. This cluster validity index will be used to determine the best partitioning obtained over multiple random starts of *k*-means when the number of clusters is fixed and to select the number of clusters in a given set of trees.

2 METHODS

2.1 *K*-means algorithm adapted for tree clustering

A phylogenetic tree is an unrooted leaf-labeled tree in which each internal node, representing an ancestor of contemporary species, has at least two children and all leaves, representing contemporary species, have different labels. Our algorithm takes as input a set of phylogenetic trees Π defined on the same set of leaves and returns as output one or several consensus trees. Each consensus tree represents a subset (i.e. group, class or cluster) of trees in Π. For each cluster identified, the algorithm returns a list of its elements (i.e. phylogenetic trees) and the corresponding consensus tree. The output is also accompanied by some statistics (e.g. the value of the *CH* cluster validity index). Our method uses a version of the popular *k*-means algorithm suitable for tree clustering. The *k*-means algorithm [9] is an unsupervised classification algorithm which iteratively regroups N objects (i.e. here, phylogenetic trees) into K clusters. The content of each cluster is chosen to minimize the intracluster distances. Generally, the most commonly used distances in the framework of *k*-means are the Euclidean distance, the Manhattan distance and the Minkowski distance. On the other hand, the Robinson and Foulds distance (*RF*) [11,12] between two trees is a well-known distance used in bioinformatics to compare topologies of two phylogenetic trees defined on the same set of taxa. The *RF* distance is a topological distance. It does not take into account the lengths of the tree edges. The problem of finding an

optimal partitioning according to the *k*-means criterion is known to be NP-hard [13]. This justifies the development of a number of polynomial-time heuristics, most of them having the time complexity of $O(K \times N \times i \times M)$, for finding an approximate partitioning solution, where i is the number of iterations in the *k*-means algorithm and M is the number of variables characterizing each of the N objects. Our main novelty is that in the proposed version of *k*-means we do not recompute the consensus trees for intermediate clusters of trees, but estimate the quality of each intermediate clustering using an approximate formula based on the computation of *CH*. This allows for a much faster partitioning of a given set phylogenetic trees into K clusters without loosing the quality of the obtained consensus trees.

The objective function in case of tree clustering can be defined as follows:

$$OF = \sum_{k=1}^{K} \sum_{i=1}^{N_k} RF(C_k, T_{ki}), \tag{1}$$

where K is the number of clusters, N_k is the number of trees in cluster k, *RF* is the Robinson and Foulds topological distance between two phylogenetic trees with n leaves, T_{ki} is the tree i of cluster k and C_k is the consensus tree of cluster k. Still, the computation of the majority rule consensus tree or extended majority rule consensus tree is time-consuming. The time complexity of the method computing the majority or extended majority rule consensus tree is $O(n^2 + nN^2)$ [4]. Even though an optimal algorithm for computing the majority rule consensus tree, running in $O(nN)$, has been recently proposed [14], this time complexity cannot be achieved when the trees are defined by their Newick strings as it is typically the case in bioinformatics [4]. Thus, the time complexity of a straightforward tree partitioning algorithm, such as the algorithm of Stockham *et al.* [8], which recomputes the consensus trees after each *basic k-means operation*, consisting of relocating an object (i.e. tree) from one cluster

to another and then in reassessing the value of the objective function (Formula 1), is $O(K \times n \times (n + N^2) \times i \times M)$.

Theorem 1

For a given cluster k containing N_k phylogenetic trees (i.e. additive trees or X-trees) the following inequalities hold:

$$\frac{1}{N_k - 1} \sum_{i=1}^{N_k - 1} \sum_{j=i+1}^{N_k} RF(T_i, T_j) \leq \sum_{i=1}^{N_k} RF(C_k, T_i) \leq \frac{2}{N_k} \sum_{i=1}^{N_k - 1} \sum_{j=i+1}^{N_k} RF(T_i, T_j), \qquad (2)$$

where N_k is the number of trees in cluster k, T_i and T_j are, respectively, the trees i and j in cluster k, and C_k is the majority rule consensus tree for cluster k.

The proof of Theorem 1 is omitted here for the sake of conciseness. We can now use the middle of the interval defined by the inferior and superior bounds of $\sum_{i=1}^{N_k} RF(C_k, T_i)$ established by Theorem 1. Thus, the value $\frac{3N_k - 2}{2N_k(N_k - 1)} \sum_{i=1}^{N_k - 1} \sum_{j=i+1}^{N_k} RF(T_i, T_j)$, can be used as an approximation of the contribution of cluster *k* to the *k*-means objective function adapted for tree clustering (Formula 1).

Hence, we can speed up the tree clustering process by using the following objective function (Formula 3):

$$OF_{appr} = \sum_{k=1}^{K} \frac{3N_k - 2}{2N_k(N_k - 1)} \sum_{i=1}^{N_k - 1} \sum_{j=i+1}^{N_k} RF(T_i, T_j), \qquad (3)$$

which can be viewed as an approximate version of the objective function defined by Formula 1. The main advantage of using OF_{appr} is that we should not recompute the consensus trees for the two clusters involved in each basic *k*-means operation. Indeed,

the time complexity of such a basic operation will be only $O(N)$ because all pairwise *RF* distances between trees in Π can be precalculated in $O(n \times N^2)$ [11]. This will lead to the global time complexity of $O(n \times N^2 + K \times N \times i \times M)$ for our majority rule *k*-means. This is usually much faster than the $O(K \times n \times (n + N^2) \times i \times M)$ time complexity of a standard approach based on Formula 1.

2.2 Caliński-Harabasz cluster validity index adapted for tree clustering

The first cluster validity index we consider here is the Caliński-Harabasz index [10]. This criterion, sometimes called the variance ratio criterion, is defined as follows:

$$CH = \frac{SS_B}{SS_W} \times \frac{N-K}{K-1}, \tag{4}$$

where SSB is the index of intergroup evaluation, SSW is the index of intragroup evaluation, K (K > 1) is the number of clusters and N is the number of objects (trees in our case). The optimal number of clusters corresponds to the highest value of CH.

We propose to use an approximate version of the coefficients SSB and SSW that does not require the computation of majority (or extended majority) rule consensus trees at each iteration of k-means. The approximate formula for SSW is as follows:

$$SS_W = \frac{1}{2} \sum_{k=1}^{K} \frac{3N_k - 2}{N_k(N_k - 1)} \sum_{i=1}^{N_k - 1} \sum_{j=i+1}^{N_k} RF(T_{ki}, T_{kj}). \tag{5}$$

Thus, the new global variance between groups, SS_B, can be calculated as follows:

$$SS_B = \frac{3N-2}{2N(N-1)} \left(\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} RF(T_i, T_j) \right) - SS_W. \tag{6}$$

3 RESULTS

We examined the evolution of 47 ribosomal proteins of the 14 organisms of Archaea originally studied by Matte-Tailliez *et al.* (see Figure 1a in [6]). Matte-Tailliez and colleagues inferred a single species phylogenetic tree after the concatenation of the available protein sequences. However, the evolution of each of these proteins can be represented by its own phylogenetic tree. The cluster analysis of these trees can tell us how many different evolutionary scenarios characterize the evolution of these sequences (i.e. how many clusters exist in the protein tree dataset).

The species phylogenetic tree S and its branch lengths were obtained by these authors using the PUZZLE program [15] with Γ-law correction. Matte-Tailliez *et al.* [6] discussed the problems encountered when reconstructing some parts of the species archaeal phylogeny and pointed out the evidence of horizontal gene transfers influencing the evolution of some of its genes. Thus, we will identify the consensus gene transfer scenarios characterizing the evolution of these proteins. Using the PUZZLE program, we first reconstructed phylogenetic trees for each of the 47 available ribosomal proteins. We used the version of our algorithm based on the *CH* index and the non-squared *RF* function (Formula 6) to infer an optimal partitioning of the given set of 47 trees. The maximum of *CH* was attained with two clusters ($K = 2$).

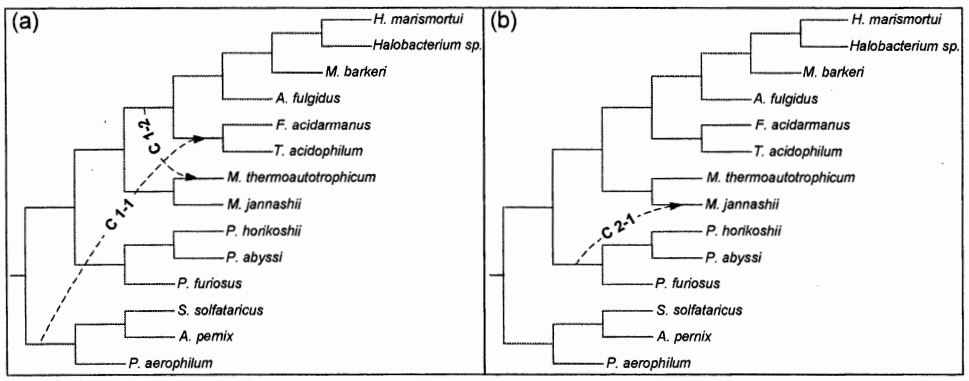


Fig. 1. Two possible scenarios of horizontal gene transfer events obtained for 47 protein trees studied by Matte-Tailliez *et al.* [6].

The first cluster contained 20 trees and the second 27 trees. We then inferred the extended majority consensus trees, $C1$ and $C2$, for the two obtained clusters of trees. Afterwards, we used the T-Rex algorithm [5] to identify the scenarios of horizontal gene transfer events that reconcile the species tree S and the consensus trees $C1$ and $C2$. The first scenario, including 2 gene transfer events (Arrows C1-1 to C1-2), is presented in Figure 3a and the second scenario, including only one gene transfer event (Arrow C2-1), is presented in Figure 3b. These transfers account for two different histories that characterize the evolution of clusters of 20 and 27 genes of the 14 considered organisms of Archaea. It is worth noting that the transfer C1-1 was also predicted by Boc *et al.* (see Figure 6 in [5]) and is in complete agreement with the results of Matte-Tailliez *et al.* [6].

4 CONCLUSION

In this article we described a new algorithm for partitioning a set of phylogenetic trees in multiple clusters in order to infer multiple consensus trees. We presented a number of new formulas allowing for using the popular Caliński-Harabasz cluster validity index and the Robinson and Foulds topological distance in the framework of tree clustering. The new algorithm can be used to address a number of important issues in bioinformatics, such as the identification of genes having similar evolutionary histories (e.g. those that underwent the same horizontal gene transfer events) or inference of multiple sub-trees of Tree of Life. In the future, it would be interesting to modify the objective function (2) in to make it work with partial phylogenetic trees (i.e. multiple supertree reconstruction problem), when the input trees are defined on different, but overlapping, sets of leaves.

5 REFERENCES

1. Letunic I, Bork P (2007) Bioinformatics 23(1):127-128.
2. Bininda-Emonds OR (ed) (2004) Phylogenetic supertrees: combining information to reveal the tree of life. Springer Science & Business Media, 4.
3. Bryant D (2003) DIMACS Ser Discrete Math Theoret Comput Sci 61:163-184.
4. Felsenstein J (2004) Inferring phylogenies. Sunderland: Sinauer Associates, 2.
5. Boc A, Philippe H, Makarenkov V (2010) Syst Biol 59:195-211.
6. Matte-Tailliez O, Brochier C, Forterre P, Philippe H (2002) Mol Biol Evol 19(5):631-639
7. Maddison DR (1991) Syst Biol 40(3):315-328.
8. Stockham C, Wang LS, Warnow T (2002) Bioinformatics 18(suppl 1):S285-S293
9. MacQueen J (1967) Some methods for classification and analysis of multivariate observations. In *Proceedings of 5th Berkeley symposium on statistics and probability*, 1, 14, 281-297.
10. Caliński T, Harabasz JA (1974) Commun Stat Theor M 3(1):1-27.
11. Makarenkov V, Leclerc B (2000) J Comput Biol 7(5):731-744.
12. Robinson DF, and Foulds LR (1981) Math Biosci 53(1), 131-147.
13. Mahajan M, Nimbhorkar P, Varadarajan K (2009) The planar k-means problem is NP-hard. In *Proceedings of Workshop on Algorithms and Computation*, 274-285.
14. Jansson J, Shen C, Sung WK (2013) An optimal algorithm for building the majority rule consensus tree. In *Proceedings of Annual International Conference on Research in Computational Molecular Biology*, Springer Berlin Heidelberg, 88-99.
15. Strimmer K, Von Haeseler A (1996) Mol Biol Evol 13(7):964-969.

A median-based consensus rule for distance exponent selection in the framework of intelligent and weighted Minkowski clustering

Renato Cordeiro de Amorim, Nadia Tahiri, Boris Mirkin and Vladimir Makarenkov

Abstract

The intelligent Minkowski and weighted Minkowski K-means are recently developed effective clustering algorithms capable of computing feature weights. Their cluster-specific weights follow the intuitive idea that a feature with a low dispersion in a specific cluster should have a greater weight in this cluster than a feature with a high dispersion. The final clustering provided by these techniques obviously depends on the selection of the Minkowski exponent. The median-based central consensus rule we introduce in this paper allows one to select an optimal value of the Minkowski exponent. Our rule takes into account the values of the Adjusted Rand Index (ARI) between clustering solutions obtained for different Minkowski exponents and selects the clustering that provides the highest average value of ARI. Our simulations, carried out with real and synthetic data, show that the proposed median-based consensus procedure usually outperforms clustering strategies based on the selection of the highest value of the Silhouette or Calinski-Harabasz cluster validity indices.

---

[6]Renato Cordeiro de Amorim
School of Computer Science, University of Hertfordshire, College Lane AL10 9AB, UK, r.amorim@herts.ac.uk

Nadia Tahiri
Département d'informatique, Université du Québec à Montréal, C.P. 8888 succ. Centre-Ville, Montreal (QC) H3C 3P8 Canada, tahiri.nadia@courrier.uqam.ca

Boris Mirkin
Department of Data Analysis and Machine Intelligence, National Research University Higher School of Economics, Moscow, Russian Federation, and Department of Computer Science and Information Systems, Birkbeck University of London, Malet Street, London WC1E 7HX, UK, bmirkin@hse.ru

Vladimir Makarenkov
Département d'informatique, Université du QuÃ©bec à Montréal, C.P. 8888 succ. Centre-Ville, Montreal (QC) H3C 3P8 Canada, makarenkov.vladimir@uqam.ca

1 Introduction

Clustering algorithms aim at revealing the structure of a given data set Y by partitioning it into a set of K clusters $S = \{S_1, S_2, ..., S_K\}$. Each cluster $S_k \in S$ should contain homogeneous entities, according to a given similarity measure. Clustering algorithms have been used to address the most diverse problems, including those in the fields of bioinformatics, business, computer vision, data mining, and security [9, 17, 24, 14].

Over the years many clustering algorithms have been developed thanks to the considerable effort put by the scientific community. These algorithms can be easily categorised by the approach they use to cluster entities, being either hierarchical or partitional. Algorithms following the former principle produce a tree-like relationship between the entities, which can be visualised using a dendrogram [18, 19]. Partitioning algorithms produce clustering solutions in which each entity is assigned either to a single cluster, i.e., in the case of crisp clustering, or to multiple clusters, i.e., in the case of overlapping clustering [17].

The K-means algorithm [13, 2] is one of the most popular clustering method. Its implementations can be found in various software frequently used in data analysis, such as MATLAB [15], R [22], SPSS [6], and SciPy [11]. Given a data set Y composed of N entities y_i, each described over the same set of V features, K-means produces a set of disjoint clusters $S = \{S_1, S_2, ..., S_K\}$. K-means also provides a set of centroids $C = \{c_1, c_2, ..., c_K\}$ of these clusters by iteratively minimising the following objective function:

$$W(S,C) = \sum_{k=1}^{K} \sum_{i \in S_k} \sum_{v=1}^{V} (y_{iv} - c_{kv})^2. \quad (1)$$

Despite its popularity, the K-means algorithm has several important weaknesses that are as follows: (i) it requires the number of clusters, K, to be known beforehand; (ii) the final clustering, S, is heavily dependant of the starting random partition; (iii) the provided solution is usually a local minimum; (iv) K-means treats all features equally, regardless of their degree of relevance.

Here, we are particularly interested in issue (iv). We have addressed this issue in our recent work, where we have introduced the Minkowski weighted K-means (MWK-means) algorithm [5] (more details are given in Section 2). However, our algorithm requires the selection of a suitable Minkowski exponent p. This selection is subjective to the data structure of Y, making it impossible to suggest a value of p that will produce good results in terms of cluster recovery in all cases. In this paper, we present a median-based central consensus rule allowing one to select a suitable value of the Minkowski exponent p in the framework of the intelligent Minkowski and weighted Minkowski clustering [5]. Our new technique will be tested empirically on real-world and synthetic datasets with different numbers of clusters and features.

2 Minkowski weighted K-means

We have recently introduced the MWK-means algorithm [5], which allows one to calculate optimal cluster-based feature weights. This calculation follows the intuitive idea that the weight of feature v in cluster k, denoted here as w_{kv}, can vary from one cluster to another. In fact, the more a feature is dispersed in a cluster, the lower will be its weight in this cluster. The Minkowski distance between entity y_i and centroid c_k is given by $\sqrt[p]{\sum_{v=1}^{V} | y_{iv} - c_{kv} |^p}$.

We added to this distance the feature weights and used its p^{th} power. The latter is analogous to the use of the squared Euclidean distance instead of the Euclidean distance:

$$d_p(y_i, c_k) = \sum_{v=1}^{V} w_{kv}^p | y_{iv} - c_{kv} |^p . \quad (2)$$

This weighted distance measure leads to the MWK-means objective function to be minimized:

$$W(S, C, w) = \sum_{k=1}^{K} \sum_{i \in S_k} \sum_{v=1}^{V} w_{kv}^p | y_{iv} - c_{kv} |^p, \quad (3)$$

subject to $\sum_{v=1}^{V} w_{kv} = 1$ and $w_{kv} \geq 0$ for $k = 1,2,...,K$ and $v = 1,2,...,V$. The computation is carried out in the framework of crisp clustering in which every entity y_i is assigned to a single cluster S_k.

The criterion (3) has some interesting implications. As both the distances and the weights use the same exponent p, one can see the feature weights as feature re-scaling factors. This interpretation is not possible in other feature weighting algorithms, such as Weighted K-means [7], Attribute weighting K-means [4], and Improved K-prototypes [10]. The criterion (3) supports cluster-dependant feature weights, and perhaps most importantly, a p-dependant cluster shape bias.

Any distance measure introduces a bias to the shapes of the clusters. Assuming a two-dimensional space for easier visualisation, the squared Euclidean distance used in (1) makes K-means biased towards circular clusters. At values of p equal to one, two and tending to infinity, the distance (2) is equivalent to the weighted versions of the Manhattan, squared Euclidean, and Tchebychev distances, respectively. For instance, a value of p between one and two leads to a bias towards a shape situated between a rhombus and a circle.

Within MWK-means, each weight w_{kv} is inversely proportional to the dispersion of v in the cluster S_k, which is given by $D_{kv} = \sum_{i \in S_k} | y_{iv} - c_{kv} |^p$. Thus, the feature weight is computed as follows:

$$w_{kv} = \frac{1}{\sum_{u \in V} [D_{kv}/D_{ku}]^{1/(p-1)}}. \quad (4)$$

This computation of weights follows the principle that feature v having the lowest dispersion in cluster S_k should have the highest weight in this cluster. The Minkowski weighted K-means iteratively minimises the criterion (3) following the steps below:

1. Initial settings. Choose the number of clusters, K, and the Minkowski exponent, p. Set $S \leftarrow \varnothing$, and $w_{kv} = 1/V$ for $k = 1,2,...,K$ and $v = 1,2,...,V$.

2. Centroids setting. Assign the values of K entities selected at random from Y to be the initial cluster centroids $c_1, c_2, ..., c_K$.

3. Clusters update. Assign each entity $y_i \in Y$ to the cluster S_k represented by the nearest c_k as per (2), generating the clustering $S' = \{S'_1, S'_2, ..., S'_K\}$. If $S' = S$, then go to Step 6 (i.e., computation finishes).

4. Centroids update. Update each $c_k \in C$ to the component-wise Minkowski center of $y_i \in S_k$.

5. Weights update. Update each feature weight w_{kv} using Equation (4). Set $S \leftarrow S'$, then go to Step 3.

6. Final step. Output the clustering $S = \{S_1, S_2, ..., S_K\}$, centroids $C = \{c_1, c_2, ..., c_K\}$, and feature weights w.

As the K-means algorithm, the MWK-means algorithm is a heuristic method that cannot guarantee to provide an optimal clustering solution. When using K-means, this issue is often addressed by running the algorithm thousands of times and selecting the clustering S that provides the optimal value of the objective function (1). This strategy can be followed within MWK-means for a given value of p. However, it cannot be used for finding an optimal value of p within MWK-means because the values of the objective function (3) are not comparable at different values of p.

To address the problem of the solution dependence from multiple random starts of MWK-means, we proposed the intelligent Minkowski weighted K-means (iMWK-means) algorithm [5], applying the concept of anomalous patterns to provide good initial centroids and weights to MWK-means. It is worth noting that the concept of anomalous patterns was originally introduced by Mirkin in the framework of the intelligent K-means algorithm [17]. Below, we present the main steps of this algorithm, including our modifications to allow the use of the Minkowski distance:

1. Initial settings. Choose a value for the threshold θ and the Minkowski exponent p. Set c_c to the component-wise Minkowski center of Y. Set $C_t \leftarrow \varnothing$, and

$w_{kv} = 1/V$ for $k = 1,2,...,K$ and $v = 1,2,...,V$.

2. Define centroid. Set a tentative centroid, c_t, as the entity farthest away from c_c according to the distance measure defined by (2).

3. Clustering procedure. Carry out the MWK-means algorithm using two centroids, c_t and c_c, in order to generate the clustering $S = \{S_t, S_c\}$ (without allowing c_c to change during the centroids update step).

4. Remove small clusters. If $|S_t| \geq \theta$, then add c_t to C_t and the corresponding weights to w. In all the cases, remove S_t from the dataset.

5. Check entities to be clustered. If some entities to be clustered remain, then go to Step 2.

6. Final step. Carry out MWK-means, initialised with the centroids from C_t and the weights from w.

The intelligent K-means algorithm, which serves as our foundation for iMWK-means, can be used for both estimating the correct number of clusters, K, in a dataset and providing good initial centroids. In this study, as in all our previous experiments [5], we were not particularly interested in estimating K. In our simulations, we assumed that the correct number of clusters was known.

3 The median-based central consensus rule

The final clustering generated by MWK-means and iMWK-means depends on the value of the Minkowski exponent p, which is also the exponent of feature weights w_{kv}, and defines the shape towards which the clustering will be biased. With this in mind, we devised a median-based central consensus rule that allow us to select a good value of this important clustering parameter.

Selecting the best clustering from a set of clustering solutions is indeed a problem that has attracted considerable attention [17]. A common approach is to apply a cluster validity index (CVI) to each of the available solutions, hoping this index is capable of sensing the structure of the data, and then output the clustering that better optimises the selected CVI. There are indeed many cluster validity indices (for a recent and comprehensive review see [1]), but no sole CVI is clearly superior to all the others in all cases. However, the Silhouette width (SW) [23] and the Calinski-Harabasz (CH) [3] index tend to be among the top performers [1, 16, 21].

Our median-based central consensus rule can be used with the algorithms that should be carried out only once, like iMWK-means, as well as with those that are based on multiple random initializations, like MWK-means. In the latter case we also employ a CVI, here using either SW or CH. In both cases, we first need to select a range of values of p for our search. We decided to use the values of p from the range 1.0 to 5.0 with the step of 0.1. We believe that there is no need in using large values of p, as in our previous comprehensive

simulations [5] the best partitions have very rarely appeared at the values of p greater than 5.

The median-based central consensus rule for MWK-means is defined as follows:

1. For each value of p, carry out MWK-means 100 times saving only the run with the optimal CVI value (i.e., the highest value of SW or CH in our case). This generates 41 clustering solutions, one for each value of the Minkowski exponent p (i.e., the values of p vary from 1.0 to 5.0 with the step of 0.1).
2. Calculate the Adjusted Rand Index (ARI) [8] between each pair of optimal clustering solutions found in Step 1.
3. Output the clustering solution with the highest average ARI value over all optimal clustering solutions.

In the case of the iMWK-means algorithm, which does not rely on a random start strategy, there is no need in applying a CVI. Here, a given value of p will provide a single clustering solution.

The median-based central consensus rule for iMWK-means is defined as follows:

1. For each value of p, carry out iMWK-means. This generates 41 clustering solutions, one for each value of the Minkowski exponent p (i.e., the values of p vary from from 1.0 to 5.0 with the step of 0.1).
2. Calculate the ARI between each pair of clustering solutions found in Step 1 (here each p generates one clustering).
3. Output the clustering solution with the highest average ARI value over all available clustering solutions.

As in [5], in all our experiments we first considered clustering solutions that generated the expected number of clusters, discarding all the others. When no such correct solutions were provided, we accepted the solutions generated by our algorithms regardless of the number of clusters.

4 Setting of the experiments

In this study, we carried out simulations with real-world benchmark datasets as well as with synthetic data, with and without noise features in both cases. We have examined 12 real-world datasets; 11 of them come from the popular UCI machine learning repository [12] and one from the study of Pal and Majumder [20].

Our synthetic data were composed of spherical Gaussian clusters so that the covariance matrices were diagonal, with the same diagonal value σ^2 generated at each cluster and varying between 0.5 and 1.5 (see also [5]). All centroid components were generated independently using the standard normal distribution. Cluster cardinalities were generated using a uniform distribution, with a constraint that each generated cluster should have included at least 20 entities. In this study, we tested six GMMs configurations that were different in terms of the number of features and clusters. They were as follows: 1000 entities over four features, partitioned into two clusters (1000x4-2); 1000 entities over six features,

partitioned into three clusters (1000x6-3); 1000 entities over eight features, partitioned into four clusters (1000x8-4); 1000 entities over 10 features, partitioned into five clusters (1000x10-5); 1000 entities over 20 features, partitioned into six clusters (1000x20-6); 1000 entities over 40 features, partitioned into eight clusters (1000x40-8). We generated 100 different datasets for each of these configurations.

Moreover, we also analyzed datasets containing noise features. Here, a noise feature is one composed entirely of uniformly random values. In these experiments, 50% of the number of original features were added to the original datasets. Thus, if an original dataset had eight features, its "noisy" version had a total of 12 features (eight original and four composed of noise).

We standardised each dataset by its range using the following formula:

$$y_{iv} = \frac{y_{iv} - \bar{y}_v}{max(y_v) - min(y_v)}. \quad (5)$$

Often clustering experiments are run with datasets standardised using the popular z-score method. We believe that the above-presented equation is a good alternative normalization option. Imagine, for instance, a dataset with two features, a unimodal feature v_1 and a multimodal feature v_2. The standard deviation of v_2 will be higher than that of v_1, leading to lower z-score values of v_2 in comparison with v_1. This means that the feature v_1 will have a higher contribution to the clustering despite the fact that the feature v_2 has a clearer cluster structure.

5 Results

In this section, we compare the results provided by the introduced median-based consensus MWK-means and iMWK-means algorithms to those given by the traditional K-means, MWK-means and iMWK-means approaches. All the tables in this section present the ARI of the generated clusterings in relation to the known ground labels. Our tables consist of two major columns. Under "Original", we present the results provided by the traditional algorithms by simply applying a CVI (SW or CH) to all generated clustering solutions, and then selecting the solution corresponding to the highest CVI value. The ARI between this solution and the known ground truth is then reported. Under "Consensus", we present the ARI results obtained after carrying out the central consensus versions of MWK-means and iMWK-means algorithms. In this case, the results of iMWK-means algorithm are not presented for SW and CH indices because our iMWK-means consensus strategy is not based on multiple random starts of the algorithm (see Section 3).

Table 1 reports the ARI results for 12 real-world datasets without noise features. Here, our consensus method applied in the framework of MWK-means, using the Silhouette index as CVI, was equal or outperformed the original MWK-means for 10 of the 12 considered datasets. For the two datasets where our method was unable to perform equally or better, i.e., Vehicle Silhouette and Wine data, the ARI difference was very small (0.011 and

0.015, respectively). In the same table, the experiments with MWK-means applied with the CH index provided equal or better results for the consensus strategy for 11 of the 12 datasets. The Wine dataset was the only one for which the original MWK-means strategy provided a higher ARI value. However, the difference in the ARI values in this case was 0.03, only. The results generated by our iMWK-means consensus strategy were rather mixed. Here, the consensus algorithm was able to provide equal or better results than the SW or CH-based original iMWK-means strategies for 7 of the 12 datasets.

Table 1: ARI results for 12 real-world datasets without noise features. Results under "Original" are the ARIs calculated between the clustering solutions obtained using the original partitioning algorithms, based on the highest CVI approach, and the known ground truth labels. Results under "Consensus" are the ARIs calculated between the clustering solutions obtained using the consensus versions of partitioning algorithms and the known ground truth labels.

	Original						Consensus		
	KM		MWK		iMWK		MWK		iMWK
	SW	CH	SW	CH	SW	CH	SW	CH	
AustraCC	0.499	0.499	0.001	0.504	0.001	0.504	0.504	0.504	0.409
BreastOriginal	0.839	0.839	0.797	0.828	0.823	0.828	0.828	0.855	0.839
Ecoli	0.693	0.454	0.707	0.424	0.004	0.038	0.760	0.564	0.038
Glass	0.560	0.483	0.274	0.507	0.306	0.306	0.292	0.557	0.231
Hepatitis	0.190	0.141	0.396	0.122	0.138	0.149	0.396	0.122	0.149
Ionosphere	0.168	0.168	0.004	0.173	0.209	0.168	0.004	0.173	0.209
Iris	0.716	0.716	0.716	0.716	0.802	0.802	0.745	0.745	0.886
PimaIndians	0.011	0.102	0.008	0.096	0.082	0.094	0.100	0.100	0.077
TuluguVowels	0.534	0.377	0.467	0.356	0.428	0.496	0.467	0.358	0.496
VehicleSil	0.078	0.076	0.085	0.081	0.074	0.075	0.074	0.083	0.086
Wine	0.868	0.868	0.850	0.867	0.819	0.819	0.835	0.837	0.772
Yeast	0.168	0.157	0.172	0.168	0.003	0.012	0.182	0.175	0.003

Table 2 presents the results of our experiments with benchmark data to which we added 50% of noise features. Again, we measured the cluster recovery in terms of ARI in relation to the known ground truth. Our consensus strategy carried out in the framework of

MWK-means, using the Silhouette index as CVI, was equal or outperformed the original MWK-means for 10 of the 12 real datasets affected by noise. It is worth noting that the ARI differences for the two remaining datasets were really minor, i.e., 0.015 for Vehicle Silhouette and 0.003 for Yeast. On the other hand, the experiments with MWK-means used with the CH index provided equal or better results for the consensus strategy for 9 of the 12 datasets. The results generated by the iMWK-means consensus algorithm were again mixed. Here, the consensus technique was able to provide equal or better results than the SW or CH-based original iMWK-means algorithms for 7 of the 12 datasets.

Table 2: ARI results for 12 real-world datasets with 50% of noise features added to the data. Table 1 column description applies here.

	Original						Consensus		
	KM		MWK		iMWK		MWK		iMWK
	SW	CH	SW	CH	SW	CH	SW	CH	
AustraCC	0.499	0.499	0.001	0.504	0.504	0.216	0.504	0.504	0.504
BreastOriginal	0.839	0.839	0.855	0.872	0.855	0.872	0.866	0.861	0.866
Ecoli	0.626	0.394	0.737	0.444	0.004	0.038	0.737	0.403	0.038
Glass	0.304	0.408	0.250	0.491	0.303	0.303	0.292	0.555	0.217
Hepatitis	0.243	0.122	0.036	0.122	0.396	0.150	0.407	0.122	0.113
Ionosphere	0.168	0.168	0.004	0.173	0.209	0.096	0.004	0.183	0.209
Iris	0.730	0.730	0.445	0.730	0.757	0.851	0.445	0.730	0.851
PimaIndians	0.011	0.103	0.002	0.100	0.100	0.087	0.099	0.100	0.101
TuluguVowels	0.402	0.388	0.417	0.402	0.478	0.488	0.439	0.403	0.479
VehicleSil	0.079	0.075	0.088	0.080	0.075	0.074	0.073	0.082	0.088
Wine	0.869	0.847	0.867	0.819	0.833	0.833	0.867	0.867	0.785
Yeast	0.069	0.069	0.109	0.174	0.003	0.012	0.103	0.099	0.012

Table 3 reports the results of our simulations with synthetic data composed of spherical Gaussian clusters without noise features. The presented results are the averages taken over 100 datasets for each considered parameter combination. The standard deviations of the obtained ARIs are also reported. Here, our median-based consensus strategy used in the framework of MWK-means algorithm with either SW or CH validity indices provided no improvement compared to the original MWK-means algorithm. The corresponding differences between the average ARI results are often negligible, being most of the times below 0.01. However, the application of our central consensus strategy in the framework of iMWK-means allowed us to improve the results of the original iMWK-means used with either SW or CH for all the six parameter configurations considered in our simulations. The greater the number of features and clusters, the greater this improvement was. For instance, under the configuration 1000x40-8 the improvement in the ARI results provided by consensus iMWK-means was 0.196 in the case of SW and 0.165 in the case of CH.

Finally, Table 4 illustrates the results of our experiments with synthetic data with 50% of added noise features. Here, all of the presented original and consensus MWK and iMWK strategies clearly outperformed the traditional K-means algorithms which did not cope well with the noise features added to the data. We can also observe an important improvement that the consensus MWK-means strategy provided over original MWK-means in the case of both SW and CH optimization for the 1000x8-4, 1000x10-5 and 1000x20-6 data configurations. In the case of iMWK-means, a clear improvement yielded by the consensus strategy can be observed for five of the six data configurations.

Table 3: Average ARI results provided by the K-means, MWK-means and iMWK-means algorithms for our synthetic data composed of spherical Gaussian clusters without noise features. The standard deviation of each ARI is indicated after a slash. Table 1 column description applies here.

	Original						Consensus		
	KM		MWK		iMWK		MWK		iMWK
	SW	CH	SW	CH	SW	CH	SW	CH	
-2	0.470/0.33	0.458/0.33	0.472/0.32	0.457/0.33	0.426/0.32	0.423/0.33	0.463/0.32	0.454/0.32	0.434/0.31
-3	0.586/0.23	0.552/0.23	0.620/0.21	0.568/0.23	0.527/0.23	0.537/0.22	0.613/0.21	0.561/0.22	0.549/0.23
-4	0.649/0.19	0.596/0.20	0.675/0.19	0.607/0.20	0.537/0.17	0.559/0.20	0.665/0.19	0.606/0.20	0.605/0.21
-5	0.685/0.18	0.650/0.19	0.712/0.16	0.660/0.18	0.594/0.17	0.575/0.15	0.706/0.17	0.664/0.18	0.625/0.18
-6	0.933/0.09	0.913/0.12	0.930/0.07	0.926/0.11	0.740/0.16	0.767/0.17	0.933/0.08	0.934/0.08	0.883/0.16
-8	0.992/0.01	0.980/0.07	0.988/0.02	0.995/0.02	0.780/0.16	0.811/0.15	0.996/0.01	0.994/0.02	0.977/0.03

Table 4: Average ARI results provided by the K-means, MWK-means and iMWK-means algorithms for our synthetic data composed of spherical Gaussian clusters with 50% of added noise features. The standard deviation of each ARI is indicated after a slash. Table 1 column description applies here.

	Original						Consensus		
	KM		MWK		iMWK		MWK		iMWK
	SW	CH	SW	CH	SW	CH	SW	CH	
-2	0.001/0.00	0.001/0.00	0.001/0.00	0.000/0.00	0.150/0.31	0.141/0.31	0.000/0.00	0.000/0.000	0.067/0.23
-3	0.050/0.11	0.010/0.06	0.117/0.20	0.010/0.06	0.142/0.20	0.176/0.24	0.100/0.20	0.010/0.060	0.180/0.26
-4	0.053/0.13	0.053/0.12	0.117/0.20	0.072/0.15	0.320/0.26	0.283/0.24	0.152/0.21	0.105/0.170	0.371/0.26
-5	0.078/0.11	0.069/0.10	0.246/0.25	0.114/0.15	0.379/0.23	0.375/0.22	0.331/0.25	0.183/0.170	0.432/0.25
-6	0.445/0.25	0.424/0.22	0.865/0.14	0.729/0.19	0.793/0.17	0.790/0.17	0.893/0.13	0.914/0.120	0.825/0.15
-8	0.859/0.18	0.789/0.19	0.983/0.03	0.993/0.01	0.859/0.09	0.862/0.09	0.970/0.04	0.981/0.030	0.966/0.15

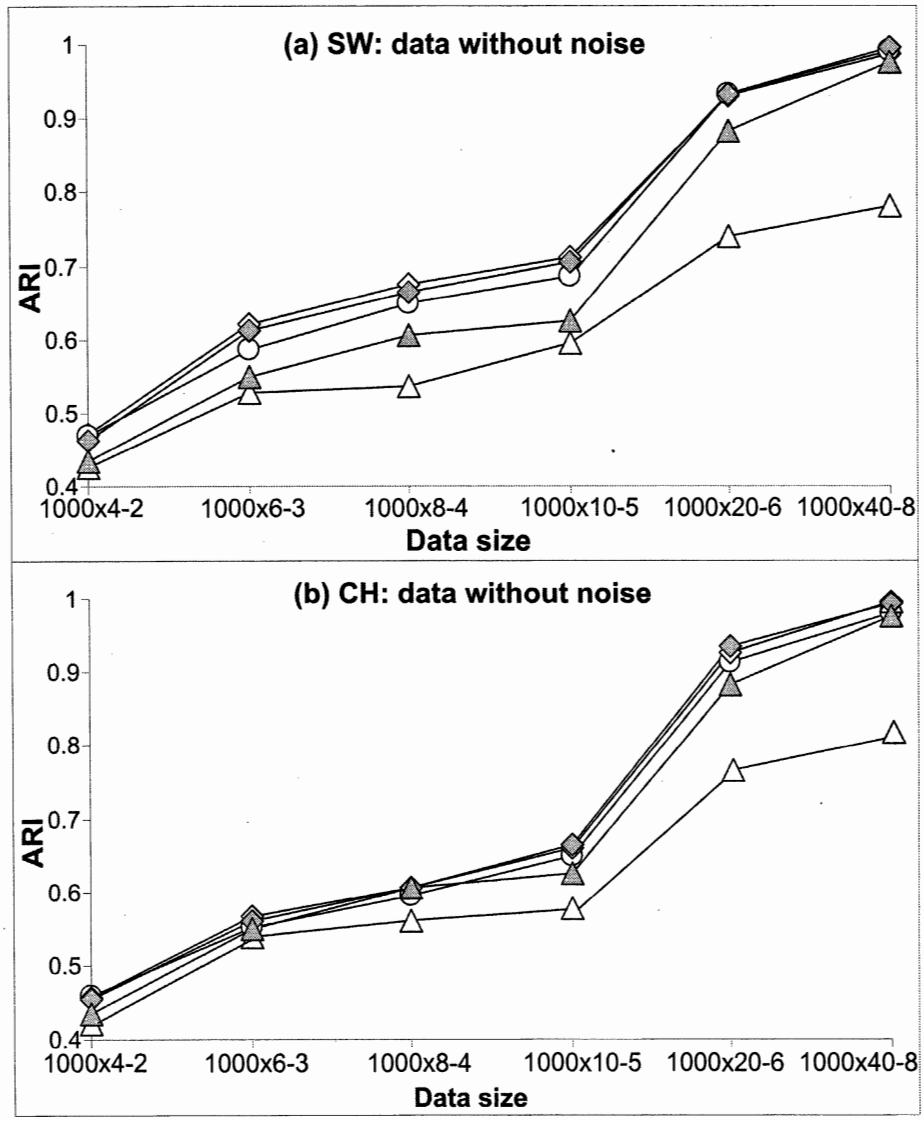


Figure 1: Average ARI results provided by the K-means, MWK-means and iMWK-means algorithms for our original synthetic data without noise, used with the Silhouette width (SW, panel a) and Calinski-Harabasz (CH, panel b) cluster validity indices. The results of the original K-means algorithm are represented by open circles, of the original MWK-means algorithm by open diamonds, and of the original iMWK-means algorithm by open triangles. The results of our median-based central consensus algorithms are represented by gray diamonds (MWK-means) and gray triangles (iMWK-means).

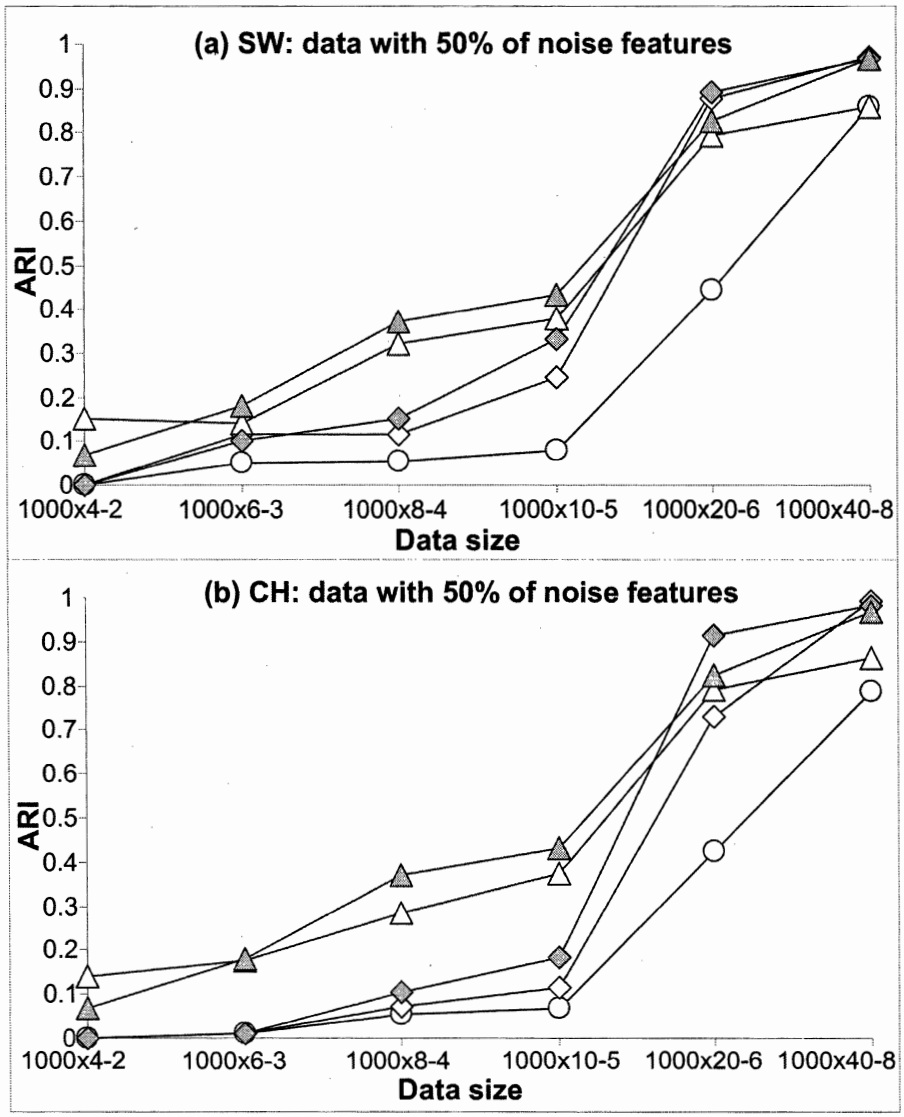


Figure 2: Average ARI results provided by the K-means, MWK-means and iMWK-means algorithms for our synthetic data composed of spherical Gaussian clusters with 50% of added noise features. Figure 1 panel description applies here.

6 Conclusion

In this article, we presented an original median-based consensus strategy that can be used in the framework of the weighted Minkowski clustering for selecting an optimal value of the Minkowski exponent p and determining a good consensus partition. Figures 1 and 2 summarize our simulations results obtained for synthetic data consisting of spherical

Gaussian clusters without and with noise features. These graphics confirm that the proposed median-based central consensus rule is effective and should be used in the framework of the weighted Minkowski clustering [5] with both MWK-means and iMWK-means algorithms. The advantages provided by the introduced consensus strategy are noticeable with the noisy data in the case of MWK-means (see Figure 2) and with both original and noisy data in the case of iMWK-means (see Figures 1 and 2). Moreover, both MWK-means and iMWK-means clearly outperform the traditional K-means algorithm when noise features are present in the data. Mention that the Silhouette width generally outperforms the Calinski-Harabasz index in the framework of our consensus MWK-means analysis. These trends have been also confirmed by the analysis of real-world benchmark data (see Tables 1 and 2). It would be interesting to see in the future if the presented central consensus rule could be also used as a cluster validity index to allow one to select the correct number of clusters in the data.

References

[1] Arbelaitz, O., Gurrutxaga, I., Muguerza, J., Prez, J. M., Perona, I.: An extensive comparative study of cluster validity indices. Pattern Recogn. 46, 243-256 (2012)
[2] Ball, G. H., Hall, D. J.: A clustering technique for summarizing multivariate data. Behav. Sci. 12, 153–155 (1967)
[3] Calinski, T., Harabasz, J.: A dendrite method for cluster analysis. Commun. Stat.-Theor. M. 3, 1–27 (1974)
[4] Chan, E. Y., Ching, W. K., Ng, M. K., Huang, J. Z.: An optimization algorithm for clustering using weighted dissimilarity measures. Pattern Recogn. 37, 943–952 (2004)
[5] de Amorim, R. C., Mirkin, B.: Minkowski metric, feature weighting and anomalous cluster initializing in K-means clustering. Pattern Recogn. 45, 1061–1075 (2012)
[6] Field, A.: Discovering Statistics Using SPSS. SAGE Publications (2005)
[7] Huang, J. Z., Ng, M. K., Rong, H., Li, Z.: Automated variable weighting in k-means type clustering. IEEE Trans. Pattern Anal. Mach. Intell. 27, 657–668 (2005)
[8] Hubert, L., Arabie, P.: Comparing Partitions. J. Classif. 2, 193–218 (1985)
[9] Jain, A.K.: 50 years beyond K-means. Pattern Recogn. Lett. 31, 651–666 (2010)
[10] Ji, J., Bai, T., Zhou, C., Ma, C., Wang, Z.: An improved k-prototypes clustering algorithm for mixed numeric and categorical data. Neurocomputing. 120, 590–596 (2013)
[11] Jones, E., Oliphant, T., Peterson, P., others: SciPy: Open source scientific tools for Python. In: R foundation for statistical computing. Vienna, Austria (2011) Available via DIALOG.
[12] Lichman, M.: UCI Machine Learning Repository. In: University of California, Irvine, School of Information and Computer Sciences (2013) Available via DIALOG.
[13] MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: Broy, M., Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, pp. 281-297. California, USA (1967)
[14] Makarenkov V., Legendre P.: Optimal variable weighting for ultrametric and additive trees and K-Means partitioning. J. Classif. 169, 245–271 (2001)
[15] MATLAB: MATLAB:2010. The MathWorks Inc., Natick, Massachusetts (2010)

[16] Milligan, G. W., Cooper, M. C.: An examination of procedures for determining the number of clusters in a data set. Psychometrika. 50, 159–179 (1985)
[17] Mirkin, B.: Clustering: A Data Recovery Approach. CRC Press, London (2012)
[18] Murtagh, F.: Complexities of hierarchic clustering algorithms: State of the art. Computation Stat. 1, 101–113 (1984)
[19] Murtagh, F., Contreras, P.: Methods of hierarchical clustering. arXiv preprint arXiv:1105.0121 (2011)
[20] Pal, S. K., Majumder, D. D.: Fuzzy sets and decision making approaches in vowel and speaker recognition. IEEE Trans. Syst., Man, Cyber. 7, 625–629 (1977)
[21] Pollard, K. S., Van Der Laan, M. J.: A method to identify significant clusters in gene expression data. bepress, 318–325 (2002)
[22] R Core Team: R: A Language and Environment for Statistical Computing. In: R Foundation for Statistical Computing. Vienna, Austria (2013) Available via DIALOG.
[23] Rousseeuw, P. J.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J. Comput. Appl. Math. 20, 53–65 (1987)
[24] Steinley, D.: K-means: a half-century synthesis. Brit. J. Math. Stat. Psy. 59, 1–34 (2006)

METHODOLOGY ARTICLE

A new fast method for inferring multiple consensus trees using k-medoids

Nadia Tahiri, Matthieu Willems and Vladimir Makarenkov*

*Correspondence:
tahiri.nadia@courrier.uqam.ca
willems.matthieu@courrier.uqam.ca
*makarenkov.vladimir@uqam.ca
Département d'Informatique, Université du Québec à Montréal, Case postale 8888, Succursale Centre-ville, H3C 3P8 Montréal, CA
Full list of author information is available at the end of the article

Abstract

Background: Gene trees carry important information about specific evolutionary patterns which characterize the evolution of the corresponding gene families. However, a reliable species consensus tree cannot be inferred from a multiple sequence alignment of a single gene family or from the concatenation of alignments corresponding to gene families having different evolutionary histories. These evolutionary histories can be quite different due to horizontal transfer events or to ancient gene duplications which cause the emergence of paralogs within a genome. Many methods have been proposed to infer a single consensus tree from a collection of gene trees. Still, the application of these tree merging methods can lead to the loss of specific evolutionary patterns which characterize some gene families or some groups of gene families. Thus, the problem of inferring multiple consensus trees from a given set of gene trees becomes relevant.

Results: We describe a new fast method for inferring multiple consensus trees from a given set of phylogenetic trees (i.e. additive trees or X-trees) defined on the same set of species (i.e. objects or taxa). The traditional consensus approach yields a single consensus tree. We use the popular k-medoids partitioning algorithm to divide a given set of trees into several clusters of trees. We propose novel versions of the well-known Silhouette and Caliński-Harabasz cluster validity indices that are adapted for tree clustering with k-medoids. The efficiency of the new method was assessed using both synthetic and real data, such as a well-known phylogenetic dataset consisting of 47 gene trees inferred for 14 archaeal organisms.

Conclusions: The method described here allows inference of multiple consensus trees from a given set of gene trees. It can be used to identify groups of gene trees having similar intragroup and different intergroup evolutionary histories. The main advantage of our method is that it is much faster than the existing tree clustering approaches, while providing similar or better clustering results in most cases. This makes it particularly well suited for the analysis of large genomic and phylogenetic datasets.

Keywords: cluster validity index; consensus tree; k-medoids; phylogenetic tree; Robinson and Foulds topological distance

Background

Various methods for computing a consensus tree for a given set of phylogenetic trees have been proposed [1]. The most known types of consensus trees are the strict consensus tree, the majority consensus tree and the extended majority consensus tree [1, 2]. The strict consensus tree contains only the edges that are common to all input trees. The majority consensus tree contains the edges that are present in more than 50% of the input trees, although higher percentages may also be considered. According to the extended majority rule, the consensus tree includes all of the majority edges to which compatible residual edges are added gradually, starting with the most frequent ones. Extended majority consensus trees are the most frequently used consensus trees in evolutionary biology because they are usually much better resolved (i.e. have lower total degree of internal nodes) than strict and majority consensus trees [2].

The output of most conventional consensus tree algorithms is a single consensus tree [1]. However, in many practical situations it is much more appropriate to infer several consensus trees. In biology, it is often risky to group phylogenetic trees corresponding to different sets of genes. Each gene has its own evolutionary history which can substantially differ from evolutionary histories of other genes. For example, some individual genes or gene clusters (e.g. operons) affected by specific horizontal gene transfer events will display different evolutionary patterns than the rest of genes under study [3–8]. The evolutionary history of such genes or gene clusters will be depicted by phylogenetic trees having different topologies from that of the species tree which represents the evolution of genes that did not undergo gene transfers. Furthermore, the homogeneity of a given set of genes can be also affected by ancient duplication events causing the emergence of paralogous alleles.

There are several methods for analyzing and visualizing sets of incompatible phylogenetic trees, including SplitsTree [9], Dendroscope [10] and DensiTree [11]. These programs allow for inferring different kinds of phylogenetic networks which can be viewed as alternatives to multiple consensus trees. Holland *et al.* [12] were among the first to discuss a consensus building approach using splits network. Holland *et al.* compared gene trees of yeast genomes and demonstrated that consensus networks can be useful to depict hidden contradictory signals existing in species phylogenies.

Thus, the question whether a unique consensus tree or multiple consensus trees best characterize a given set of phylogenies arises as an alternative to phylogenetic network reconstruction approaches. If the given phylogenies are topologically congruent, they should be combined into a single consensus tree. However, if these phylogenies encompass conflicting genetic signals, they should be organized into multiple consensus trees, each of which accounts for a specific evolutionary pattern [13–15]. Figure 1 shows four phylogenetic trees T_1, T_2, T_3 and T_4 with seven leaves. Here, the solution consisting of two majority-rule consensus trees, T_{12} and T_{34}, seems to be much more appropriate than the solution consisting of a single majority consensus tree, T_{1234}, i.e. a star tree here, given by the traditional majority consensus approach.

In this paper we describe a new algorithm for determining clusters of homogeneous trees which can be combined in order to infer multiple consensus trees. The idea of building several consensus trees was originally formulated by Maddison [16] who found that consensus trees of some subsets of a given set of trees may differ and that they are usually better resolved than the consensus tree of the whole set. Then, Stockham *et al.* [17] proposed two variants of a tree clustering algorithm based on k-means, which were meant to infer a set of strict consensus trees (called characteristic

trees) minimizing the information loss. However, these methods were very expensive in terms of the running time because the consensus trees had to be determined for each set of clusters in all intermediate partitioning solutions tested by k-means. Bonnard *et al.* [13] described a method, called Multipolar Consensus, to display all the splits of a given set of phylogenetic trees having a support above a predefined threshold, using a minimum possible number of consensus trees. The authors indicated that biologically relevant secondary signals, which would be normally absent in a classical consensus tree, can be captured by the Multipolar Consensus method thus providing a convenient exploratory tool for phylogenetic analysis. This method allows one to display more secondary evolutionary signals than it is proposed by the extended majority rule consensus, without making possible arbitrary choices which are usually made in this consensus method. In his recent paper, Guénoche [14] has presented a method for partitioning phylogenetic trees into one cluster ($K = 1$, when given gene trees are homogeneous) or several clusters ($K > 1$, when given gene trees are divergent). A generalized partition score, computed over a set of tree partitions, is calculated by the Guénoche method in order to determine the number of clusters, K, in which a given set of gene trees should be partitioned. Guénoche validated his method on both simulated data, i.e. random sets of trees organized in different topological groups, and real data, i.e. a set of non homogeneous gene trees of 30 *E. coli* strains assumed to be affected by horizontal gene transfers. The MCT (Multiple Consensus Trees) program developed by the author remains one of the rare pieces of software for inferring multiple consensus trees, available for the research community.

We will describe a new tree clustering method that relies on specific versions of the Silhouette (SH) [18] and Caliński-Harabasz (CH) [19] indices adapted for tree clustering with k-medoids. These cluster validity indices will be used to determine the best partitioning obtained over multiple random starts of k-medoids [20] when the number of clusters is fixed and then to select the optimal number of clusters for a given set of trees.

Methods

K-medoids algorithm adapted for tree clustering

A phylogenetic tree is an unrooted leaf-labeled tree in which each internal node, representing an ancestor of contemporary species, has at least two children and all leaves, representing contemporary species, have different labels [2, 21, 22]. Our algorithm takes as input a set of phylogenetic trees Π defined on the same set of leaves and returns as output one or several consensus trees. Each consensus tree represents a subset (i.e. group, class or cluster) of trees from Π. For each cluster identified, the algorithm returns a list of its elements (i.e. phylogenetic trees) and the corresponding consensus tree. The output is also accompanied by some statistics (e.g. the value of the selected cluster validity index). Our method uses a version of the popular k-medoids algorithm suitable for tree clustering. The k-medoids algorithm [20] is a clustering method which can be viewed as a robust version of the popular k-means algorithm. The k-medoids algorithm divides N elements (i.e. phylogenetic trees in our case) into K clusters using the cluster centers (i.e. the medoids) which belong to the set of original elements (i.e. original trees from Π in our case). The medoid of a given cluster is chosen to minimize the overall distance to the other elements of this cluster. The content of each cluster is chosen to minimize the total intracluster distance. Generally, the most commonly used distances in the framework of k-medoids are the Euclidean distance, the Manhattan distance and the Minkowski distance.

Several measures have been proposed to estimate the distance between phylogenetic trees. The most popular of them are the Robinson and Foulds (RF) topological distance [23], the quartet

distance [24], the SPR (Subtree Prune and Regraft) distance [25], the MAST (Maximum Agreement Subtree) distance [26] and the bipartition dissimilarity [3]. Unfortunately, the popular SPR distance, which is often used to identify horizontal gene transfer events, takes an exponential time to calculate. This is also the case of the MAST distance which cannot be calculated in polynomial time. The *RF* and the quartet distances are topological distances which are among the quickest to calculate. Indeed, both of them can be computed in $\mathcal{O}(n^2)$ when two Newick strings, representing two phylogenies with n leaves defined on the same set of species, are considered. Moreover, Barthélemy and McMorris [27] have shown that the majority consensus tree of a set of trees is a median tree of this set in the sense of the *RF* distance [27]. Thus, the *RF* topological distance seems to be an appropriate distance to be used within k-means or k-medoids algorithms adapted for phylogenetic tree clustering. In our work, we define the median tree as a tree based on the *RF* distance. It is worth noting that one could also use an alternative type of median trees, those based on the SPR distance [28], to infer SPR-distance-based multiple consensus trees. For instance, Bruen and Bryant showed that the maximum parsimony tree can be viewed as a type of median consensus tree in the sense of the SPR distance.

The Robinson and Foulds distance [14,23,29,30] between two trees is a well-known distance used in computational biology to compare the topologies of two phylogenetic trees defined on the same set of species. The Robinson and Foulds distance is a topological distance. It does not take into account the lengths of the tree edges. The time complexity of a typical implementation of the k-medoids algorithm is $\mathcal{O}(K \times (N-K)^2 \times i \times M)$, where K is the number of clusters, i is the number of iterations in k-medoids and M is the number of variables characterizing each of the N objects. One of the advantages of our new algorithm is that it does not need to recompute the consensus trees for intermediate clusters of trees. Instead, it estimates the quality of each intermediate clustering using an approximate formula. This allows a much faster partitioning of a given set of phylogenetic trees into K clusters without losing the quality of the obtained consensus trees.

In the case of tree clustering using k-means, the objective function to be minimized can be defined as follows:

$$OF = \sum_{k=1}^{K} \sum_{i=1}^{N_k} RF(T_k^{maj}, T_{ki}), \tag{1}$$

where K is the number of clusters, N_k is the number of trees in cluster k, *RF* is the Robinson and Foulds topological distance between two phylogenetic trees with n leaves, T_{ki} is the tree i of cluster k and T_k^{maj} is the majority rule consensus tree of cluster k. Still, the computation of the majority rule consensus tree or of the extended majority rule consensus tree is time-consuming. The time complexity of the method computing the majority or the extended majority rule consensus tree is $\mathcal{O}(n^2 + nN^2)$ [2], where N is the number of trees and n is the number of leaves in each tree. Jansson *et al.* [31] have recently proposed a number of deterministic algorithms for constructing the majority rule consensus tree and some of its variants. The authors presented the algorithms running in $\mathcal{O}(nN \times logN)$ time - for a majority rule consensus tree, in $\mathcal{O}(nN)$ time - for a loose consensus tree, and in $\mathcal{O}(n^2N)$ time - for a greedy consensus tree. However, if the trees are defined by their Newick strings, as it is typically done in evolutionary biology [32], one will need a conversion program to transform each Newick string into the format required by the algorithms of Jansson *et al.*

Stockham *et al.* [17] originally proposed two variants of the popular k-means algorithm to infer a set of strict consensus trees (called characteristic trees) that minimize the loss information. However, the approach of Stockham *et al.* seems to be very expensive in terms of the running time because the consensus trees must be determined for each set of clusters in all intermediate partitioning solutions tested by k-means. We present a new algorithm that does not need to recompute the consensus trees at each of its iterations. The time complexity of a straightforward tree partitioning algorithm, such as the algorithm of Stockham *et al.* [17] based on k-means, which recomputes the consensus trees after each basic k-means operation, consisting of relocating an object (i.e. tree) from one cluster to another and then in reassessing the value of the objective function (Formula 1), is $\mathcal{O}(K \times n \times (n + N^2) \times i)$.

We propose to use the following approximate formula, based on the properties of k-medoids:

$$OF_{med} = \sum_{k=1}^{K} \sum_{i=1}^{N_k} RF(T_k^m, T_{ki}), \tag{2}$$

where T_k^m is the medoid tree of cluster k. The medoid tree T_k^m of cluster k is the tree belonging to cluster k that minimizes the sum of RF distances between this tree and all other trees in k. By contrast with the k-means-based approach, we do not need to compute in our algorithm cluster centroids or majority consensus trees of clusters. Using Formula 2, we reduce the time complexity of the method to $\mathcal{O}(nN^2 + K \times (N-K)^2 \times i)$, where $\mathcal{O}(nN^2)$ is the time complexity of precalculating the matrix of pairwise RF distances, of size $(N \times N)$, between all trees in Π.

Silhouette cluster validity index adapted for tree clustering with k-medoids

The first cluster validity index we consider in this study is the Silhouette width (SH) [18]. This index assesses the average rate of similarity between the objects belonging to the same cluster (i.e. a cohesion function) versus the rate of similarity between the objects of different clusters (i.e. a separation function). The Silhouette width for cluster k is defined as follows:

$$sh(k) = \frac{1}{N_k} \left[\sum_{i=1}^{N_k} \frac{b(i) - a(i)}{\max(a(i), b(i))} \right], \tag{3}$$

where N_k is the number of elements (i.e. trees) in cluster k, $a(i)$ is the average distance between the element i and all other elements of k and $b(i)$ is the smallest of all distances between the element i of cluster k and the elements in the other clusters (i.e. those different from k). The optimal number of clusters corresponds to the highest value of Silhouette.

The following equations for calculating $a(i)$ and $b(i)$ can be used when clustering trees. The formula for $a(i)$ is as follows:

$$a(i) = \frac{\sum_{j=1}^{N_k} RF(T_{ki}, T_{kj})}{N_k}, \tag{4}$$

where T_{ki} is the tree i of cluster k and T_{kj} is the tree j of cluster k. The formula for $b(i)$ is as follows:

$$b(i) = \min_{1 \leq k' \leq K, \text{ and } k' \neq k} \frac{\sum_{j=1}^{N_{k'}} RF(T_{ki}, T_{k'j})}{N_{k'}}, \tag{5}$$

where $T_{k'j}$ is the tree j of cluster k' and $N_{k'}$ is the number of trees in cluster k'.

Finally, the optimal number of clusters K corresponds to the maximum average Silhouette width, $SH(K)$, defined as follows:

$$SH(K) = \sum_{k=1}^{K} [sh(k)] / K. \tag{6}$$

Caliński-Harabasz cluster validity index adapted for tree clustering with k-medoids

The second cluster validity index we consider here is the Caliński-Harabasz index (CH) [19]. This criterion is a ratio between the overall between-cluster distance and the overall within-cluster distance, calculated by taking into account the number of degrees of freedom. The exact formula for the CH computation is as follows:

$$CH = \frac{SS_B}{SS_W} \times \frac{N-K}{K-1}, \tag{7}$$

where SS_W is the overall within-cluster distance involving the elements of the same cluster, SS_B is the overall between-cluster distance involving the elements of different clusters, K is the number of clusters and N is the number of elements. The optimal number of clusters corresponds to the maximum of CH.

The traditional formula for calculating the overall within-cluster variance, SS_W, is as follows:

$$SS_W = \sum_{k=1}^{K} \sum_{y \in C_k} \|y - m_k\|^2, \tag{8}$$

where y is an element of cluster C_k, m_k is the centroid of cluster k and $\|y - m_k\|^2$ is the Euclidean distance (L^2 norm) between y and m_k.

The traditional formula for calculating the overall between-cluster variance, SS_B, is as follows:

$$SS_B = \sum_{k=1}^{K} N_k \|m_k - m\|^2, \tag{9}$$

where N_k is the number of trees in cluster k, m is the overall mean of the sample data and $\|m_k - m\|^2$ is the Euclidean distance between m_k and m. Good clusterings have a large overall between-cluster distance (SS_B) and a small overall within-cluster distance (SS_W).

Clearly, we cannot use the Euclidean distance when clustering trees [33]. The Robinson and Foulds topological distance (RF) has been used instead. Namely, the following formulas for the overall within-cluster and between-cluster distances were used in our study:

$$SS_W = \sum_{k=1}^{K} \sum_{i=1}^{N_k} RF(T_k^m, T_{ki}), \text{ and} \tag{10}$$

$$SS_B = \sum_{k=1}^{K} N_k \times RF(T_k^m, T^m), \tag{11}$$

where T_k^m is the medoid tree of cluster k, T_{ki} is the tree i of cluster k and T^m is the medoid tree of the sample data.

Adjusted Rand index

The quality of clustering results was evaluated by using the Adjusted Rand Index (ARI) [34–36]. The values of ARI are located in the interval [-1; 1]. When two partitions are exactly the same, the corresponding value of ARI is 1. This popular index is the corrected for chance version of the Rand index [37,38]. ARI is often used in simulations to compare the known original partitions with those generated by methods under study.

Given a set of n objects and two partitions of these objects, namely $X = X_1, X_2, \ldots, X_r$ with r clusters and $Y = Y_1, Y_2, \ldots, Y_s$ with s clusters, the overlap between X and Y can be summarized using a contingency matrix $[n_{ij}]$, where each entry n_{ij} denotes the number of objects in common between partitions X_i and Y_j.
The ARI index is calculated using Formula (12):

$$ARI = \frac{\sum_{ij}\binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}\right] / \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \left[\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}\right] / \binom{n}{2}}, \tag{12}$$

where $n_{ij} = |X_i \cap Y_j|$, $a_i = \sum_{j=1}^{s} |X_i \cap Y_j|$ and $b_j = \sum_{i=1}^{r} |X_i \cap Y_j|$, and X_i and Y_j are the sets of objects in clusters i and j, respectively.

Results and Discussion

Simulation design

We tested our new algorithm for computing multiple consensus trees using the two following simulation protocols.

Our first simulation included two main steps. During the first step, we randomly generated a species phylogenetic tree (i.e. first consensus tree here) T_1 with n leaves using the HybridSim [39] program. Then, using the same program, we generated $K-1$ other consensus trees, $T_2, \ldots, T_K$ with n leaves, each of which differed from T_1 by a specified number of hybridization events (the value of the *hybridization rate* parameter in the HybridSim program varied from 1 to 4 in our simulation; it was drawn randomly using a uniform distribution). In our first simulation, the number of clusters, K, ranged from 2 to 10 (see Figs. 2-4), while the number of tree leaves, n, was taking the values 8, 16, 32 and 64 (see Fig. 3).

The HybridSim program developed by Woodhams *et al.* [39] allows generation of phylogenies in the presence of hybridization and horizontal gene transfer events. This program can generate trees differing from each other by a specified number of coalescence/incomplete lineage sorting producing patterns of incongruence across gene trees. In our simulations with HybridSim, we varied the values of the *hybridization rate* (as indicated above) and the *coalescence rate* (as indicated below) parameters. The rest of parameters used in our simulations were the default parameters of HybridSim.

During the second step of the first simulation, for each consensus phylogenetic tree T_i $(i = 1, \ldots, K)$ representing cluster i, we generated a set of 100 trees belonging to cluster i using a specified value of the coalescence rate parameter in HybridSim. In our study, the value of this coalescence parameter,

introducing noise into gene phylogenies, varied between 10 (low noise) and 1 (high noise). Thus, each element T of cluster i differed from the consensus tree T_i of this cluster by a certain (fixed) coalescence degree. The simulation results presented in Figures 2 and 3 correspond to the case in which the coalescence rate parameter in HybridSim was fixed to 5. Figure 4 illustrates how the methods' results change with respect to the change in the coalescence parameter. The number of clusters, K, was assumed to be known in this simulation. The strategies based on both the squared and non-squared RF distances (used in Formula 2) were evaluated.

In the second simulation, we compared our algorithm based on the Silhouette index and the non-squared Robinson and Foulds distance (Formula 2) with the traditional approach based on the recalculation of majority-rule consensus trees, representing the cluster centroids, after each basic operation of k-means (a variant of Formula 1 using the squared RF distance, as suggested by Stockham *et al.* [17]). This comparison was performed in terms of quality of clustering results returned by competing methods (Fig. 5a and b) and of running time (Fig. 5c and d). The number of tree leaves, n, in this second simulation was equal to: 8, 16, 32, 64 and 128. The number of clusters, K, in the second simulation was equal to 5 and the coalescence rate parameter in the HybridSim program was fixed to 5 in this simulation. Once again, 100 different datasets were generated for each parameter combination and the number of clusters was assumed to be known. Our simulations were carried out using a 64-bit computer equipped with an Intel i5-4690T CPU (2.5 GHz) and 8 Gb of RAM.

Simulation study

The results of the first simulation are illustrated in Figures 2, 3 and 4. The presented results are the averages taken over all combinations of our parameters (see the section "Simulation design"), except the featured one (i.e. the number of clusters in Fig. 2, the number of tree leaves in Fig. 3 and the coalescence rate in Fig. 4).

The curves depicted in Figure 2 indicate that the clustering quality provided by our algorithm depends on the number of clusters, the cluster validity index (SH or CH) and the selected objective function (i.e. the non-squared RF distance or the squared RF distance in Formula 2). Obviously, we were not able to address the case of homogeneous data (i.e. $K = 1$) in this study because the SH and CH cluster validity indices are not adapted for the case of one cluster only. The ARI results improve noticeably when the number of clusters, K, increases from 2 to 6 and stabilize starting from 6 clusters. Also, the strategy based on the SH cluster validity index and the non-squared RF distance outperforms the three other competing strategies regardless of the number of clusters.

Figure 3 shows a slight increase in the ARI values as the number of tree leaves increases. Once again, the scores of ARI for the SH criterion are higher than those for the CH criterion and the methods based on the non-squared RF distance are slightly more efficient than those based on the squared RF metric. One can also observe that the ARI scores become more stable when the number of clusters varies between 6 and 10 (Fig. 3b) compared to the case when it varies between 2 and 5 (Fig. 3a).

Figure 4 shows that the CH index is more affected by coalescence (i.e. noise) than SH when clustering trees using k-medoids. Once again, one can notice that both versions of our algorithm, based on SH and CH, yield better results when the non-squared RF distance is used in Formula 2 instead of its squared counterpart. It is worth noting that Stockham *et al.* [17] used the squared RF

distance in their algorithm. Given these results, our second simulation (see Fig. 5) was conducted with the algorithm based on the Silhouette index and the non-squared Robinson and Foulds distance.

The results of the second simulation are shown in Figure 5. The curves presented in Figures 5 (a and b) indicate that the new algorithm based on Formulas 2 to 6 works better than the straightforward tree clustering approach by Stockham *et al.* [17] in terms of the clustering quality when the number of tree leaves varies from 8 to 32, but is slightly less efficient than the approach by Stockham *et al.* when the number of tree leaves varies from 64 to 128. However, our algorithm is by far the best method in terms of the running time for both simulation parameters considered: the number of tree leaves (Fig. 5c) and the number of trees (Fig. 5d). These results suggest that our new algorithm is well suited for the analysis of large phylogenetic datasets.

Clustering trees of 47 ribosomal proteins of Archaea

We applied the new algorithm to analyze the evolution of 47 ribosomal proteins of 14 organisms of Archaea, including 11 species of Euryarchaeota and 3 species of Crenarchaeota. These data were originally studied by Matte-Tailliez *et al.* (see Fig. 1a in [4] or Fig. 6a in our paper). Matte-Tailliez *et al.* inferred a single species phylogenetic tree after the concatenation of the considered protein sequences (see Fig. 6a). However, the evolution of each of these proteins can be represented by its own phylogenetic tree. The cluster analysis of these trees can tell us how many different evolutionary scenarios characterize the evolution of these sequences (i.e. how many clusters of trees exist in the protein tree dataset). We first considered the complete set of 52 multiple sequence alignments of archaeal ribosomal proteins studied by Matte-Tailliez *et al.* We selected for our analysis 47 of these 52 alignments, i.e. those including the data for the same 14 archaeal organisms. The 5 remaining alignments were incomplete (i.e. they included 12 or 13 organisms only). Using the 47 complete alignments, we inferred 47 phylogenetic trees by means of the PHYML method [40] (these alignments and trees are available at: in `https://github.com/TahiriNadia/CKMedoidsTreeClustering/`).

First, we carried out the version of our k-medoids tree clustering algorithm based on the SH index (Formulas 3 to 6) and the non-squared RF distance in order to infer a partitioning of the obtained set of 47 trees. The maximum of SH was attained with five clusters ($K = 5$) which correspond to five different horizontal gene transfer scenarios presented in Figure 6 (panels b to f).
The first cluster contained 11 trees, the second 4 trees, the third 20 trees, the fourth 11 trees and the fifth 1 tree. We inferred the extended majority consensus trees, $SH1$, $SH2$, $SH3$, $SH4$ and $SH5$, for these five clusters of trees. Afterwards, using the gene transfer detection algorithm by Boc *et al.* [3], we identified the scenarios of horizontal gene transfer events which reconcile the species tree (Fig. 6a) and each of the obtained consensus trees, $SH1$ to $SH5$. In the end of the tree reconciliation process, consisting of SPR moves (corresponding to horizontal gene transfers) of clusters of the species tree, the transformed topology of the species tree becomes identical to that of the gene tree. The version of the algorithm available on the T-Rex web site [41] was used in our computations. The obtained transfers account for five different histories which characterize the evolution of the 47 considered ribosomal proteins. Two transfers predicted for these data by Boc *et al.* (see Fig. 6 in [3]), which are in agreement with the results of Matte-Tailliez *et al.* [4] and Boc *et al.* 2013 [42], can be found in these five scenarios. Precisely, the transfers - $SH3-2$ (or its equivalent transfer $SH5-4$) and $SH5-1$ - have been predicted by Boc *et al.* 2010 (see Fig. 6 in [3]), the transfers - $SH3-2$ (or its equivalent transfer $SH5-4$) and $SH3-1$ - have been predicted by Boc *et al.* 2013 (see Fig. 2b in [42]). Finally, the transfers - $SH1-1$ (or its equivalent transfers $SH2-2$ and

$SH4-1$), $SH5-1$, $SH3-2$ (or its equivalent transfer $SH5-4$), $SH2-1$, $SH4-3$ and $SH4-2$ - have been predicted by Boc *et al.* 2013 (see Fig. 3 in [42]) as partial horizontal gene transfers (i.e. transfers leading to the formation of chimeric genes composed of portions of two or more coding sequences; see [43]).

We also used the MCT program by Guénoche [14] (Multiple Consensus Trees) to analyze this Archaea dataset. The average linkage hierarchical algorithm and the Robinson and Foulds distance were the parameters which we selected in MCT. We compared our consensus trees of classes with the consensus trees found by the algorithm by Guénoche [14]. For example for K=5 (this was the optimal number of clusters found using our algorithm with the SH index), the MCT program returned consensus gene trees whose topologies led to the horizontal gene transfers $SH2-1$, $SH2-3$, $SH4-2$ and $SH4-3$ (see Fig. 6).

Second, we carried out the version of our k-medoids tree clustering algorithm based on the CH index (Formulas 7 to 11) and the non-squared RF distance to classify the same set of 47 gene trees. The maximum of CH was attained with three clusters ($K=3$). Here, the first cluster contained 25 trees, the second 14 trees and the third 8 trees. We then inferred the extended majority consensus trees, $CH1$, $CH2$ and $CH3$, for these clusters of trees. Similarly to the case of SH, we identified the scenarios of horizontal gene transfer events that reconcile the species tree (Fig. 6a) and each of the consensus trees $CH1$, $CH2$ and $CH3$ (see Fig. 7, panels a to c). Here, the transfer - $CH2-1$ - has been predicted by Boc *et al.* 2010 (see Fig. 6 in [3]), the transfer - $CH2-2$ - has been predicted by Boc *et al.* 2013 (see Fig. 2b in [42]), and finally the transfers - $CH1-1$ (or its equivalent transfer $CH3-1$), $CH1-2$, $CH1-3$, $CH2-1$, $CH3-2$ and $CH4-3$ - have been predicted by Boc *et al.* 2013 as partial horizontal gene transfer events (see Fig. 3 in [42]). Interestingly, all the transfers found in the horizontal gene transfer scenarios shown in Figure 7 (a-c) can be found in the gene transfer scenarios presented in Figure 6 (b-f).

Finally, we ran the MCT program with K=3 (this was the optimal number of clusters found using our algorithm with the CH index) and compared the obtained consensus trees with those found by our method. The consensus trees found by MCT in this case allowed for four horizontal gene transfers which were equivalent to the transfers $CH1-2$, $CH1-3$, $CH3-2$ and $CH3-3$ (see Fig. 7) found by our algorithm with the CH index.

Still using the horizontal gene transfer detection algorithm by Boc *et al.* [3], we found scenarios of gene transfer events reconciling the species tree (Fig. 7a) and the obtained consensus trees which play the role of gene trees in this context. The overall horizontal gene transfer results comparing the frequencies of the intragroup and intergroup gene transfers found by our algorithm using the SH and CH indices are reported Table 1. They suggest that gene transfers have been more frequent within the species of the same phylum than between the species of different phyla (i.e. Crenarchaeota and Euryarchaeota).

Conclusions

In this article we described a new algorithm for partitioning a set of phylogenetic trees in several clusters in order to infer multiple consensus trees. We presented new formulas allowing for using the popular Silhouette and Caliński-Harabasz cluster validity indices as well as the Robinson and Foulds topological distance in the framework of tree clustering based on the popular k-medoids

algorithm. The new algorithm can be used to address a number of important issues in evolutionary biology, such as the identification of genes having similar evolutionary histories, e.g. those that have undergone the same horizontal gene transfers or those that have been affected by the same ancient duplication events. The presented algorithm could be extended to the case where the input trees have different, but mutually overlapping, sets of leaves. In order to compute the Robinson and Foulds topological distance between such trees, we could first reduce them to the common set(s) of leaves. After this reduction, the Robinson and Foulds distance normalized by its maximum value, which is $2n - 6$ for two binary trees with n leaves, could be used in Formulas 1 and 2 in order to infer multiple consensus trees. Overall, good performances achieved by the new algorithm in terms of both clustering quality and running time makes it well suited for the analysis of large genomic and phylogenetic datasets. A C++ program, called KMTC (K-Medoids Tree Clustering), implementing the discussed tree partitioning algorithm is freely available at: `https://github.com/TahiriNadia/CKMedoidsTreeClustering/`.

List of Abbreviations

ARI: Adjusted Rand Index; **CH**: Caliński-Harabasz cluster validity indice; **KMTC**: K-Medoids Tree Clustering; **MCT**: Multiple Consensus Trees program; **RF**: Robinson and Foulds topological distance; **SH**: Silhouette cluster validity indice; $\mathbf{SS_B}$: overall between-cluster distance; $\mathbf{SS_W}$: overall within-cluster distance; **ToL**: Tree of Life project

Declarations

Acknowledgements

We thank Dr. Bernard Fichet, Dr. Peter G. Foster and two anonymous reviewers for their valuable comments on this manuscript.

Funding

This work was supported by Natural Sciences and Engineering Research Council of Canada and Fonds de Recherche sur la Nature et Technologies of Québec.

Availability of data and materials

All the data presented in this article are available at: https://github.com/TahiriNadia/CKMedoidsTreeClustering.

Author's contributions

NT implemented the program, carried out the simulation study, and wrote the manuscript. MW supervised the study. VM designed and supervised the study, and wrote the manuscript.

Author details

Département d'Informatique, Université du Québec à Montréal, C.P. 8888, succ. Centre-Ville, Montreal, QC H3C 3P8, Canada.

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Competing interests

The authors declare that they have no competing interests.

Additional Files

Additional file 1 — Alignments

The folder contains the complete set of 52 multiple sequence alignments of archaeal ribosomal proteins studied by Matte-Tailliez *et al.*, 2002. These alignments are available at: `https://github.com/TahiriNadia/CKMedoidsTreeClustering/`.

Additional file 2 — Trees

Using the 47 complete alignments, we inferred 47 phylogenetic trees by means of the PHYML method [40] . These trees are available at: `https://github.com/TahiriNadia/CKMedoidsTreeClustering/`.

References

1. Bryant, D.: A classification of consensus methods for phylogenetics. DIMACS series in discrete mathematics and theoretical computer science **61**, 163–184 (2003)
2. Felenstein, J.: Inferring Phylogenies vol. 2. Sinauer associates, Sunderland (2004)
3. Boc, A., Philippe, H., Makarenkov, V.: Inferring and validating horizontal gene transfer events using bipartition dissimilarity. Systematic biology, 103 (2010)
4. Matte-Tailliez, O., Brochier, C., Forterre, P., Philippe, H.: Archaeal phylogeny based on ribosomal proteins. Molecular biology and evolution **19**, 631–639 (2002)
5. Badescu, D., Tahiri, N., Makarenkov, V.: A new fast method for detecting and validating horizontal gene transfer events using phylogenetic trees and aggregation functions. Pattern Recognition in Computational Molecular Biology: Techniques and Approaches, 483–504 (2015)
6. Than, C., Jin, G., Nakhleh, L.: Integrating sequence and topology for efficient and accurate detection of horizontal gene transfer. In: RECOMB-CG, pp. 113–127 (2008). Springer
7. Than, C., Ruths, D., Innan, H., Nakhleh, L.: Confounding factors in hgt detection: statistical error, coalescent effects, and multiple solutions. Journal of computational biology **14**, 517–535 (2007)
8. Than, C., Ruths, D., Nakhleh, L.: Phylonet: a software package for analyzing and reconstructing reticulate evolutionary relationships. BMC bioinformatics **9**, 322 (2008)
9. Huson, D.H., Bryant, D.: Application of phylogenetic networks in evolutionary studies. Molecular biology and evolution **23**(2), 254–267 (2005)
10. Huson, D.H., Scornavacca, C.: Dendroscope 3: an interactive tool for rooted phylogenetic trees and networks. Systematic biology **61**(6), 1061–1067 (2012)
11. Bouckaert, R.R.: Densitree: making sense of sets of phylogenetic trees. Bioinformatics **26**(10), 1372–1373 (2010)
12. Holland, B.R., Huber, K.T., Moulton, V., Lockhart, P.J.: Using consensus networks to visualize contradictory evidence for species phylogeny. Molecular Biology and Evolution **21**(7), 1459–1461 (2004)
13. Bonnard, C., Berry, V., Lartillot, N.: Multipolar consensus for phylogenetic trees. Systematic biology **55**(5), 837–843 (2006)
14. Guénoche, A.: Multiple consensus trees: a method to separate divergent genes. BMC bioinformatics **14**, 46 (2013)
15. Szöllősi, G.J., Daubin, V.: Modeling gene family evolution and reconciling phylogenetic discord. Evolutionary Genomics: Statistical and Computational Methods **2**, 29–51 (2012)
16. Maddison, D.R.: The discovery and importance of multiple islands of most-parsimonious trees. Systematic Biology **40**(3), 315–328 (1991)
17. Stockham, C., Wang, L.-S., Warnow, T.: Statistically based postprocessing of phylogenetic analysis by clustering. Bioinformatics **18**, 285–293 (2002)
18. Rousseeuw, P.J.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Journal of computational and applied mathematics **20**, 53–65 (1987)
19. Caliński, T., Harabasz, J.: A dendrite method for cluster analysis. Communications in Statistics-theory and Methods **3**, 1–27 (1974)
20. Kaufman, L., Rousseeuw, P.J.: Finding Groups in Data: an Introduction to Cluster Analysis vol. 344. John Wiley & Sons, New York (2009)
21. Makarenkov, V., Leclerc, B.: Circular orders of tree metrics, and their uses for the reconstruction and fitting of phylogenetic trees. In: Mathematical Hierarchies and Biology, pp. 183–208 (1996)
22. Leclerc, B., Makarenkov, V.: On some relations between 2-trees and tree metrics. Discrete Mathematics **192**(1-3), 223–249 (1998)
23. Robinson, D.F., Foulds, L.R.: Comparison of phylogenetic trees. Mathematical biosciences **53**, 131–147 (1981)
24. Bryant, D., Tsang, J., Kearney, P.E., Li, M.: Computing the quartet distance between evolutionary trees. In: Symposium on Discrete Algorithms: Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, vol. 9, pp. 285–286 (2000)
25. Hickey, G., Dehne, F., Rau-Chaplin, A., Blouin, C.: The computational complexity of the unrooted subtree prune and regraft distance. Technical report, Technical Report CS-2006-06, Faculty of Computer Science, Dalhousie University (2006)
26. Amir, A., Keselman, D.: Maximum agreement subtree in a set of evolutionary trees: Metrics and efficient algorithms. SIAM Journal on Computing **26**(6), 1656–1669 (1997)
27. Barthélemy, J.-P., McMorris, F.R.: The median procedure for n-trees. Journal of classification **3**(2), 329–334 (1986)
28. Bruen, T.C., Bryant, D.: Parsimony via consensus. Systematic biology **57**(2), 251–256 (2008)
29. Makarenkov, V., Leclerc, B.: Comparison of additive trees using circular orders. Journal of Computational Biology **7**, 731–744 (2000)
30. Dong, J., Fernández-Baca, D., McMorris, F.R.: Constructing majority-rule supertrees. Algorithms for Molecular Biology **5**, 2 (2010)
31. Jansson, J., Shen, C., Sung, W.-K.: Improved algorithms for constructing consensus trees. Journal of the ACM (JACM) **63**(3), 28 (2016)

32. Felsenstein, J.: Numerical Taxonomy vol. 1. Springer, New York (2013)
33. Fichet, B.: K-means et Plongements Isométriques: la Distance de Robinson-Foulds Comme Exemple, Lyon, France (2017). Proceeding of Société Francophone de Classification 2017
34. Hubert, L., Arabie, P.: Comparing partitions. Journal of classification **2**(1), 193–218 (1985)
35. Steinley, D.: Properties of the hubert-arable adjusted rand index. Psychological methods **9**, 386 (2004)
36. Steinley, D., Brusco, M.J., Hubert, L.: The variance of the adjusted rand index. Psychological methods **21**, 261 (2016)
37. Rand, W.M.: Objective criteria for the evaluation of clustering methods. Journal of the American Statistical association **66**, 846–850 (1971)
38. Hoffman, M., Steinley, D., Brusco, M.J.: A note on using the adjusted rand index for link prediction in networks. Social Networks **42**, 72–79 (2015)
39. Woodhams, M.D., Lockhart, P.J., Holland, B.R.: Simulating and summarizing sources of gene tree incongruence. Genome biology and evolution **8**(5), 1299–1315 (2016)
40. Guindon, S., Gascuel, O.: A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood. Systematic biology **52**, 696–704 (2003)
41. Boc, A., Diallo, A.B., Makarenkov, V.: T-rex: a web server for inferring, validating and visualizing phylogenetic trees and networks. Nucleic acids research **40**(W1), 573–579 (2012)
42. Boc, A., Legendre, P., Makarenkov, V.: An efficient algorithm for the detection and classification of horizontal gene transfer events and identification of mosaic genes. In: Algorithms from and for Nature and Life, pp. 253–260. Springer, Francfort (2013)
43. Boc, A., Makarenkov, V.: Towards an accurate identification of mosaic genes and partial horizontal gene transfers. Nucleic acids research **39**(21), 144–144 (2011)

Figures

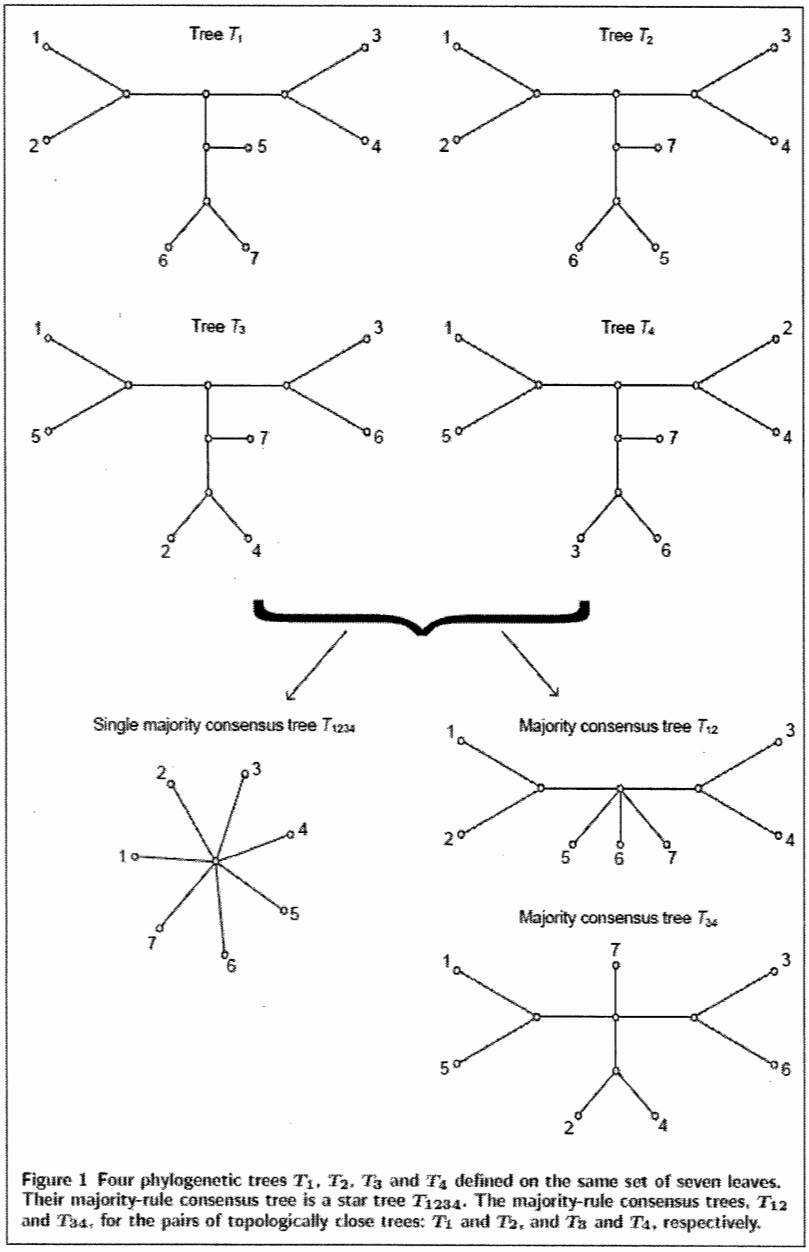


Figure 1 Four phylogenetic trees T_1, T_2, T_3 and T_4 defined on the same set of seven leaves. Their majority-rule consensus tree is a star tree T_{1234}. The majority-rule consensus trees, T_{12} and T_{34}, for the pairs of topologically close trees: T_1 and T_2, and T_3 and T_4, respectively.

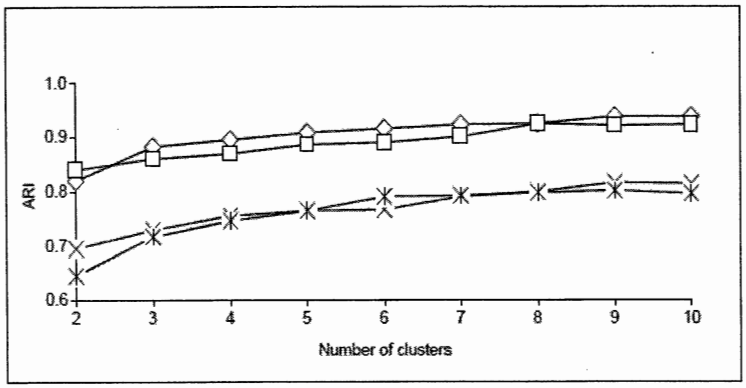


Figure 2 Classification performances of the four versions of our k-medoids tree clustering algorithm in terms of ARI with respect to the number of clusters, ranging from 2 to 10. The four tested versions of our algorithm were those based on: 1) SH with RF ($\diamond$), 2) CH with RF ($\times$), 3) SH with RF squared ($\square$) and 4) CH with RF squared ($*$). The coalescence rate parameter in the HybridSim program was fixed to 5 in this simulation. The presented results are the averages taken over all considered numbers of tree leaves.

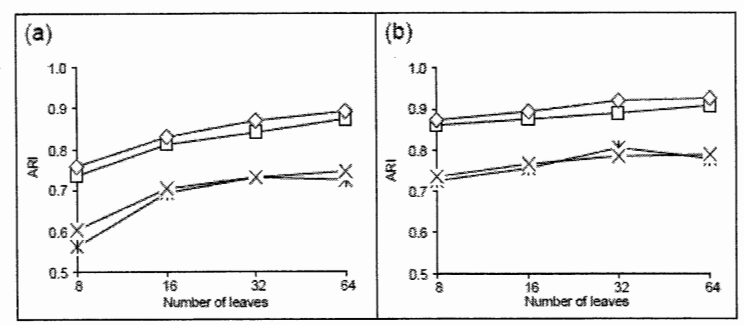


Figure 3 Classification performances of the four versions of our k-medoids tree clustering algorithm in terms of ARI with respect to the number of tree leaves: (a) the case of 2 to 5 clusters and (b) the case of 6 to 10 clusters. The four tested versions of our algorithm were based on: 1) SH with RF ($\diamond$), 2) CH with RF ($\times$), 3) SH with RF squared ($\square$) and 4) CH with RF squared ($*$). The coalescence rate parameter in the HybridSim program was fixed to 5 in this simulation. The presented results are the averages taken over all considered numbers of clusters.

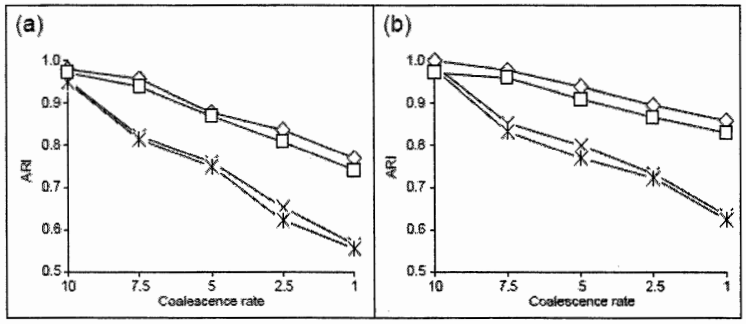


Figure 4 Classification performances of the four tested versions of our k-medoids tree clustering algorithm in terms of ARI with respect to the coalescence rate: (a) the case of 2 to 5 clusters and (b) the case of 6 to 10 clusters. The four tested versions of our algorithm were based on: 1) SH with RF ($\diamond$), 2) CH with RF ($\times$), 3) SH with RF squared ($\square$) and 4) CH with RF squared ($*$). The coalescence rate parameter in the HybridSim program varied from 10 to 1 in this simulation. The presented results are the averages taken over all considered numbers of clusters and all considered numbers of tree leaves.

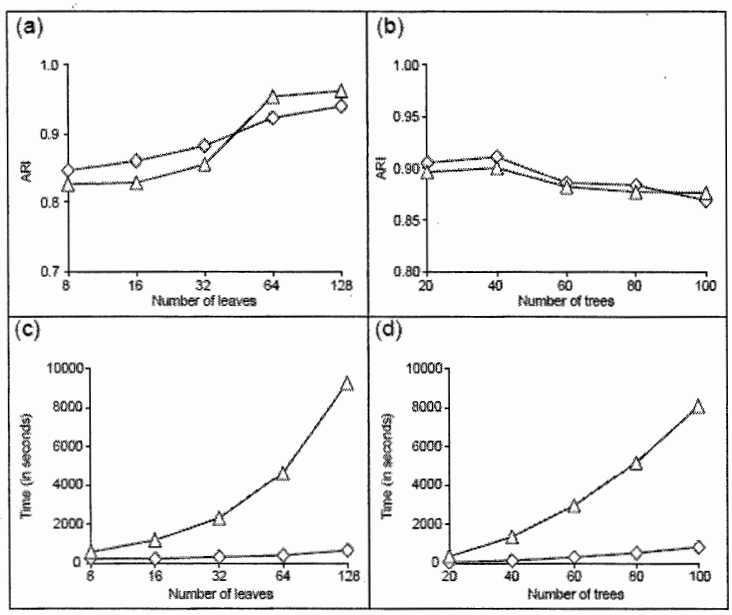


Figure 5 Comparison of our algorithm ($\diamond$) based on the k-medoids clustering, the non-squared RF distance and the SH cluster validity index to the traditional approach ($\triangle$) based on the k-means clustering, on the squared RF distance and on the recomputing the majority consensus trees within k-means (Stockham et al. [17]). The coalescence rate parameter in the HybridSim program was fixed to 5 in this simulation. The comparison was made in terms of ARI (panels a and b) and the running time (measured in seconds) of the methods (panels c and d) with respect to the number of tree leaves and trees.

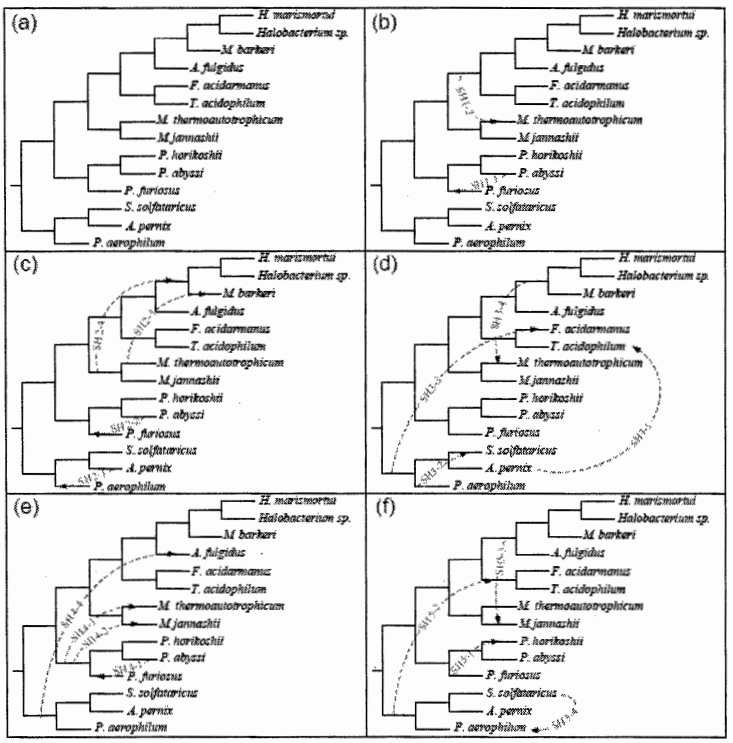


Figure 6 (a) Species tree for the Archaea dataset and five consensus horizontal gene transfer scenarios (panels b to f) obtained for 47 protein trees, originally studied by Matte-Tailliez et al. [4], using the SH cluster validity index and the non-squared RF distance in the k-medoids tree clustering algorithm.

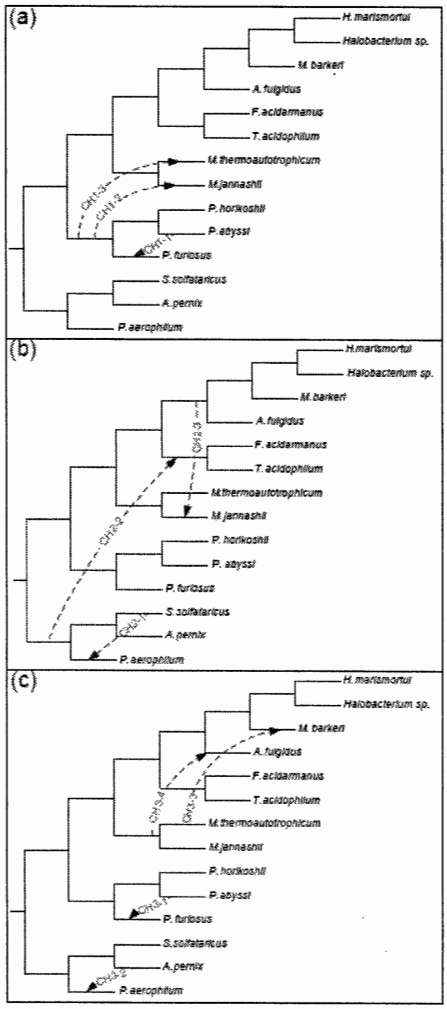


Figure 7 Three consensus horizontal gene transfer scenarios (panels a to c) obtained for 47 protein trees, originally studied by Matte-Tailliez et al. [4], using the CH cluster validity index and the non-squared RF distance in the k-medoids tree clustering algorithm.

Tables

Table 1 Gene transfer statistics for 47 ribosomal protein trees constructed for 14 species of Archaea obtained using the SH and CH cluster validity indices and the non-squared RF distance in the k-medoids tree clustering algorithm. The Crenarchaea group is composed of *S. solfactaricus*, *A. pernix* and *P. aerophilum* species, and the Euryarchaeota group is composed of *P. furiosus*, *P. abyssi*, *P. horikoshii*, *M. jannashii*, *M. thermoautotrophicum*, *T. acidophilum*, *F. acidamanus*, *A. fulgidus*, *M. barkeri*, *Halobacterium sp.* and *H. marismortui* species.

Criterion	Type of gene transfer	Number of transfers detected	Percentage of transfers detected
SH	Intragroup	14	77.78%
	Intergroup	4	22.22%
CH	Intragroup	9	90%
	Intergroup	1	10%

APPENDICE B

EXEMPLES DE CODE SOURCE

«Il y a deux façons de faire la conception d'un logiciel. Une façon est de le rendre si simple qu'il n'y a selon toute apparence aucun défaut. Et l'autre est de le faire si compliqué qu'il n'y a pas de défaut apparent.», Tony Hoare, 1960.

L'appendice B présente quelques exemples de code source développés dans le cadre de ce projet doctoral. Nous organisons cet appendice en six parties. La première partie illustre un exemple d'implémentation de la fonction objective pour le partitionnement d'arbres utilisant la borne inférieure (détaillée au chapitre V dans la section 5.3.2). La deuxième partie contient l'implémentation de la fonction permettant le calcul du critère de validité des clusters de Caliński-Harabasz adapté au cas de partitionnement d'arbres phylogénétiques. La troisième partie contient la procédure de fusion de deux clusters dans le cas d'inférence de super-arbres multiples. La quatrième partie présente le code de calcul de l'*ARI*. La cinquième partie contient le code pour la vérification de la validité des clusters, le calcul de l'indice de Caliński-Harabasz (*CH*) et la fonction objective pour le cas des super-arbres multiples. Finalement, la sixième partie inclut le code pour le calcul de l'indice de validité des clusters Silhouette (*SH*).

Le code source au complet est disponible sur le site suivant : https://github.com/TahiriNadia/CKMedoidsTreeClustering.

B.1 Calcul de la fonction objective dans le cas de la borne inférieure

Cette fonction objective a pour rôle de partitionner les n arbres données en K classes. La meilleure distribution des objets en K classes est celle qui minimise la valeur de la fonction objective. Le code suivant, écrit en langage C++, est également disponible dans le répertoire Github[1].

```
1.   double FO_borne_inf(int &n,int &kmax,double** mat,double* Dvec,int* list,int* howmany,double &S
SE,int &kk,vector <string> monTableau)
2.   {
3.       double *clusterK_same = new double [kmax+1];
4.       int *nk_CH = new int [kmax+1];
5.       int cluster_k=0;
6.       double RF = 0.0;
7.       double Dref=0;
8.       int kref=0;
9.
10.      SSE=0;
11.      int k_source = 0;
12.      int new_k = 0;
13.      int old_k = 0;
14.      int nb_cluster_dest = 0;
15.      int nb_cluster_source = 0;
16.      double FO_old = 0.0;
17.      double FO_new = 0.0;
18.      double tmp_calc = 0.0;
19.      double tmp_calc_dest = 0.0;
20.      double tmp_calc_source = 0.0;
21.
22.      //initialisation des variables
23.      for(int k=1;k<=kmax; k++){
24.          nk_CH[k]=0;
25.          clusterK_same[k]=0.0;
26.      }
27.
28.      //Compter le nombre d'Ć©lĆ©ments par cluster
29.      for(int k=1;k<=kmax; k++){
30.          nk_CH[list[k]]++;
31.      }
32.
33.      //compute for each cluster initially, SSW value (intra groupe distance)
34.      //compute SSW
35.      for (int i=1;i<n;i++){
36.          cluster_k=list[i];
```

[1] https://github.com/TahiriNadia/CKMeansTreeClustering

```
37.          for (int j=i+1;j<=n;j++){
38.              if (list[j]==cluster_k){
39.                  RF = mat[i-1][j-1];
40.                  clusterK_same[cluster_k]+=RF;
41.              }
42.          }
43.      }
44.
45.      // Compute thes last objective function (for reference)
46.      for (int k=1;k<=kk;k++){
47.          if(nk_CH[k]>1){
48.              FO_old += (clusterK_same[k]/(nk_CH[k]-1.0));
49.          }
50.      }
51.
52.      // Compute squared distances to group centroids.
53.      // Assign objects to nearest one
54.      for (int i=1;i<=n; i++)          //do 20 i=1,n
55.      {
56.          k_source = list[i];
57.          if(nk_CH[list[i]]>1){
58.              for (int k=1;k<=kk;k++)
59.              {
60.                  // Compute RF distance for each point i
61.                  // and assign the point i to his best cluster
62.                  // Compute RF distance to the centroid k
63.
64.                  //test if point i doesn't originally belong to k
65.                  //k_source!=k: To avoid revising an item that has changed cluster with its orig
inal cluster
66.                  if(list[i]!=k && k_source!=k){
67.                      nb_cluster_dest = nk_CH[k];
68.                      nb_cluster_source = nk_CH[list[i]];
69.
70.                      if(nk_CH[k]>1 && nk_CH[list[i]]>1){
71.                          FO_new = FO_old - (clusterK_same[k]/(nk_CH[k]-1.0)) -
 (clusterK_same[list[i]]/(nk_CH[list[i]]-1.0));
72.                      }else if(nk_CH[list[i]]>1){
73.                          FO_new = FO_old - (clusterK_same[list[i]]/(nk_CH[list[i]]-1.0));
74.                      }else if(nk_CH[k]>1){
75.                          FO_new = FO_old - (clusterK_same[k]/(nk_CH[k]-1.0));
76.                      }else{
77.                          FO_new = FO_old;
78.                      }
79.
80.                      //compute Function objective
81.                      //For destination cluster
82.                      tmp_calc_dest = clusterK_same[k];
83.                      for(int j=1;j<=n; j++){
84.                          if(list[j]==k){
85.                              tmp_calc_dest += mat[i-1][j-1];
86.                          }
87.                      }
88.                      nb_cluster_dest +=1;
89.
90.                      if(nb_cluster_dest>1){
91.                          FO_new = FO_new + (tmp_calc_dest/(nb_cluster_dest-1.0));
92.                      }
93.
94.                      //For source cluster
95.                      tmp_calc_source = clusterK_same[list[i]];
96.                      for(int j=1;j<=n; j++){
97.                          if(list[j]==list[i]){
98.                              tmp_calc_source -= mat[i-1][j-1];
99.                          }
100.                     }
101.                     nb_cluster_source -=1;
```

```
102.
103.
104.                    if(nb_cluster_source>1){
105.                        FO_new = FO_new + (tmp_calc_source/(nb_cluster_source-1.0));
106.                    }
107.
108.                    if(FO_new<=FO_old){
109.                        Dref=FO_new;
110.                        kref=k;
111.
112.                        //update nk_CH[]
113.                        nk_CH[k] = nb_cluster_dest;
114.                        nk_CH[list[i]] = nb_cluster_source;
115.
116.                        //update RF distance intra-groupe of these two clusters was modified
117.                        clusterK_same[list[i]] = tmp_calc_source;
118.                        clusterK_same[k] = tmp_calc_dest;
119.
120.                        //update objective function FO_old
121.                        FO_old = FO_new;
122.
123.                        //update the distribution list of elements
124.                        list[i] = k;
125.
126.                        new_k = k;
127.                        old_k = list[i];
128.                    }else{
129.                        Dref=FO_old;
130.                    }
131.
132.                }
133.
134.            }
135.        }
136.        SSE=SSE+Dref;          //SSE=SSE+Dref
137.        howmany[kref]++;     //howmany(kref)=howmany(kref)+1
138.
139.    }
140.
141.    //Clean memory
142.    delete [] clusterK_same;
143.    delete [] nk_CH;
144.
145.    return Dref;
146.
147. }//  end FO_borne_inf
```

B.2 Fonction de calcul de l'indice de validité des clusters de Caliński-Harabasz adapté au cas des arbres

L'indice de Caliński-Harabasz est un critère considérant à la fois la distance intragroupe et la distance intergroupe. Un bon partitionnement est celui qui minimise

la distance intragroupe et maximise la distance intergroupe. Le code suivant, écrit en C++, est également disponible dans le répertoire Github[1].

```
1.   double DistanceCH(int &n,int &kmax,double** mat,int* list,double** Ww,double FO_new,double fact
eur) {
2.
3.       // Create and initialize variables
4.       double SSB=0.0;
5.       double SSW=0.0;
6.       double dist_all = 0.0;
7.       double distance_total = 0.0;
8.       int cluster_k=0;
9.       int *nk_CH = new int [kmax+1];
10.      int k_cluster = 0;
11.      double RF = 0.0;
12.
13.      // Initialize the count of element
14.      // for each cluster to 0
15.      for(int k=1;k<=kmax; k++){
16.          nk_CH[k]=0;
17.      }
18.
19.      // Compute element of each cluster
20.      for(int k=1;k<=kmax; k++){
21.          nk_CH[list[k]]++;
22.      }
23.
24.      // Compute all cluster not empty
25.      for(int k=1;k<=kmax; k++){
26.          if(nk_CH[k]!=0){
27.              k_cluster++;
28.          }
29.      }
30.
31.      //compute dist_all
32.      // all element of matrice RF
33.      for (int i=1;i<n;i++){
34.          for (int j=i+1;j<=n;j++){
35.              RF = mat[i-1][j-1];
36.              dist_all += RF;
37.          }
38.      }
39.
40.      dist_all = dist_all*facteur;
41.
42.      //compute SSW
43.      SSW = FO_new;
44.
45.      //compute SSB
46.      SSB = dist_all - SSW;
47.
48.      // Compute the criterion CH
49.      if((fabs(SSW)>0.000001) && (k_cluster>1)){
50.          distance_total=(SSB/SSW)*((n-k_cluster)/(k_cluster-1.0));
51.      }
52.      else if(fabs(SSW)<=0.000001  && (k_cluster>1)){
```

[1] https://github.com/TahiriNadia/CKMeansTreeClustering

```
53.          distance_total=10000000.0*SSB*((n-k_cluster)/(k_cluster-1.0));
54.      }
55.      //Clean memory
56.      delete [] nk_CH;
57.      return distance_total;
58.
59.
60.  }//  end of Distances DistanceCH
```

B.3 Fusion de deux clusters dans le cas des super-arbres

L'étape de fusion des clusters fusionne deux clusters les plus proches. À la suite de cette étape, la procédure de partitionnement sera recommencée, mais pour un nombre de clusters décrémentés. Dans le cas des super-arbres, *c.-à-d.*, des arbres phylogénétiques ayant des ensembles de feuilles différents, mais partiellement chevauchants, nous devons également s'assurer que les deux clusters à fusionner soient connexes. Pour mener à bien cette étape, nous avons utilisé la recherche en profondeur (DFS).

```
1.   if(intParam==20){ // Cas of supertree
2.       //Initialisation of variables
3.       int i_fusion = 1;
4.       int j_fusion = kk;
5.       int fusion_OK = 0;
6.
7.       //Loop until each partition is connected
8.       while (fusion_OK==0 && i_fusion<kk){
9.           j_fusion = kk;
10.          while (fusion_OK==0 && j_fusion>i_fusion){
11.
12.              //Build adjacence matrix
13.              buildMatAdjacenceFusion(i_fusion,j_fusion,n,list,n_identique,mat_adjacence);
14.
15.              //visited is initialized to zero
16.              nb_visited=0;
17.              for(int i=0;i<n;i++)
18.                  visited[i]=0;
19.
20.
21.              //connaitre la premiere ligne et la premiere (la premiere cellule)
22.              //qui ne contient pas un 2
23.              aret_de_recherche = 0;
24.              depart_graphe_i = 0;
25.              while(depart_graphe_i<n && aret_de_recherche==0){
26.                  depart_graphe_j = 0;
27.                  while(depart_graphe_j<n && aret_de_recherche==0){
28.                      if(mat_adjacence[depart_graphe_i][depart_graphe_j]!=2){
29.                          aret_de_recherche = 1;
```

```
30.                     }
31.                     depart_graphe_j++;
32.                 }
33.                 depart_graphe_i++;
34.             }
35.
36.             //Depth-first search (Recherche en profondeur)
37.             DFS(depart_graphe_i-1,visited,mat_adjacence,n,nb_visited);
38.             if(nb_visited==(howmany[i_fusion]+howmany[j_fusion])){
39.                 fusion_OK=1;
40.             }
41.             j_fusion--;
42.         }
43.         i_fusion++;
44.     }
45.
46.     if(fusion_OK==1){
47.         i1ref=i_fusion-1;
48.         i2ref=j_fusion+1;
49.     }else{
50.         kk=k2-1;
51.     }
52. }
```

B.4 Fonction du calcul de l'*ARI*

La qualité de nos partitionnements d'arbres a été évaluée en utilisant l'indice Rand ajusté (*ARI*) (Rand, 1971). Les valeurs de l'*ARI* sont dans l'intervalle [-1; 1]. Quand les deux partitionnements comparés sont exactement les mêmes, la valeur correspondante de l'*ARI* est 1. La fonction suivante, implémentée au langage C++, prend en paramètres : les deux partitionnements (*c.-à-d.*, partitionnement observé et le partitionnement prédit par notre algorithme), le nombre de clusters, ainsi que la taille de nos données.

```
1.  //compute adjusted rand index
2.  double f_ARI(int Strouve[],int Sref[],const char *K_real,int N)
3.  {
4.      int kReal = atoi(K_real);
5.      int tabCongruence [kReal+1][kReal+1];
6.      int sumLigne [kReal+1];
7.      int sumColonne[kReal+1];
8.
9.      //initialisation du tableau sumLigne à 0
10.     for(int i=0;i<=kReal;i++){
11.         sumLigne[i]=0;
12.     }
13.
14.     //initialisation du tableau sumColonne à 0
15.     for(int i=0;i<=kReal;i++){
16.         sumColonne[i]=0;
```

```
    }

    //initialisation du tableau des congruence à 0
    for(int i=0;i<=kReal;i++){
        for(int j=0;j<=kReal;j++){
            tabCongruence[i][j]=0;
        }
    }


    for(int i=0;i<N;i++){
        tabCongruence[Sref[i]][Strouve[i]]++;
    }

    for(int i=1;i<=kReal;i++){
        for(int j=1;j<=kReal;j++){
            sumLigne[i]+=tabCongruence[i][j];
            sumColonne[j]+=tabCongruence[i][j];
        }
    }

    double a=0.0;
    double b=0.0;
    double c=0.0;
    double d=0.0;

    double comb = 1.0;

    for (int i=N; i>=(N-2+1); i--)
    {
        comb*=i;
    }

    comb/=2;

    for (int i=0; i<N-1; i++){
        for (int j=i+1; j<N; j++){
            if(Sref[i]!=Sref[j]){
                if(Strouve[i]!=Strouve[j]){
                    d++;
                }else{
                    c++;
                }
            }else{
                if(Strouve[i]==Strouve[j]){
                    a++;
                }else{
                    b++;
                }
            }
        }
    }

    double ARI = 0.0;

    if(a*2.0==((b+a)+(c+a))){
        ARI = 1.0;
    }else{
        ARI = a - ((b+a)*(c+a))/comb;
        ARI=ARI/((((b+a)+(c+a))/2.0)-(((b+a)*(c+a))/comb));
    }

    return ARI;
}
```

B.5 La vérification de la validité des clusters, l'indice de Caliński-Harabasz (*CH*) et la fonction objective pour le cas de super-arbres multiples

Dans le cas d'analyse d'arbres phylogénétiques ayant des ensembles de feuilles différents, mais partiellement chevauchants, nous avons quelques étapes de traitement à effectuer. En effet, avant de procéder au déplacement d'un élément d'un cluster source au cluster de destination, nous devons au préalable vérifier que le cluster source soit toujours connexe en absence de l'élément en question. De plus, pour pouvoir ajouter cet élément au cluster de destination, nous devons vérifier que le cluster de destination soit toujours connexe en présence de cet élément.

La fonction objective des *k*-moyennes et l'indice de validité des clusters de Caliński-Harabasz (*CH*) (Caliński et Harabasz, 1974) sont codés en langage C++ permettant la validation de ces deux étapes.

```
1.   double DistanceCH_supertree(int &n,int &kmax,double** mat,int* list,double** Ww,double FO_new,d
ouble facteur, double ** tree_cluster_leaves, double ** n_identique) {
2.
3.       double SSB=0.0;
4.       double SSW=0.0;
5.       double dist_all = 0.0;
6.       double distance_total = 0.0;
7.       int cluster_k=0;
8.       int *nk_CH = new int [kmax+1];
9.
10.      int k_cluster = 0;
11.
12.      for(int k=1;k<=kmax; k++){
13.          nk_CH[k]=0;
14.      }
15.
16.      for(int k=1;k<=kmax; k++){
17.          nk_CH[list[k]]++;
18.      }
19.
20.      for(int k=1;k<=kmax; k++){
21.          if(nk_CH[k]!=0){
22.              k_cluster++;
23.          }
24.      }
25.
26.      //compute dist_all
27.      int Nc = 0;
28.      double dist_by_el = 0.0;
29.      for (int i=1;i<=n;i++){
30.           Nc = 0;
31.           dist_by_el = 0.0;
32.          for (int j=1;j<=n;j++){
33.              if(n_identique[i-1][j-1]>3){
34.                  dist_by_el += mat[i-1][j-1];
35.                  Nc++;
```

```
            }
        }
        if(Nc>=2){
            if(Nc!=n){
                dist_all+=((dist_by_el/(Nc-1))*(n-(Nc-1.0)));
            }else{
                dist_all+=dist_by_el;
            }
        }else{
            dist_all+=n-1;
        }
    }
    if(n!=0){
        dist_all = dist_all/(2.0*n);
    }

    //compute SSW
    SSW = FO_new;

    //compute SSB
    SSB = dist_all-SSW;
    if((fabs(SSW)>0.000001) && (k_cluster>1)){
        distance_total=(SSB/SSW)*((n-k_cluster)/(k_cluster-1.0));
    }
    else if(fabs(SSW)<=0.000001  && (k_cluster>1)){
        distance_total=10000000.0*SSB*((n-k_cluster)/(k_cluster-1.0));
    }

    delete [] nk_CH;
    return distance_total;

}//  end ************************End of Distances DistanceCH_supertree
```

```
double FO_super_tree(int &n,int &kmax,double** mat,double* Dvec,int* list,int* howmany,double &
SSE,int &kk,vector <string> monTableau, double ** n_identique, double ** tree_cluster_leaves)
{
    double *clusterK_same = new double [kmax+1];
    int *nk_CH = new int [kmax+1];
    int cluster_k=0;
    double RF = 0.0;
    double Dref=0;
    int kref=0;     //Integer list(nmax),howmany(kmax),kref
    // Integer ishort(pmax)
    // Compute squared distances to group centroids. Assign objects to nearest one
    SSE=0;      //SSE=0.0

    int k_source = 0;
    int new_k = 0;
    int old_k = 0;
    int nb_cluster_dest = 0;
    int nb_cluster_source = 0;
    double FO_old = 0.0;
    double FO_new = 0.0;
    double tmp_calc_dest = 0.0;
    double tmp_calc_source = 0.0;

    int * p = new int [n+1]; //nbarbres n'ayant pas de feuilles communes < à 3 au même cluster
    int **mat_adjacence = new int *[n];
    //Boucle interne de la recherche du pivot de matrice d'adjecence (celui ne contient pas 2)

    int aret_de_recherche = 0;
    int depart_graphe_i = 0;
    int depart_graphe_j = 0;
    double nb_element_dest_i = 0;
    double RF_element_dest_i = 0;
```

```
31.      int nb_el_dest=0;
32.      int trouve = 0;
33.      int k_cluster = 0;
34.      double tmp_dist_supp = 0;
35.
36.      int * visited = new int[n];//indique si les elements ont ete visites
37.      int nb_visited = 0; //compte le nombre d'élément visite
38.
39.      for(int i=0;i<n;i++){
40.          mat_adjacence[i]=new int [n];
41.      }
42.
43.      //initialisation des variables
44.      for(int k=1;k<=kmax; k++){
45.          nk_CH[k]=0;
46.          p[k] = 0;
47.          clusterK_same[k]=0.0;
48.      }
49.
50.      //Initialisation du tableau tree_cluster_leaves
51.      for(int i=0; i<n; i++){
52.          tree_cluster_leaves[i][0]=i+1.0;
53.          tree_cluster_leaves[i][1]=list[i+1];
54.          tree_cluster_leaves[i][2]=0.0;
55.          tree_cluster_leaves[i][3]=0.0;
56.      }
57.
58.      //Compter le nombre d'élément par cluster
59.      for(int k=1;k<=kmax; k++){
60.          nk_CH[list[k]]++;
61.      }
62.
63.      //mise à jour du nombre d'élément par cluster
64.      for(int k=1;k<=kmax; k++){
65.          if(nk_CH[k]!=0){
66.              k_cluster++;
67.          }
68.          howmany[k]=nk_CH[k];
69.      }
70.
71.      //compute for each cluster initially, SSW value (intra groupe distance)
72.      //compute SSW
73.      for (int i=1;i<=n;i++){
74.          cluster_k=list[i];
75.          for (int j=1;j<=n;j++){
76.              if (list[j]==cluster_k){
77.                  if(n_identique[i-1][j-1]>3){
78.                      RF = mat[i-1][j-1];
79.                      clusterK_same[cluster_k]+=RF;
80.                      tree_cluster_leaves[i-1][2]+=1;
81.                      tree_cluster_leaves[i-1][3]+=RF;
82.                  }else{
83.                      //NB d'arbres i et j ayant un nb de feuilles chevauchants inf à 3
84.                      p[cluster_k]++;
85.                  }
86.              }
87.          }
88.      }
89.
90.      for (int k=1;k<=kk;k++){
91.
92.          for(int nb_p_arbre = 0; nb_p_arbre<n; nb_p_arbre++){
93.              if(tree_cluster_leaves[nb_p_arbre][1]==double(k)){
94.                  if(tree_cluster_leaves[nb_p_arbre][2]>=2){
95.                      if(nk_CH[k]!=tree_cluster_leaves[nb_p_arbre][2]){
96.                          tree_cluster_leaves[nb_p_arbre][3]+=((tree_cluster_leaves[nb_p_arbre][3
]/(tree_cluster_leaves[nb_p_arbre][2]-1))*(nk_CH[k]-(tree_cluster_leaves[nb_p_arbre][2]-1)));
```

```
97.                         clusterK_same[k]+=tree_cluster_leaves[nb_p_arbre][3];
98.                     }
99.                 }
100.            }
101.        }
102.        if(nk_CH[k]>1){
103.            FO_old += (clusterK_same[k]/(2.0*nk_CH[k]));
104.        }
105.    }
106.
107.    Dref=FO_old;
108.
109.
110.    for (int i=1;i<=n; i++){
111.
112.        //Pour le cluster source
113.        k_source = list[i];
114.        nb_cluster_source = nk_CH[k_source];
115.        nb_cluster_source -=1;
116.
117.        //on considere que l'element est dans un autre groupe (fictif son ancien cluster+100)
118.        list[i]=list[i]+100;
119.
120.        //Inferer la matrice d'adjacence du cluster k_source
121.        buildMatAdjecence(k_source,n,list,n_identique,mat_adjacence);
122.
123.        //visited is initialized to zero
124.        for(int ii=0;ii<n;ii++)
125.            visited[ii]=0;
126.
127.        //Va permettre de compter le nombre de point connecté
128.        nb_visited=0;
129.
130.        aret_de_recherche = 0;
131.        depart_graphe_i = 0;
132.        while(depart_graphe_i<n && aret_de_recherche==0){
133.            depart_graphe_j = 0;
134.            while(depart_graphe_j<n && aret_de_recherche==0){
135.                if(mat_adjacence[depart_graphe_i][depart_graphe_j]!=2){
136.                    aret_de_recherche = 1;
137.                }
138.                depart_graphe_j++;
139.            }
140.            depart_graphe_i++;
141.        }
142.
143.        //Regarder si le graphe du cluster k_source est connexe sans l'element i
144.        DFS(depart_graphe_i-1,visited,mat_adjacence,n,nb_visited);
145.        list[i]=list[i]-100;
146.        //Si le cluster source est tjs connexe en l'abs de l'arbre i
147.        if(nb_cluster_source==nb_visited){
148.            if(nk_CH[list[i]]>1){
149.                for (int k=1;k<=kk;k++){
150.
151.                    //Calcul de la distance RF de chaque point i
152.                    // et assignation du point i au bon cluster
153.                    // Compute a RF distance to the centroid k
154.
155.                    //test si le point i n'appartenait pas initiallement à k
156.                    //k_source!=k pour éviter de revifier un élément qui a changé de cluster av
ec son cluster d'origine
157.                    if(list[i]!=k && k_source!=k && old_k!=k){
158.
159.                        nb_cluster_dest = nk_CH[k];
160.                        nb_el_dest=0;
161.                        trouve = 0;
162.                        while(nb_el_dest<n && trouve==0){
```

```
163.                            if(list[nb_el_dest+1]==k){
164.                                if(n_identique[nb_el_dest][i-1]>3){
165.                                    trouve=1;
166.                                }
167.                            }
168.                            nb_el_dest++;
169.                        }
170.                        //si l'element du cluster source a au moins un lien avec
171.                        //les élements du cluster de destination
172.                        if(trouve!=0){
173.
174.                            //retirer les anciens calculs de la FO_new
175.                            if(nk_CH[k]>=1 && nk_CH[list[i]]>=1){
176.                                FO_new = FO_old - (clusterK_same[k]/(2.0*nk_CH[k])) -
 (clusterK_same[list[i]]/(2.0*nk_CH[list[i]]));
177.                            }else if(nk_CH[list[i]]>=1){
178.                                FO_new = FO_old -
 (clusterK_same[list[i]]/(2.0*nk_CH[list[i]]));
179.                            }else if(nk_CH[k]>=1){
180.                                FO_new = FO_old - (clusterK_same[k]/(2.0*nk_CH[k]));
181.                            }else{
182.                                FO_new = FO_old;
183.                            }
184.
185.                            //compute la distance intra
186.                            //Pour le cluster de destination
187.                            nb_element_dest_i = 0;
188.                            RF_element_dest_i = 0;
189.                            tmp_calc_dest = 0;
190.                            p[k]=0;
191.
192.                            //calcul de la nouvelle distance RF de l'element i
193.                            //avec les autres elements du cluster k
194.                            for (int i_dest=1;i_dest<=n;i_dest++){
195.                                if (list[i_dest]==k){
196.                                    if(n_identique[i-1][i_dest-1]>3){
197.                                        RF_element_dest_i+=(2.0*mat[i-1][i_dest-1]);
198.                                        nb_element_dest_i++;
199.                                    }else{
200.                                        //nb d'arbres i et j ayant nb feuilles chevauchants< 3
201.                                        p[k]++;
202.                                    }
203.                                }
204.                            }
205.
206.                            nb_cluster_dest +=1;
207.                            if((nb_element_dest_i+1)>=2){
208.                                if((nb_element_dest_i+1)!=nb_cluster_dest){
209.                                    RF_element_dest_i+=((RF_element_dest_i/(nb_element_dest_i))
*(nb_cluster_dest-(nb_element_dest_i)));
210.                                    tmp_calc_dest+=(RF_element_dest_i);
211.                                    tmp_calc_dest+=clusterK_same[k];
212.                                }else{
213.                                    tmp_calc_dest += (RF_element_dest_i);
214.                                    tmp_calc_dest += clusterK_same[k];
215.                                }
216.                            }
217.                            tmp_calc_dest = arrondir(tmp_calc_dest,3);
218.                            if(nb_cluster_dest>0){
219.                                FO_new = FO_new + (tmp_calc_dest/(2.0*nb_cluster_dest));
220.                            }
221.                            //compute la distance intra
222.                            //Pour le cluster source
223.
224.                            tmp_calc_source = clusterK_same[k_source];
225.                            for(int j=1;j<=n; j++){
226.                                if(list[j]==k_source){
```

```
227.                                    if(n_identique[i-1][j-1]>3){
228.                                        tmp_calc_source -= (2.0*mat[i-1][j-1]);
229.                                    }
230.                                }
231.                            }
232.                            tmp_calc_source = arrondir(tmp_calc_source,3);
233.
234.                            if(nb_cluster_source>0){
235.                                FO_new = FO_new + (tmp_calc_source/(2.0*nb_cluster_source));
236.                            }
237.
238.                            if(FO_new<FO_old){
239.                                Dref=FO_new;
240.                                kref=k;
241.                                k_source = k;
242.
243.                                //mise à jour de nk_CH[]
244.                                nk_CH[k] = nb_cluster_dest;
245.                                nk_CH[list[i]] = nb_cluster_source;
246.
247.                                howmany[k] = nb_cluster_dest;
248.                                howmany[list[i]] = nb_cluster_source;
249.
250.                                //mise à jour de la distance intra de l'élément i
251.                                tree_cluster_leaves[i-1][1] = k;
252.                                tree_cluster_leaves[i-1][2] = nb_element_dest_i;
253.                                tree_cluster_leaves[i-1][3]= RF_element_dest_i;
254.
255.                                //mise à jour distance intragroupe des deux clusters modifiés
256.                                clusterK_same[list[i]] = tmp_calc_source;
257.                                clusterK_same[k] = tmp_calc_dest;
258.                                //mise à jour de la fonction objective FO_old
259.                                FO_old = FO_new;
260.                                //mise à jour la liste de distribution des éléments
261.                                old_k = list[i];
262.                                list[i] = k;
263.                                new_k = k;
264.                            }
265.                        }
266.                    }
267.
268.                }
269.            }
270.        }
271.        SSE=SSE+Dref;
272.
273.    }
274.
275.    delete [] clusterK_same;
276.    delete [] nk_CH;
277.    delete [] p;
278.    delete [] visited;
279.
280.    for (int i=0;i<n;i++)
281.    {
282.        delete [] mat_adjacence[i];
283.    }
284.
285.    delete [] mat_adjacence;
286.
287.    return Dref;
288.
289. }//  end FO_super_tree
```

B.6 L'indice de validité des clusters Silhouette (*SH*)

L'indice Silhouette (Rousseew, 1987) est un indice de classification populaire permettant d'évaluer la qualité d'un partitionnement d'objets (ou d'arbres) donné. Le code ci-dessous a été écrit en langage C++.

```
1.   double DistanceSilhouette(int &n,int &nmax,int &p,int &pmax,int &kmax,double** mat,double** xba
r, int* howmany, int &kk,int* list,double* weight,int* ishort) {
2.
3.       //--Note: we suppose that we have a
4.
5.         double Dref=0,D1=0;
6.         int kref=0;
7.         double* ai =new double[n+1]; //Distance of i with all elements in the same cluster than i
8.         double* bi =new double[n+1];
9.         double* si =new double[n+1];
10.        double* sk =new double[kk+1];
11.        double* dist_cluster=new double[kmax+1]; //--distance of i to other cluster (tmp)
12.        int cluster_i=0;
13.        double distance_i=0.0;
14.        int cluster_j=0;
15.        int nk = 0;
16.
17.        int *nk_silhouette = new int [kmax+1];
18.
19.      for(int k=1;k<=kmax; k++){
20.         nk_silhouette[k]=0;
21.      }
22.
23.      //Compute the number of element for each cluster and
24.      //stocke them on nk_silhouette[]
25.      //nk_silhouette[] this array start at 1
26.      for(int k=1;k<=kmax; k++){
27.         nk_silhouette[list[k]]++;
28.      }
29.
30.        for (int k=1; k<=kk;k++){
31.          sk[k]=0.0;
32.        }
33.
34.        for (int i=1;i<=n; i++)
35.        {
36.            //--Calculate a[i] -> calculate the
37.            //--Calculate the distance of i to all other point in its cluster
38.            cluster_i=list[i];
39.            distance_i=0.0;
40.
41.            for (int j=1;j<=n;j++){
42.                if (list[j]==cluster_i) {
43.                    distance_i+=mat[i-1][j-1];
44.                }
45.            }
46.
47.            nk = nk_silhouette[cluster_i];
48.             if(nk>1){
49.                 ai[i]=(distance_i/(nk-1.0));
50.             }else{
51.                 ai[i]=0.0; //--Alone element in cluster
52.             }
53.
54.             //--Calculate b[i] distance of i to the mean distance of each other cluster
55.            bi[i]=100000000.0;
```

```
56.
57.            for (int k=1;k<=kk;k++){
58.              dist_cluster[k]=0.0;
59.            }
60.
61.            for (int j=1;j<=n;j++) {
62.                cluster_j=list[j];
63.
64.                if (cluster_j!=cluster_i) {
65.                    dist_cluster[cluster_j]+=mat[i-1][j-1];
66.                }
67.            }
68.
69.            //--Mean distance
70.             for (int k=1;k<=kk;k++) {
71.                //printf("%lf ",dist_cluster[k]);
72.                if (nk_silhouette[k]>0){
73.                  dist_cluster[k]=dist_cluster[k]/(nk_silhouette[k]);
74.                }
75.            }
76.
77.
78.          //--b[i] is the minimum
79.          for (int k=1;k<=kk;k++){
80.              if (k!=cluster_i && dist_cluster[k]<bi[i]){
81.                  bi[i]=dist_cluster[k];
82.              }
83.          }
84.
85.          if(ai[i]==bi[i]){
86.              si[i]=0.0;
87.          }else if(ai[i]<bi[i]){
88.              if(bi[i]!=0){
89.                  si[i]=1.0-(ai[i]/bi[i]);
90.              }
91.          }else if(ai[i]>bi[i]){
92.              if(ai[i]!=0){
93.                  si[i]=(bi[i]/ai[i])-1.0;
94.              }
95.          }
96.
97.          sk[cluster_i]+=si[i];
98.
99.        }
100.
101.       //--Average the sk (note: nk_silhouette can be 0)
102.       for (int k=1;k<=kk;k++){
103.         if(nk_silhouette[k]>0){
104.             sk[k]=sk[k]/nk_silhouette[k];
105.         }
106.       }
107.
108.        double C=0.0;
109.         for (int k= 1;k <= kk; k++){
110.             C+=sk[k];
111.         }
112.         if(kk!=0){
113.             C = C/(kk*1.0);
114.         }
115.
116.       //--Clean up
117.       delete [] si;
118.       delete [] ai;
119.       delete [] bi;
120.       delete [] sk;
121.       delete [] dist_cluster;
122.       delete [] nk_silhouette;
```

```
123.
124.       return C;
125.
126.
127. }//  end ************************End of Distances
128.
129. double FO_silhouette(int &n,int &kmax,double** mat,double* Dvec,int* list,int* howmany,double &
SSE,int &kk,vector<int> ¢roid_k_pos)
130. {
131.        //--Note: we suppose that we have a
132.
133.       double D1=0;
134.       int kref=0;
135.       double* dist_cluster=new double[kmax+1]; //--distance of i to other cluster (tmp)
136.       double min_distance = 10000000000.0;
137.
138.       for (int i=1;i<n; i++)
139.       {
140.           for (int j=1;j<=n;j++) {
141.               dist_cluster[list[j]]+=mat[i-1][j-1];
142.           }
143.
144.           for (int k=1;k<=kk;k++) {
145.             if(howmany[k]>0){
146.                 dist_cluster[k] = dist_cluster[k]/howmany[k];
147.             }
148.           }
149.
150.           //trouver le min des distance du point i pour l'affecter
151.           min_distance = 10000000000.0;
152.
153.           for (int k=1;k<=kk;k++) {
154.             if(min_distance<dist_cluster[k]){
155.                 min_distance = dist_cluster[k];
156.                 list[i] = k;
157.             }
158.           }
159.
160.
161.       }
162.
163.       //--Clean up
164.       delete [] dist_cluster;
165.
166.       return D1;
167.
168. }//  end ************************End of FO_silhouette
```

APPENDICE C

LISTE DES LANGUES INDO-EUROPÉENNES

«*S'il y avait des synonymes parfaits, il y aurait deux langues dans une même langue.*», César Chesneau, sieur Dumarsais, 1730.

C.1 Liste de Swadesh

Nous présentons dans cet appendice l'ensemble des langues Indo-Européennes de la liste de Swadesh (voir le glossaire pour une définition plus précise). Cette liste est composée de 95 langues. Les données originales se trouvent sur le site web suivant : http://www.trex.uqam.ca/bioling_interactive/iedata.txt ou encore sur le site Github suivant https://github.com/TahiriNadia/dataset/tree/master/data_linguistiques. Pour notre étude impliquant l'analyse des langues Indo-Européennes, nous avons utilisé les données de Dyen *et al.* (Dyen, Kruskal et Black, 1992), modifiées dans le laboratoire du Professeur Vladimir Makarenkov à l'Université du Québec à Montréal, voir aussi le site web suivant : http://www.trex.uqam.ca/bioling_interactive/.

Voici les langues incluses dans notre étude :

TABLE OF LISTS

Classification	L#	List Name	O#	Original Name	Meanings
1.1.1.	01	Irish A	42	EIRE	199
1.1.2.	02	Irish B	43	IRISH	194
1.2.1.1.	03	Welsh N	41	WELSH N	199
1.2.1.2.	04	Welsh C	40	WELSH C	200
1.2.2.1.	05	Breton List	39	BRETON	199
1.2.2.2.	06	Breton SE	38	BRETON SE	198
1.2.2.3.	07	Breton ST	37	BRETON ST	198
2.1.1.1.	08	Rumanian List	53	RUMANIAN	200
2.1.1.2.	09	Vlach	92	VLACH	183
2.1.2.1.	10	Italian	50	ITALIAN	199
2.1.2.2.	11	Ladin	52	LADIN	198
2.1.2.3.1.	12	Provencal	47	PROVENCAL	200
2.1.2.3.2.	13	French	44	FRENCH	200
2.1.2.3.3.	14	Walloon	46	WALLOON	200
2.1.2.4.1.	15	French Creole C	87	CREOLE D	200
2.1.2.4.2.	16	French Creole D	45	CREOLE DF	196
2.1.2.5.	17	Sardinian N	89	SARDINIAN N	198
2.1.2.6.1.	18	Sardinian L	90	SARDINIAN L	200
2.1.2.6.2.	19	Sardinian C	51	SARDINIAN C	200
2.1.2.7.1.	20	Spanish	48	SPANISH	200
2.1.2.7.2.1.	21	Portuguese ST	06	PORTUGUESE	200
2.1.2.7.2.2.	22	Brazilian	05	BRAZILIAN	197
2.1.2.8.	23	Catalan	49	CATALAN	197
2.2.1.1.1.	24	German ST	36	GERMAN	199
2.2.1.1.2.	25	Penn. Dutch	28	DUTCH P	191
2.2.1.1.3.1.	26	Dutch List	27	DUTCH	199
2.2.1.1.3.2.	27	Afrikaans	26	AFRIKAANS	200
2.2.1.1.3.3.	28	Flemish	29	FLEMISH	197
2.2.1.1.4.	29	Frisian	30	FRISIAN	191
2.2.1.2.1.1.1.	30	Swedish Up	94	SWEDISH UP	200
2.2.1.2.1.1.2.	31	Swedish VL	95	SWEDISH VL	199
2.2.1.2.1.1.3.	32	Swedish List	33	SWEDISH	199
2.2.1.2.1.2.	33	Danish	32	DANISH	200
2.2.1.2.1.3.	34	Riksmal	34	RIKSMAL NORWAY	198
2.2.1.2.2.1.	35	Icelandic ST	35	MOD ICELANDIC	195
2.2.1.2.2.2.	36	Faroese	31	FAROESE	197
2.2.2.1.	37	English ST	01	NEW ENGLISH	200
2.2.2.2.	38	Takitaki	02	TAKITAKI	193
2.3.1.1.1.	39	Lithuanian O	67	LITHUANIAN X	199
2.3.1.1.2.	40	Lithuanian ST	66	LITHUANIAN	200
2.3.1.2.	41	Latvian	68	LATVIAN	199
2.3.2.1.	42	Slovenian	84	SLOVENIAN	199
2.3.2.2.1.1.	43	Lusatian L	73	LUSATIAN LP	192
2.3.2.2.1.2.	44	Lusatian U	74	LUSATIAN UP	192
2.3.2.2.2.1.1.	45	Czech	71	CZECH	199
2.3.2.2.2.1.2.	46	Slovak	82	SLOVAK	200
2.3.2.2.2.2.	47	Czech E	10	EAST CZECH	199
2.3.2.2.3.1.	48	Ukrainian	08	UKRAINIAN	200
2.3.2.2.3.2.	49	Byelorussian	09	BYELORUSSIAN	199
2.3.2.2.4.	50	Polish	76	POLISH	200
2.3.2.2.5.	51	Russian	78	RUSSIAN	200
2.3.2.2.6.1.	52	Macedonian	20	MACEDONIAN	192
2.3.2.2.6.2.	53	Bulgarian	07	BULGARIAN	200
2.3.2.2.6.3.	54	Serbocroatian	80	SERBOCROATIAN	200
3.1.	55	Gypsy Gk	93	GYPSY GK	183
3.2.	56	Singhalese	60	SINGHALESE	200

```
64. 3.3.1.           57 Kashmiri       59 KASHMIRI       195
65. 3.3.2.1.1.1.     58 Marathi        16 MARATHI        200
66. 3.3.2.1.1.2.     59 Gujarati       19 GUJARATI       194
67. 3.3.2.1.2.1.1.   60 Panjabi ST     13 PANJABI        199
68. 3.3.2.1.2.1.2.   61 Lahnda         57 LAHNDA         199
69. 3.3.2.1.2.2.     62 Hindi          14 HINDI          200
70. 3.3.2.1.3.       63 Bengali        15 BENGALI        198
71. 3.3.2.2.1.       64 Nepali List    58 NEPALI         199
72. 3.3.2.2.2.       65 Khaskura       12 KHASKURA       187
73. 4.1.1.           66 Greek ML       62 GREEK ML       199
74. 4.1.2.           67 Greek MD       64 GREEK MD       195
75. 4.1.3.           68 Greek Mod      61 GREEK MOD      200
76. 4.1.4.           69 Greek D        65 GREEK D        198
77. 4.2.             70 Greek K        63 GREEK K        198
78. 5.1.             71 Armenian Mod   11 MOD ARMENIAN   196
79. 5.2.             72 Armenian List  04 ARMENIAN       192
80. 6.1.             73 Ossetic        88 OSSETIC        184
81. 6.2.1.1.         74 Afghan         54 AFGHAN         200
82. 6.2.1.2.         75 Waziri         03 WAZIRI         192
83. 6.2.2.1.1.       76 Persian List   17 PERSIAN        197
84. 6.2.2.1.2.       77 Tadzik         21 TADZIK         196
85. 6.2.2.2.         78 Baluchi        55 BALOCHI        195
86. 6.2.2.3.         79 Wakhi          56 WAKHI          177
87. 7.1.1.1.         80 Albanian T     25 ALBANIAN T     198
88. 7.1.1.2.         81 Albanian Top   91 ALBANIAN TOP   200
89. 7.1.2.           82 Albanian G     22 ALBANIAN SG    190
90. 7.2.             83 Albanian K     24 ALBANIAN K     200
91. 7.3.             84 Albanian C     23 ALBANIAN C     190
92. ---------------- -- ------------- -- ------------- --------
```

C.2 Arbre d'évolution des langues Indo-Européennes

La figure C.1 montre l'arbre d'évolution des langues Indo-Européennes, reproduit depuis l'article de Gray et Atkinson de 2003. Cet article de référence en linguistique montre la disposition des langues Indo-Européennes (IE) dans un arbre tout en supportant l'hypothèse Anatolienne pour l'origine des langues IE (Gray et Atkinson, 2003).

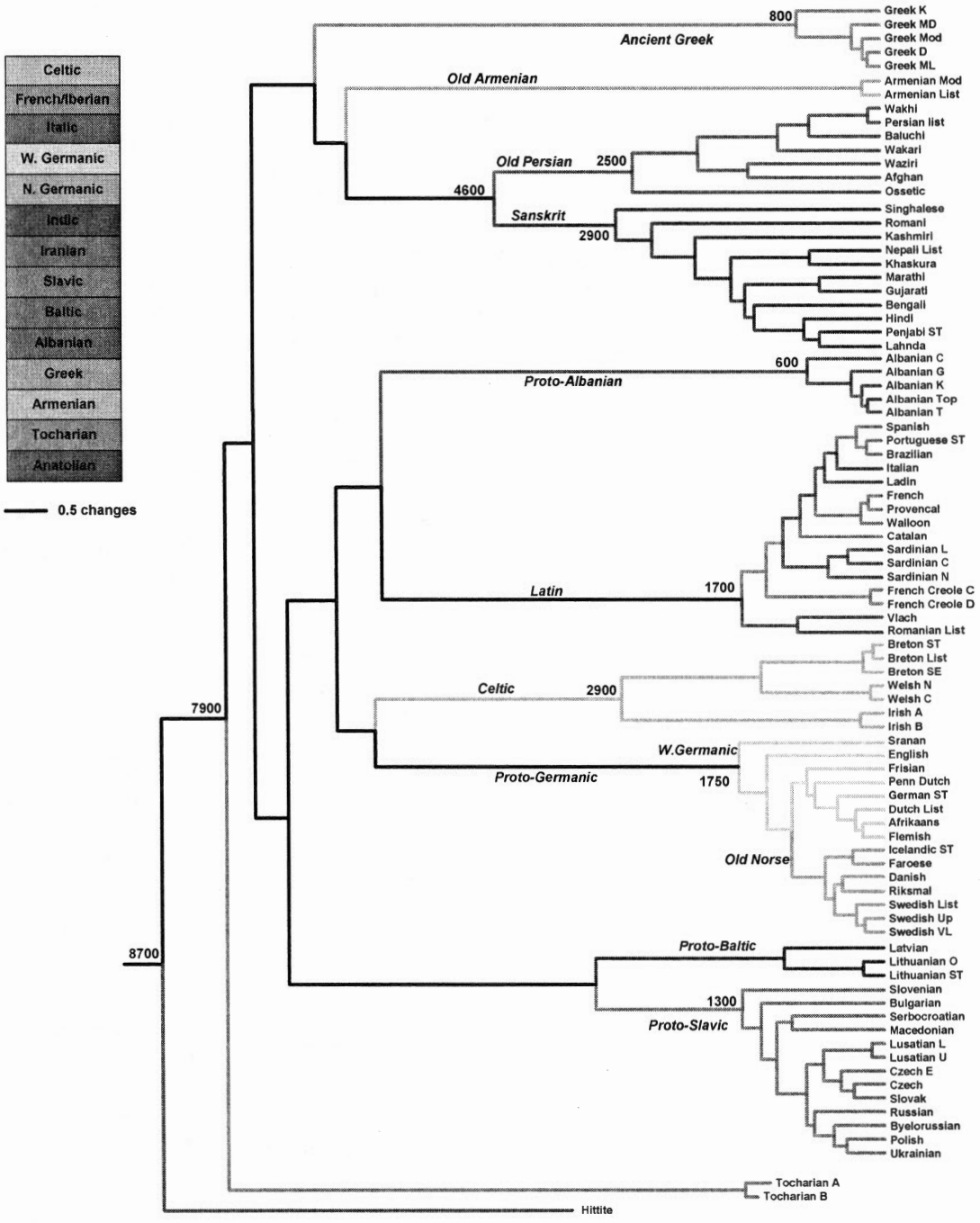


Figure C.1 Arbre de référence pour les langues Indo-Européennes (reproduit d'après Gray et Atkinson, 2003).

C.3 Composition des différents groupes des langues Indo-Européennes

Les langues Indo-Européennes sont classifiées en 14 groupes organisés comme illustré au tableau C.1.

Tableau C.1 Résumé de la composition des groupes des langues Indo-Européennes (Gray et Atkinson, 2003).

Nom du groupe	Nombre de langues compris dans le groupe
Celtic	7
French/Iberian	9
Italic	7
West Germanic	8
North Germanic	7
Indic	11
Iranian	7
Slavic	13
Baltic	3
Albanian	5
Greek	5
Armenian	2
Tocharian	2
Anatolian	1

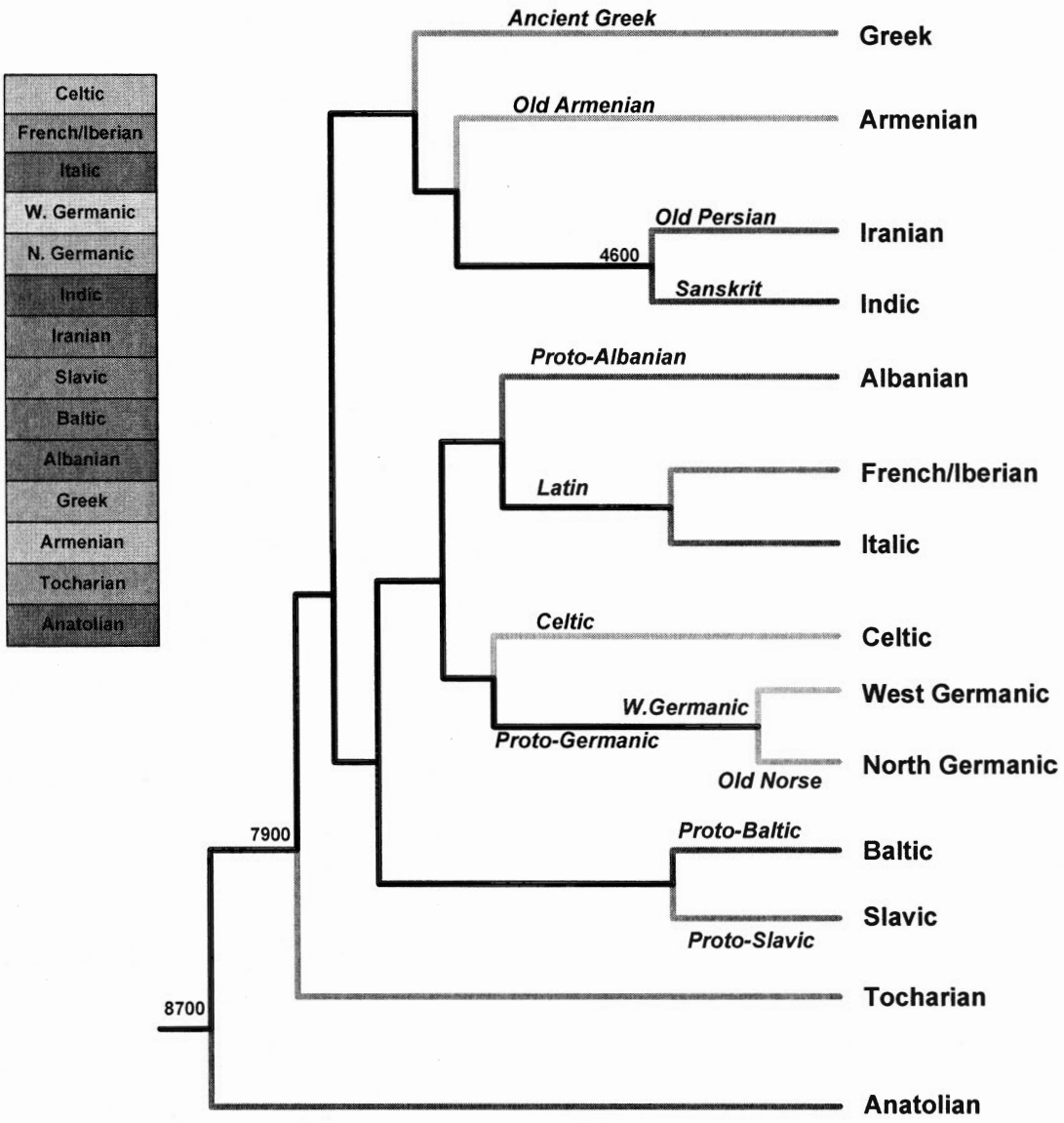


Figure C.2 Arbre de référence pour les groupes des langues Indo-Européennes abrégé.

APPENDICE D

PRÉTRAITEMENT DES DONNÉES

«L'un des axiomes de Sherlock est le suivant : C'est une grossière erreur que d'émettre des hypothèses avant d'avoir des données... car on a tendance à déformer les faits pour étayer les hypothèses, au lieu que les hypothèses viennent étayer les faits.», Harlan Coben, 2007.

L'appendice D mettra en avant le protocole de prétraitement des données des langues Indo-Européennes du stade du téléchargement jusqu'à leur utilisation. Le protocole comporte trois principales étapes : 1) le téléchargement des données, 2) le classement des arbres linguistiques selon leur groupe d'appartenance et) le nettoyage des arbres linguistiques afin de ne retenir que les langues appartenant au groupe en question. Le code source présenté est également disponible à l'adresse URL suivante : https://github.com/TahiriNadia/traitement_data_linguistic.

D.1 Téléchargements des arbres linguistiques

Pour mener à bien cette tâche d'analyse des données des langues Indo-Européennes, nous avons téléchargé les arbres linguistiques depuis le site de T-Rex[1]. Le script a été

---

[1] http://trex.uqam.ca/bioling_interactive/

écrit en Python version 3.2 dans lequel nous avons utilisé la librairie *BeautifulSoup*[1]. Cette librairie permet d'extraire les données des fichiers HTML et XML.

```
import urllib2
import httplib2
from BeautifulSoup import BeautifulSoup, SoupStrainer
import re

def rreplace(s, old, new, occurrence):
    li = s.rsplit(old, occurrence)
    return new.join(li)

#== Liste de toutes les langues Indo-European
all_languages = ["","IrishA","IrishB","WelshN","WelshC","BretonL","BretonSE","BretonST","Romanian","Vlach","Italian","Ladin","Provencal","French","Walloon","CreoleC","CreoleD","SardiniaN","SardiniaL","SardiniaC","Spanish","Portugues","Brazilian","Catalan","GermanST","PennDutch","Dutch","Afrikaans","Flemish","Frisian","SwedishUp","SwedishVL","Swedish","Danish","Riksmal","Icelandic","Faroese","English","Sranan","LithuaO","LithuaST","Latvian","Slovenian","LusatiaL","LusatiaU","Czech","Slovak","CzechE","Ukrainian","Byelo","Polish","Russian","Macedonia","Bulgarian","Serbo","Romani","Singhales","Kashmiri","Marathi","Gujarati","Panjabi","Lahnda","Hindi","Bengali","Nepali","Khaskura","GreekML","GreekMD","GreekMod","GreekD","GreekK","ArmenianM","ArmenianL","Ossetic","Afghan","Waziri","Persian","Tadzik","Baluchi","Wakhi","AlbanT","AlbanTop","AlbanG","AlbanK","ALbanC"]

nb_languages = []

for x in range(0, len(all_languages)):
    nb_languages.append(0)

response = urllib2.urlopen('http://www.trex.uqam.ca/bioling_interactive/')
html = response.read()

links = re.findall("a href=\w+\.\w+\?\w+=\w+",html)

for link in links:
    if(link != "LINK FOR THE ORIGINAL INDO-EUROPEAN LANGUAGE DATABASE COLLECTED BY ISIDORE DYEN"):

        response = urllib2.urlopen('http://www.trex.uqam.ca/bioling_interactive/'+link.split("a href=")[1])
        html = response.read()
        linkWords = re.findall("cognat\.php\?\w+=\w+\&\w+=\d+",html)
        for linkWord in linkWords:
            response = urllib2.urlopen('http://www.trex.uqam.ca/bioling_interactive/'+linkWord)
            html = response.read()
            stringNewick = re.findall("<input type='hidden' name='chaineNewick' value='[\(+\w+\-+\)*\:+\.+\,+]+;",html)
            str_newick = stringNewick[1].split("<input type='hidden' name='chaineNewick' value='")[1]

            str_newick = re.sub(r"[A-Za-z\.\*]+\-"," ",str_newick)
            str_newick = re.sub(r"\-\d+","",str_newick)

            for x in range(1, len(all_languages)):
                if x<=9:
                    str_newick = re.sub(" 0"+str(x)+":"," "+all_languages[x]+":",str_newick)
                else:
                    str_newick = re.sub(" "+str(x)+":"," "+all_languages[x]+":",str_newick)

                tmp = re.findall(" "+all_languages[x]+":",str_newick)

                if len(tmp)>1:
                    i = 1
                    str_newick = rreplace(str_newick, " "+all_languages[x]+":"," AAA"+str(i)+":", str_newick.count(" "+all_languages[x]) - 1)
                    tmp = re.findall(" AAA"+str(i)+":",str_newick)

                    while len(tmp)>1:
                        str_newick = rreplace(str_newick, "AAA"+str(i)+":", "AAA"+str(i+1)+":", str_newick.count(" AAA"+str(i)+":") - 1)
                        tmp = re.findall(" AAA"+str(i+1)+":",str_newick)
                        i = i + 1
            str_newick = re.sub(" ","",str_newick)
            print (str_newick)
```

[1] https://www.crummy.com/software/BeautifulSoup/bs4/doc/

```
59.  print("\n")
60.
```

D.2 Classement des arbres par groupes

Une fois le téléchargement effectué, nous avons trié les données selon 14 groupes, à savoir Celtic, Italic, French, West Germanic, North Germanic, Slavic, Iranian, Albanian, Greek, Latin, Latin avec Germanic, Latin avec West Germanic, French avec West Germanic, et North Germanic avec West Germanic. Le script a été écrit en Perl version 5.26.1. La sortie du script est un fichier d'arbres linguistiques appartenant à un groupe (indiqué comme paramètre d'entrée).

```
1.   use strict;
2.   use warnings;
3.
4.   #== En parametres, donner au programme le fichier avec les arbres des langues
5.   #== Exemple >perl biolinguistic_statCluster.pl out_trees_1302.txt
6.   die("Erreur...\nperl $0 filename") if((scalar @ARGV) != 1);
7.
8.
9.   my $fichier = $ARGV[0];
10.
11.  #== Liste des langues par cluster
12.  my @celtic = ("IrishA","IrishB","WelshN","WelshC","BretonL","BretonSE","BretonST");
13.  my @italic = ("Romanian","Vlach","Ladin","Italian","SardinianN","SardiniaC","SardiniaL");
14.  my @french = ("Provencal","French","Walloon","CreoleC","CreoleD","Spanish","Portugues","Brazilian","Catalan");
15.  my @west_germanic = ("GermanST","PennDutch","Dutch","Afrikaans","Flemish","Frisian","English","Sranan");
16.  my @north_germanic = ("SwedishUp","SwedishVL","Swedish","Riksmal","Icelandic","Faroese","Danish");
17.  my @baltic = ("LithuaO","LithuaST","Latvian");
18.  my @slavic = ("Slovenian","Macedonia","Bulgarian","Serbo","LusatiaL","LusatiaU","Czech","CzechE","Slovak","Ukrainian","Byelo","Russian","Polish");
19.  my @indic = ("Romani","Singhales","Marathi","Gujarati","Panjabi","Lahnda","Hindi","Bengali","Nepali","Khaskura","Kashmiri");
20.  my @iranian = ("Ossetic","Wakhi","Persian","Tadzik","Baluchi","Afghan","Waziri");
21.  my @albanian = ("AlbanT","AlbanG","AlbanTop","AlbanK","ALbanC");
22.  my @greek = ("GreekML","GreekMD","GreekMod","GreekD","GreekK");
23.  my @latin = ("Provencal","French","Walloon","CreoleC","CreoleD","Spanish","Portugues","Brazilian","Catalan","SardiniaL","SardiniaN","SardiniaC","Italian","Romanian","Vlach","Ladin");
24.  my @latin_germanic = ("SwedishUp","SwedishVL","Swedish","Riksmal","Icelandic","Faroese","Danish","GermanST","PennDutch","Dutch","Afrikaans","Flemish","Frisian","English","Sranan","Provencal","French","Walloon","CreoleC","CreoleD","Spanish","Portugues","Brazilian","Catalan","SardiniaL","SardiniaN","SardiniaC","Italian","Romanian","Vlach","Ladin");
25.  my @latin_west_germanic = ("GermanST","PennDutch","Dutch","Afrikaans","Flemish","Frisian","English","Sranan","Provencal","French","Walloon","CreoleC","CreoleD","Spanish","Portugues","Brazilian","Catalan","SardiniaL","SardiniaN","SardiniaC","Italian","Romanian","Vlach","Ladin");
26.  my @french_west_germanic = ("Provencal","French","Walloon","CreoleC","CreoleD","Spanish","Portugues","Brazilian","Catalan","GermanST","PennDutch","Dutch","Afrikaans","Flemish","Frisian","English","Sranan");
27.  my @other = ("TocharianA","TocharianB","Hittite","ArmenianM","ArmenianL");
28.  my @north_and_west_germanic = ("SwedishUp","SwedishVL","Swedish","Riksmal","Icelandic","Faroese","Danish","GermanST","PennDutch","Dutch","Afrikaans","Flemish","Frisian","English","Sranan");
29.
30.  #-- Noms des fichiers de sorties
31.  my $file_celtic = "celtic.txt";
32.  my $file_italic = "italic.txt";
33.  my $file_french = "french.txt";
34.  my $file_west_germanic = "west_germanic.txt";
35.  my $file_north_germanic = "north_germanic.txt";
36.  my $file_baltic = "baltic.txt";
37.  my $file_slavic = "slavic.txt";
38.  my $file_indic = "indic.txt";
39.  my $file_iranian = "iranian.txt";
40.  my $file_albanian = "albanian.txt";
41.  my $file_greek = "greek.txt";
42.  my $file_latin = "latin.txt";
43.  my $file_latin_germanic = "latin_germanic.txt";
44.  my $french_west_germanic = "french_west_germanic.txt";
```

```
45.  my $file_latin_west_germanic = "latin_west_germanic.txt";
46.  my $north_and_west_germanic = "north_and_west_germanic.txt";
47.
48.  print "STATISTIC celtic:\n";
49.  &statListeLangue($file_celtic,@celtic);
50.
51.  print "STATISTIC italic:\n";
52.  &statListeLangue($file_italic,@italic);
53.
54.  print "STATISTIC french:\n";
55.  &statListeLangue($file_french,@french);
56.
57.  print "STATISTIC west_germanic:\n";
58.  &statListeLangue($file_west_germanic,@west_germanic);
59.
60.  print "STATISTIC north_germanic:\n";
61.  &statListeLangue($file_north_germanic,@north_germanic);
62.
63.  print "STATISTIC baltic:\n";
64.  &statListeLangue($file_baltic,@baltic);
65.
66.  print "STATISTIC slavic:\n";
67.  &statListeLangue($file_slavic,@slavic);
68.
69.  print "STATISTIC indic:\n";
70.  &statListeLangue($file_indic,@indic);
71.
72.  print "STATISTIC iranian:\n";
73.  &statListeLangue($file_iranian,@iranian);
74.
75.  print "STATISTIC albanian:\n";
76.  &statListeLangue($file_albanian,@albanian);
77.
78.  print "STATISTIC greek:\n";
79.  &statListeLangue($file_greek,@greek);
80.
81.  print "STATISTIC latin:\n";
82.  &statListeLangue($file_latin,@latin);
83.
84.  print "STATISTIC latin_germanic:\n";
85.  &statListeLangue($file_latin_germanic,@latin_germanic);
86.
87.  print "STATISTIC latin_west_germanic:\n";
88.  &statListeLangue($file_latin_west_germanic,@latin_west_germanic);
89.
90.  print "STATISTIC french_west_germanic:\n";
91.  &statListeLangue($french_west_germanic,@french_west_germanic);
92.
93.  print "STATISTIC north_and_west_germanic:\n";
94.  &statListeLangue($north_and_west_germanic,@north_and_west_germanic);
95.
96.  print STDOUT "\n\n\nFin normale du script $0\n\n\n";
97.
98.  #=========================================================
99.  #= Fonction statListeLangue
100. #= input:
101. #= 1) nom de fichier de sortie
102. #= 2) liste des langue d'un cluster
103. #= output:
104. #= 1) fichier avec les arbres avec 4 langues d'un même
105. #= cluster.
106. #=========================================================
107. sub statListeLangue{
108.
109.     my ($name_file,@liste_langues) = @_;
110.
111.     open ( IN , $fichier) or die($!);
112.     open ( OUT, ">".$name_file) or die($!);
113.     my $compteur = 0;
114.     my $i = 0;
115.
116.     while( my $ligne = <IN> ){
117.         chomp($ligne);
118.         $i = 0;
119.         $compteur = 0;
120.         while( $i<scalar(@liste_langues)){
121.             if( $ligne =~ /\($liste_langues[$i]:|,$liste_langues[$i]:/ ){
122.                 $compteur++;
123.             }
124.             $i++;
125.         }
126.
127.         if($compteur>=4){
128.             print OUT "$ligne\n";
```

```
129.            #print "$ligne\n";
130.        }
131.    }
132.    close ( IN );
133.    close ( OUT );
134. }
```

D.3 Nettoyage des données des langues Indo-Européennes

Dans cette section, nous présentons la procédure de nettoyage des données des langues Indo-Européennes. Pour chaque groupe étudié, cette étape consiste à supprimer, dans le string Newick, les langues n'appartenant pas au groupe en question. Cette étape est nécessaire pour enlever les bruits pouvant être occasionnés par les autres langues n'appartenant pas au groupe analysé. Le script a été écrit en Python (version 3.2) en utilisant la librairie *dendropy*[1]. Cette librairie nous permet de supprimer les feuilles d'un arbre phylogénétique donné en entrée.

```
1.  #! /usr/bin/env python
2.
3.  """
4.  Manipulation d'arbres phylogenetiques
5.
6.  Auteur  : Nadia Tahiri
7.  Date    : January 2017
8.  version : 1.0
9.  """
10.
11. __author__ = "Nadia Tahiri"
12. __date__   = "January 2017"
13.
14. import dendropy
15. import sys
16. import re
17.
18. nameFile = sys.argv[1]
19. nameFileWE = nameFile.replace(".txt", "") #name file without extension
20. f_in = open(nameFile, 'r')
21. f_out = open(nameFileWE,'w')
22.
23. #== Liste des langues par cluster
24. celtic = ["IrishA","IrishB","WelshN","WelshC","BretonL","BretonSE","BretonST"]
25. italic = ["Romanian","Vlach","Ladin","Italian","SardinianN","SardiniaC","SardiniaL"]
26. french = ["Provencal","French","Walloon","CreoleC","CreoleD","Spanish","Portugues","Brazilian","Catalan"]
27. west_germanic = ["GermanST","PennDutch","Dutch","Afrikaans","Flemish","Frisian","English","Sranan"]
28. north_germanic = ["SwedishUp","SwedishVL","Swedish","Riksmal","Icelandic","Faroese","Danish"]
29. baltic = ["LithuaO","LithuaST","Latvian"]
30. slavic = ["Slovenian","Macedonia","Bulgarian","Serbo","LusatiaL","LusatiaU","Czech","CzechE","Slovak","Ukrainian",
    "Byelo","Russian","Polish"]
31. indic = ["Romani","Singhales","Marathi","Gujarati","Panjabi","Lahnda","Hindi","Bengali","Nepali","Khaskura","Kashm
    iri"]
32. iranian = ["Ossetic","Wakhi","Persian","Tadzik","Baluchi","Afghan","Waziri"]
33. albanian = ["AlbanT","AlbanG","AlbanTop","AlbanK","ALbanC"]
34. greek = ["GreekML","GreekMD","GreekMod","GreekD","GreekK"]
35. latin = ["Provencal","French","Walloon","CreoleC","CreoleD","Spanish","Portugues","Brazilian","Catalan","SardiniaL
    ","SardinianN","SardiniaC","Italian","Romanian","Vlach","Ladin"]
36. other = ["TocharianA","TocharianB","Hittite","ArmenianM","ArmenianL"]
```

[1] https://www.dendropy.org/

```
37.  latin_germanic = ["SwedishUp","SwedishVL","Swedish","Riksmal","Icelandic","Faroese","Danish","GermanST","PennDutch
     ","Dutch","Afrikaans","Flemish","Frisian","English","Sranan","Provencal","French","Walloon","CreoleC","CreoleD","S
     panish","Portugues","Brazilian","Catalan","SardiniaL","SardiniaN","SardiniaC","Italian","Romanian","Vlach","Ladin"
     ]
38.  latin_west_germanic = ["GermanST","PennDutch","Dutch","Afrikaans","Flemish","Frisian","English","Sranan","Provenca
     l","French","Walloon","CreoleC","CreoleD","Spanish","Portugues","Brazilian","Catalan","SardiniaL","SardiniaN","Sar
     diniaC","Italian","Romanian","Vlach","Ladin"]
39.  french_west_germanic = ["Provencal","French","Walloon","CreoleC","CreoleD","Spanish","Portugues","Brazilian","Cata
     lan","GermanST","PennDutch","Dutch","Afrikaans","Flemish","Frisian","English","Sranan"]
40.  vide_lang = [""]
41.  lang_AAA = ["AAA1","AAA2","AAA3","AAA4","AAA5","AAA6","AAA7","AAA8","AAA9","AAA10","AAA11","AAA12"]
42.  north_and_west_germanic = ["SwedishUp","SwedishVL","Swedish","Riksmal","Icelandic","Faroese","Danish","GermanST","
     PennDutch","Dutch","Afrikaans","Flemish","Frisian","English","Sranan"]
43.
44.
45.  if(sys.argv[2] == "celtic"):
46.      language = celtic
47.  elif(sys.argv[2] == "italic"):
48.      language = italic
49.  elif(sys.argv[2] == "french"):
50.      language = french
51.  elif(sys.argv[2] == "west_germanic"):
52.      language = west_germanic
53.  elif(sys.argv[2] == "north_germanic"):
54.      language = north_germanic
55.  elif(sys.argv[2] == "baltic"):
56.      language = baltic
57.  elif(sys.argv[2] == "slavic"):
58.      language = slavic
59.  elif(sys.argv[2] == "indic"):
60.      language = indic
61.  elif(sys.argv[2] == "iranian"):
62.      language = iranian
63.  elif(sys.argv[2] == "albanian"):
64.      language = albanian
65.  elif(sys.argv[2] == "greek"):
66.      language = greek
67.  elif(sys.argv[2] == "latin"):
68.      language = latin
69.  elif(sys.argv[2] == "other"):
70.      language = other
71.  elif(sys.argv[2] == "latin_germanic"):
72.      language = latin_germanic
73.  elif(sys.argv[2] == "latin_west_germanic"):
74.      language = latin_west_germanic
75.  elif(sys.argv[2] == "french_west_germanic"):
76.      language = french_west_germanic
77.  elif(sys.argv[2] == "north_and_west_germanic"):
78.      language = north_and_west_germanic
79.  else:
80.      language = vide_lang
81.
82.  all_languages = celtic + baltic + slavic + indic + iranian  + albanian + greek + latin_germanic + other + lang_AAA

83.
84.  # all_languages = celtic + italic + french + west_germanic + north_germanic + baltic + slavic + indic + iranian  +
      albanian + greek + latin + other
85.  print ("")
86.  print (all_languages)
87.  print ("")
88.  #remove the language isn't include on the good cluster
89.  for i in range (0,len(language)):
90.      all_languages.remove(language[i])
91.  print ("")
92.  print (all_languages)
93.  print ("")
94.  for ligne in f_in:
95.      ligne.rstrip()
96.      tree_str = "[&R] " + ligne
97.
98.      tree = dendropy.Tree.get(
99.          data=tree_str,
100.         schema="newick")
101.     print("Before:")
102.     print(tree.as_string(schema='newick'))
103.     tree.prune_taxa_with_labels(all_languages)
104.
105.     print("after:")
106.     print(tree.as_string(schema='newick'))
107.     str = tree.as_string(schema='newick')
108.     str = re.sub(r':\d+\.\d+;',';',str)
109.     print(str.replace("[&R] ", ""), file=f_out, end="")
110.
111. f_out.close()
```

```
112. f_in.close()
```

GLOSSAIRE

ADN (acide désoxyribonucléique) : macromolécule constituée de deux chaînes enroulées en double hélice. Ses deux brins sont assemblés à partir de nucléotides. Chaque nucléotide comprend un sucre, le désoxyribose, un phosphate et une des quatre bases azotées (adénine, guanine, cytosine et thymine). L'ADN est le support de l'information génétique des organismes vivants.

Alignement : opération qui consiste à disposer les unes en dessous des autres des portions de séquences similaires en minimisant leurs différences (on peut aligner entre eux des gènes d'une même famille multigénique ou des gènes d'espèces différentes). Si ces gènes sont homologues, les différences d'aminoacides ou d'acides nucléiques entre les séquences actuelles sont le témoignage de mutations qui ont eu lieu dans le passé.

Aminoacide (acide aminé) : unité constitutive des protéines. Il existe 20 acides aminés communs : alanine, arginine, asparagine, aspartate, cystéine, glutamine, glycine, histidine, isoleucine, leucine, lysine, méthionine, phénylalanine, proline, glutamate, sérine, thréonine, tryptophane, tyrosine et valine.

ARN (acide ribonucléique) : polymère linéaire dont la sous-unité de base, un ribonucléotide, contient le sucre ribose.

Clade : vient du grec clados qui signifie arête. Taxon strictement monophylétique, c'est-à-dire contenant un ancêtre et tous ses descendants.

Cognat : les cognats, ou mots apparentés, sont des mots qui ont une origine commune. Le terme peut désigner des mots d'une même langue, ou bien (le plus couramment) des mots dans des langues différentes. Par exemple, les mots nuit (en français), night (en anglais), et nacht (en allemand) sont apparentés, car ils sont issus

d'une même racine Indo-Européenne. De même, les mots père et paternel sont apparentés, car tous deux issus du latin pater.

Extragroupe (outgroup) : on dit aussi groupe extérieur ou encore "outgroup" tiré de l'anglais. Groupe que l'on sait *a priori* placé en dehors d'un ensemble de taxons dont on cherche les relations de parenté.

Horloge moléculaire (hypothèse) : l'hypothèse selon laquelle les molécules d'une même classe fonctionnelle évoluent régulièrement dans le temps à un rythme égal dans différentes lignées. Ainsi la quantité des différences moléculaires constatées de nos jours dans des séquences homologues d'espèces distinctes peut être utilisée pour estimer le temps écoulé depuis le dernier ancêtre commun à ces espèces (ou le temps de divergence).

Liste Swadesh : est une liste de mots de signification de base établie par le linguiste et l'anthropologue Morris Swadesh. Cette liste est très utilisée dans différence champs d'études tels que la linguistique comparée, la linguistique historique et aussi l'anthropologie.

Racine : le segment de l'arête en amont du nœud du rang le plus important, définissant le groupe extérieur (voir l'Extragroupe). En d'autres termes, c'est la position dans l'arbre du groupe extérieur. La racine peut être considérée comme un point de référence pour l'interprétation des caractères : les états de caractères de l'extragroupe (outgroup) sont des états plésiomorphes, les états qui en diffèrent sont apomorphes. Remarque : pour pouvoir comparer aisément deux arbres, il faut les enraciner chacun avec la même espèce ou avec le même taxon.

Taxon : ensemble des organismes reconnus et définis dans chacune des catégories de la classification biologique hiérarchisée. En d'autres termes : contenu concret d'une catégorie. Exemple : *Canis lupus*, le loup, est un taxon de rang spécifique (catégorie :

espèce); les canidés (Chien, Loup, Renard) constituent un taxon de rang familial (catégorie : famille).

BIBLIOGRAPHIE

Abello, J., Kreinovich, V., Nguyen, H. T., Sudarsky, S. et Yen, J., 1994. Computing an appropriate control strategy based only on a given plant's rule-based model is NP-hard. Industrial Fuzzy Control and Intelligent Systems Conference, and the NASA Joint Technology Workshop on Neural Networks and Fuzzy Logic, IEEE, 331-332.

Adams, E. N., 1972. Consensus techniques and the comparison of taxonomic trees. Systematic Biology, 21(4), 390-397.

Abfalg, A. et Erdfelder, E., 2012. CAML—Maximum likelihood consensus analysis. Behavior research methods, 44(1), 189-201.

Arief, V. N., DeLacy, I. H., Basford, K. E. et Dieters, M. J., 2017. Application of a dendrogram seriation algorithm to extract pattern from plant breeding data. Euphytica, 213(4), 85.

Atkinson, Q. D. et Gray, R. D., 2005. Curious parallels and curious connections—phylogenetic thinking in biology and historical linguistics. Systematic biology, 54(4), 513-526.

Aurenhammer, F., 1991. Voronoi diagrams—a survey of a fundamental geometric data structure. ACM Computing Surveys (CSUR), 23(3), 345-405.

Avni, E., Cohen, R. et Snir, S., 2014. Weighted quartets phylogenetics. Systematic biology, 64(2), 233-242.

Ball, G. H. et Hall, D. J., 1967. A clustering technique for summarizing multivariate data. Behavioral science, 12(2), 153-155.

Bandelt, H. J. et Dress, A. W., 1992. Split decomposition: a new and useful approach to phylogenetic analysis of distance data. Molecular phylogenetics and evolution, 1(3), 242-252.

Barthélemy J.-P. et Guénoche A., 1988. Les arbres et les représentations des proximités, Méthodes et Programmes, Masson, Paris.

Barthélemy J.-P. et Guénoche A., 1991. Trees and proximity representations. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, Ltd., Chichester.

Barthélemy, J. P. et McMorris, F. R., 1986. The median procedure for n-trees. Journal of Classification, 3(2), 329-334.

Barthelemy, J. P. et Monjardet, B., 1981. The median procedure in cluster analysis and social choice theory. Mathematical social sciences, 1(3), 235-267.

Baum, B. R., 1992. Combining trees as a way of combining data sets for phylogenetic inference, and the desirability of combining gene trees. Taxon, 3-10.

Bennett, K. D., 2013. Is the number of species on Earth increasing or decreasing? Time, chaos and the origin of species. Palaeontology, 56(6), 1305-1325.

Beck, R. M., Bininda-Emonds, O. R., Cardillo, M., Liu, F. G. R. et Purvis, A., 2006. A higher-level MRP supertree of placental mammals. BMC Evolutionary Biology, 6(1), 93.

Bertolini, F., Schiavo, G., Scotti, E., Ribani, A., Martelli, P. L., Casadio, R. et Fontanesi, L., 2014. High - throughput SNP discovery in the rabbit (Oryctolagus cuniculus) genome by next - generation semiconductor - based sequencing. Animal genetics, 45(2), 304-307.

Bhatt, H. B. et Singh, S. P., 2017. Phylogenetic and phenogram based diversity of haloalkaliphilic bacteria from the saline desert. Microbial Biotechnology: Technological Challenges and Developmental Trends, 373.

Bininda-Emonds, O. R., 2004. The evolution of supertrees. Trends in Ecology & Evolution, 19(6), 315-322.

Bininda-Emonds, O. R., Gittleman, J. L. et Steel, M. A., 2002. The (super) tree of life: procedures, problems, and prospects. Annual Review of Ecology and Systematics, 33(1), 265-289.

Bonnard, C., Berry, V., et Lartillot, N., 2006. Multipolar consensus for phylogenetic trees. Systematic biology, 55(5), 837-843.

Boc, A., Diallo, A. B. et Makarenkov, V., 2012. T-REX: a web server for inferring, validating and visualizing phylogenetic trees and networks. Nucleic acids research, 40(W1), W573-W579.

Boc, A., Di Sciullo, A. M. et Makarenkov, V., 2010. Classification of the indo-european languages using a phylogenetic network approach. In H. Locarek-Junge et C. Weihs (dir.), Classification as a Tool for Research, Berlin Heidelberg, Allemagne:Springer, 647-655.

Boc, A., Legendre, P. et Makarenkov, V., 2013. An efficient algorithm for the detection and classification of horizontal gene transfer events and identification of mosaic genes. In Algorithms from and for nature and life, Springer International Publishing, 253-260.

Boc, A., Philippe, H. et Makarenkov, V., 2010. Inferring and validating horizontal gene transfer events using bipartition dissimilarity. Systematic biology, 59(2), 195-211.

Bock, H. H., 2007. Clustering methods: a history of k-means algorithms. Selected contributions in data analysis and classification. Springer, Berlin, Heidelberg, 161-172.

Bouckaert, R., Lemey, P., Dunn, M., Greenhill, S., Alekseyenko, A., Drummond, A., ray, R., Suchard, M. et Atkinson, Q., 2012. Mapping the origins and expansion of he indo-european language family. Science, 337(6097), 957-960.

Bradley, P. S., Mangasarian, O. L. et Street, W. N., 1997. Clustering via concave minimization. Advances in neural information processing systems, 368-374.

Bandelt, H. J. et Dress, A. W., 1992. A canonical decomposition theory for metrics on a finite set. Advances in mathematics, 92(1), 47-105.

Brower, A.V., 2016. What is a cladogram and what is not?. Cladistics, 32(5), 573-576.

Bruen, T. C. et Bryant, D., 2008. Parsimony via consensus. Systematic biology, 57(2), 251-256.

Bryant, D., 2003. A classification of consensus methods for phylogenetics. DIMACS series in discrete mathematics and theoretical computer science, 61, 163-184.

Bryant, D., Filimon, F. et Gray, R. D., 2005. Untangling our past: languages, trees, splits and networks. The evolution of cultural diversity: A phylogenetic approach, 67-83.

Bryant, D. et Moulton, V., 2004. Neighbor-net: an agglomerative method for the construction of phylogenetic networks. Molecular biology and evolution, 21(2), 255-265.

Bryant, D., Tsang, J., Kearney, P. et Li, M., 2000. Computing the quartet distance between evolutionary trees. In Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics, 285-286.

Caliński, T. et Harabasz, J., 1974. A dendrite method for cluster analysis. Communications in Statistics-theory and Methods, 3(1), 1-27.

Cavalli-Sforza, L. L. et Edwards, A. W., 1967. Phylogenetic analysis. Models and estimation procedures. American journal of human genetics, 19(3 Pt 1), 233.

Celebi, M. E., Kingravi, H. A. et Vela, P. A., 2013. A comparative study of efficient initialization methods for the k-means clustering algorithm. Expert systems with applications, 40(1), 200-210.

Christenhusz, M. J. et Byng, J. W., 2016. The number of known plants species in the world and its annual increase. Phytotaxa, 261(3), 201-217.

Cotton, J. A. et Wilkinson, M., 2007. Majority-rule supertrees. Systematic biology, 56(3), 445-452.

Cosner, M. E., Jansen, R. K., Moret, B. M., Raubeson, L. A., Wang, L. S., Warnow, T. et Wyman, S., 2000. An empirical comparison of phylogenetic methods on chloroplast gene order data in Campanulaceae. In Comparative Genomics (99-121). Springer Netherlands.

Creevey, C. J., Fitzpatrick, D. A., Philip, G. K., Kinsella, R. J., O'Connell, M. J., Pentony, M. M., Travers S. A., Wilkinson M. et McInerney, J. O., 2004. Does a tree–like phylogeny only exist at the tips in the prokaryotes?. Proceedings of the Royal Society of London. Series B: Biological Sciences, 271(1557), 2551-2558.

Creevey, C. J. et McInerney, J. O., 2004. Clann: investigating phylogenetic information through supertree analyses. Bioinformatics, 21(3), 390-392.

Creevey, C. J. et McInerney, J. O., 2009. Trees from trees: Construction of phylogenetic supertrees using clann. Bioinformatics for DNA Sequence Analysis, 139-161.

Czelusniak, J., Goodman, M., Moncrief, N. D. et Kehoe, S. M., 1990. Maximum parsimony approach to construction of evolutionary trees from aligned homologous sequences. Methods in enzymology.

Darwin, C., 1871. The descent of man. London, 81(1), 130-1.

Daskalakis, C. et Roch, S., 2013. Alignment-free phylogenetic reconstruction: Sample complexity via a branching process analysis. The Annals of Applied Probability, 23(2), 693-721.

De Queiroz, A. et Gatesy, J., 2007. The supermatrix approach to systematics. Trends in Ecology & Evolution, 22(1), 34-41.

Dong, J., Fernández-Baca, D. et McMorris, F. R., 2010. Constructing majority-rule supertrees. Algorithms for Molecular Biology, 5(2).

Dong, J., Fernández-Baca, D., McMorris, F. R. and Powers, R. C., 2010. Majority-rule (+) consensus trees. Mathematical biosciences, 228(1), 10-15.

Doyle, J. J., 1992. Gene trees and species trees: molecular systematics as one-character taxonomy. Systematic Botany, 144-163.

Downie, S. R. et Palmer, J. D., 1992. Use of chloroplast DNA rearrangements in reconstructing plant phylogeny. Plant molecular systematics, 14-35.

Drineas, P., Frieze, A., Kannan, R., Vempala, S. et Vinay, V., 2004. Clustering large graphs via the singular value decomposition. Machine learning, 56(1), 9-33.

Dunn, J. C., 1974. Well-separated clusters and optimal fuzzy partitions. Journal of cybernetics, 4(1), 95-104.

Dyen, I., Kruskal, J. et Black, P., 1992. An Indoeuropean Classification: A Lexicostatistical Experiment. American Philosophical Society: Transactions of the American Philosophical Society. Philadelphie, PA, États-Unis: American Philosophical Society.

Edgar, R. C., 2004. MUSCLE: multiple sequence alignment with high accuracy and high throughput. Nucleic acids research, 32(5), 1792-1797.

Efron, B., 1979. Bootstrap methods: another look at the jackknife. The annals of Statistics, 1-26.

Eulenstein, O., Chen, D., Burleigh, J. G., Fernández-Baca, D., et Sanderson, M. J., 2004. Performance of flip supertree construction with a heuristic algorithm. Systematic Biology, 53(2), 299-308.

Felsenstein, J., 1981. Evolutionary trees from DNA sequences: a maximum likelihood approach. Journal of molecular evolution, 17(6), 368-376.

Felsenstein, J., 1985. Confidence limits on phylogenies: an approach using the bootstrap. Evolution, 39(4), 783-791.

Felsenstein, J., 1990. PHYLIP manual version 3.3. University Herbarium, University of California, Berkeley.

Felsenstein, J., 2004. Inferring phylogenies. Sunderland: Sinauer Associates, 2.

Felsenstein, J., 2005. PHYLIP (phylogeny inference package) Distributed by the author. Department of Genome Sciences, University of Washington, Seattle.

Felsenstein, J. et Sober, E., 1986. Parsimony and likelihood: an exchange. Systematic Zoology, 617-626.

Fitch, W. M., 2000. Homology: a personal view on some of the problems. Trends in genetics, 16(5), 227-231.

Fitch, W. M., 1971. Toward defining the course of evolution: minimum change for a specific tree topology. Systematic Biology, 20(4), 406-416.

Fitch, W. M. et Margoliash, E., 1967. Construction of phylogenetic trees. Science, 155(760), 279-284.

Forterre, P., 1997. Protein versus rRNA: problems in rooting the universal tree of life. ASM News-American Society for Microbiology, 63, 89-95.

Geisler, H. et List, J. M., 2013. Do languages grow on trees? The tree metaphor in the history of linguistics. Classification and evolution in biology, linguistics and the history of science. concepts–methods–visualization. Stuttgart: Franz Steiner Verlag, 111-24.

Goloboff, P. A. et Pol, D., 2002. Semi - strict supertrees. Cladistics, 18(5), 514-525.

Gordon, A. D., 1986. Consensus supertrees: the synthesis of rooted trees containing overlapping sets of labeled leaves. Journal of Classification, 3(2), 335-348.

Gray, R. D. et Atkinson, Q. D., 2003. Language-tree Divergence Times Support the natolian Theory of Indo-European Origin. Nature, 426(6965), 435–439.

Gray, R. D., Bryant, D. et Greenhill, S. J., 2010. On the shape and fabric of human history. Philosophical Transactions of the Royal Society of London B: Biological Sciences, 365(1559), 3923-3933.

Guénoche, A., 2013. Multiple consensus trees: a method to separate divergent genes. BMC bioinformatics, 14(1), 46.

Guindon, S. et Gascuel, O., 2003. A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood. Systematic biology, 52(5), 696-704.

Haeckel, E., 1866. Generelle Morphologie der Organismen allgemeine Grundzuge der organischen Formen-Wissenschaft, mechanisch begrundet durch die von Charles Darwin reformirte Descendenz-Theorie von Ernst Haeckel: Allgemeine Entwickelungsgeschichte der Organismen kritische Grundzuge der mechanischen Wissenschaft von den entstehenden Formen der Organismen, begrundet durch die Descendenz-Theorie, 2, Verlag von Georg Reimer.

Huang, Z., 1998. Extensions to the k-means algorithm for clustering large data sets with categorical values. Data Mining and Knowledge Discovery, 2(3), 283-304.

Hubert, L. et Arabie, P., 1985. Comparing partitions. Journal of classification, 2(1), 193-218.

Huelsenbeck, J. P. et Ronquist, F., 2001. MRBAYES: Bayesian inference of phylogenetic trees. Bioinformatics, 17(8), 754-755.

Huson, D. H. et Bryant, D., 2005. Application of phylogenetic networks in evolutionary studies. Molecular biology and evolution, 23(2), 254-267.

Jansson, J., Shen, C. et Sung, W. K., 2013. September. Algorithms for the majority rule (+) consensus tree and the frequency difference consensus tree. In International Workshop on Algorithms in Bioinformatics (141-155). Springer, Berlin, Heidelberg.

Jiang, Y. et Zhang, J., 2014. June. Parallel K-Medoids clustering algorithm based on Hadoop. In Software Engineering and Service Science (ICSESS), 2014 5th IEEE International Conference on (pp. 649-652). IEEE.

Jukes, T. H. et Cantor, C., 1969. Mammalian Protein Metabolism. In Evolution of protein molecules (H. N. Munro, eds). New York: Academic Press, 21-132.

Katoh, K., Kuma, K. I., Toh, H. et Miyata, T., 2005. MAFFT version 5: improvement in accuracy of multiple sequence alignment. Nucleic acids research, 33(2), 511-518.

Katoh, K. et Standley, D. M., 2013. MAFFT multiple sequence alignment software version 7: improvements in performance and usability. Molecular biology and evolution, 30(4), 772-780.

Kaufman, L. R. et Rousseeuw, P. P. J., 1990. Finding groups in data: An introduction to cluster analysis. Seriesin Probability & MathematicalStatistics, 34(1), 111-112.

Kaur, N. K., Kaur, U. and Singh, D. D., 2014. K-Medoid Clustering Algorithm-A Review. International Journal of Computer Application and Technology (IJCAT) Volume, 1, 2349-1841.

Kimura, M., 1980. A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. Journal of Molecular Evolution, 16(2), 111-120.

Kuhner, M. K. et Yamato, J., 2015. Assessing Differences Between Ancestral Recombination Graphs. Journal of molecular evolution, 80(5-6), 258-264.

Lapointe, F.-J. et Cucumel, G., 1997. The average consensus procedure: combination of weighted trees containing identical or overlapping sets of taxa. Systematic Biology, 46(2), 306-312.

Layeghifard, M., Peres-Neto, P. R. and Makarenkov, V., 2013. Inferring explicit weighted consensus networks to represent alternative evolutionary histories. BMC evolutionary biology, 13(1), 274.

Letunic, I. et Bork, P., 2006. Interactive Tree Of Life (iTOL): an online tool for phylogenetic tree display and annotation. *Bioinformatics*, *23*(1), 127-128.

Letunic, I. et Bork, P., 2016. Interactive tree of life (iTOL) v3: an online tool for the display and annotation of phylogenetic and other trees. Nucleic acids research, 44(W1), W242-W245.

Li, K. B., 2003. ClustalW-MPI: ClustalW analysis using distributed and parallel computing. Bioinformatics, 19(12), 1585-1586.

Lloyd, S. P., 1957. Binary block coding. Bell Labs Technical Journal, 36(2), 517-535.

MacQueen, J., 1967. Some methods for classification and analysis of multivariate observations. In Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, 1(281-297), 14).

Maddison, D. R., 1991. The discovery and importance of multiple islands of most-parsimonious trees. Systematic Biology, 40(3), 315-328.

Makarenkov, V., 2001. T-Rex: reconstructing and visualizing phylogenetic trees and reticulation networks. Bioinformatics, 17, 664-668.

Makarenkov, V., Boc, A. et Diallo, Alpha. B., 2007. La dissimilarite de bipartition et son utilisation pour detecter les transferts horizontaux de genes, Proceedings of the 14-emes Rencontres de la SFC 2007, ENST de Paris, France, 90-93.

Makarenkov, V. et Leclerc, B., 1999. The fitting of a tree metric to a given dissimilarity with the weighted least squares criterion. Journal of Classification, 16, 3-26.

Makarenkov, V. et Leclerc, B., 2000. Comparison of additive trees using circular orders. Journal of Computational Biology, 7(5), 731-744.

Makarenkov, V. et Legendre, P., 2001. Optimal variable weighting for ultrametric and additive trees and K-means partitioning: Methods and software. Journal of Classification, 18(2), 245-271.

Makarova, K. S., Aravind, L., Galperin, M. Y., Grishin, N. V., Tatusov, R. L., Wolf, Y. I. et Koonin, E. V., 1999. Comparative genomics of the Archaea (Euryarchaeota): evolution of conserved protein families, the stable core, and the variable shell. Genome research, 9(7), 608-628.

Margush, T. et McMorris, F. R., 1981. Consensusn-trees. Bulletin of Mathematical Biology, 43(2), 239-244.

Matte-Tailliez, O., Brochier, C., Forterre, P. et Philippe, H., 2002. Archaeal phylogeny based on ribosomal proteins. Molecular Biology and Evolution, 19(5), 631-639.

McMahon, M. M. et Sanderson, M. J., 2006. Phylogenetic supermatrix analysis of GenBank sequences from 2228 papilionoid legumes. Systematic biology, 55(5), 818-836.

Moon, J. et Eulenstein, O., 2017. Synthesizing large-scale species trees using the strict consensus approach. Journal of Bioinformatics and Computational Biology, 1740002.

Morgenstern, B., 2014. Multiple Sequence Alignment with DIALIGN. In Multiple Sequence Alignment Methods, Humana Press, 191-202.

Nelson, G., 1979. Cladistic analysis and synthesis: principles and definitions, with a historical note on Adanson's Familles des Plantes (1763–1764). Systematic Biology, 28(1), 1-21.

Ng, R. T. et Han, J., 2002. CLARANS: A method for clustering objects for spatial data mining. IEEE transactions on knowledge and data engineering, 14(5), 1003-1016.

Olsen, G. J. et Woese, C. R., 1997. Archaeal genomics: an overview. Cell, 89(7), 991-994.

Pace, N. R., 1997. A molecular view of microbial diversity and the biosphere. Science, 276, 734-740.

Page, R. D., 1989. Comments on component - compatibility in historical biogeography. Cladistics, 5(2), 167-182.

Phipps, J.B., 1971. Dendrogram topology. Systematic zoology, 20(3), 306-308.

Ragan, M. A., 1992. Matrix representation in reconstructing phylogenetic relationships among the eukaryotes. Biosystems, 28(1), 47-55.

Rand, W. M., 1971. Objective criteria for the evaluation of clustering methods. Journal of the American Statistical association, 66(336), 846-850.

Renfrew, C., 1988. Archaeology and language: the puzzle of Indo-European origins. Londres, Royaume-Uni: J. Cape.

Renfrew, C., 1990. Archaeology and language: the puzzle of Indo-European origins. CUP Archive.

Rivera, M. C., Jain, R., Moore, J. E. et Lake, J. A., 1998. Genomic evidence for two functionally distinct gene classes. Proceedings of the National Academy of Sciences, 95(11), 6239-6244.

Robinson, D. F. et Foulds, L. R., 1981. Comparison of phylogenetic trees. Mathematical Biosciences, 53(1), 131-147.

Rogozin, I. B., Pavlov, Y. I., Bebenek, K., Matsuda, T. et Kunkel, T. A., 2001. Somatic mutation hotspots correlate with DNA polymerase η error spectrum. Nature immunology, 2(6), 530-536.

Rousseeuw, P. J., 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Journal of computational and applied mathematics, 20, 53-65.

Saitou, N. et Nei, M., 1987. The neighbor-joining method: a new method for reconstructing phylogenetic trees. Journal of Molecular Biololy Evolution., 4, 406-425.

Sanderson, M. J., 2002. Estimating absolute rates of molecular evolution and divergence times: a penalized likelihood approach. Molecular biology and evolution, 19(1), 101-109.

Sanderson, M. J., Purvis, A. et Henze, C., 1998. Phylogenetic supertrees: assembling the trees of life. Trends in Ecology & Evolution, 13(3), 105-109.

Santos, J. M. et Embrechts, M., 2009. September. On the use of the adjusted rand index as a metric for evaluating supervised classification. In International Conference on Artificial Neural Networks (175-184). Springer, Berlin, Heidelberg.

Schleicher, A., 1863. Darwinism tested by the science of language, trans. AV M. Bikkers. Böhlau. Reprinted in:(1983) Linguistics and evolutionary theory: Three essays, ed. K. Koerner.

Sneath, P. H. et Sokal, R. R., 1973. Numerical taxonomy. The principles and practice of numerical classification.

Soares, A., Râbelo, R. et Delbem, A., 2017. Optimization based on phylogram analysis. Expert Systems with Applications, 78, 32-50.

Sokol, R. R. et Rohlf, F. J., 1981. Biometry: the principles and practice of statistics in biological research. WH Freeman Inc, San Francisco, 849.

Stamatakis, A., Hoover, P. et Rougemont, J., 2008. A rapid bootstrap algorithm for the RAxML web servers. Systematic biology, 57(5), 758-771.

Steel, M. A. and Penny, D., 1993. Distributions of tree comparison metrics—some new results. Systematic biology, 42(2), 126-141.

Steinley, D., Brusco, M. J. et Hubert, L., 2016. The variance of the adjusted Rand index. Psychological methods, 21(2), 261.

Stockham, C., Wang, L. S. et Warnow, T., 2002. Statistically based postprocessing of phylogenetic analysis by clustering. Bioinformatics, 18(suppl_1), S285-S293.

Strimmer, K. et Von Haeseler, A., 1996. Quartet puzzling: a quartet maximum-likelihood method for reconstructing tree topologies. Molecular Biology and Evolution, 13(7), 964-969.

Sul, S. J. et Williams, T. L., 2008. September. An Experimental Analysis of Robinson-Foulds Distance Matrix Algorithms. In Esa (793-804).

Swadesh, M., 1952. Lexico-statistic dating of prehistoric special reference to north american indians and eskimos. Philosophical Society, 96(4), 452–463.

Swenson, M. S., Suri, R., Linder, C. R. et Warnow, T., 2011. An experimental study of Quartets MaxCut and other supertree methods. Algorithms for Molecular Biology, 6(1), 7.

Swofford, D. L., 2002. PAUP* version 4.0. Phylogenetic analysis using parsimony (and other methods).

Thompson, J. D., Gibson, T. et Higgins, D. G., 2002. Multiple sequence alignment using ClustalW and ClustalX. Current protocols in bioinformatics, 2-3.

Tibshirani, R., Walther, G. et Hastie, T., 2001. Estimating the number of clusters in a data set via the gap statistic. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 63(2), 411-423.

Trask, R. L., 2000. The dictionary of historical and comparative linguistics. Psychology Press.

Vachaspati, P. et Warnow, T., 2017. FastRFS: fast and accurate Robinson-Foulds Supertrees using constrained exact optimization. Bioinformatics, 33(5), 631-639.

Wareham, H. T., 1985. An efficient algorithm for computing Ml consensus trees (Doctoral dissertation, Memorial University of Newfoundland.).

Wheeler, W. C., 2003. Implied alignment: a synapomorphy - based multiple - sequence alignment method and its use in cladogram search. Cladistics, 19(3), 261-268.

Wilkinson, A., Hill, M. et Gollan, P., 2001. The sustainability debate. International Journal of Operations & Production Management, 21(12), 1492-1502.

Wilkinson, M., Pisani, D., Cotton, J. A. et Corfe, I., 2005. Measuring support and finding unsupported relationships in supertrees. Systematic Biology, 54(5), 823-831.

Willems, M., Lord, E., Laforest, L., Labelle, G., Lapointe, F. J., Di Sciullo, A. M. et Makarenkov, V., 2016. Using hybridization networks to retrace the evolution of Indo-European languages. BMC evolutionary biology, 16(1), 180.

Willems, M., Tahiri, N. et Makarenkov, V., 2014. A new efficient algorithm for inferring explicit hybridization networks following the Neighbor-Joining principle. Journal of bioinformatics and computational biology, 12(05), p.1450024.

Witten, D. M. et Tibshirani, R., 2010. A framework for feature selection in clustering. Journal of the American Statistical Association, 105(490), 713-726.

Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G.J., Ng, A., Liu, B., Philip, S.Y. et Zhou, Z.H., 2008. Top 10 algorithms in data mining. Knowledge and information systems, 14(1), 1-37.

Yeung, K. Y. and Ruzzo, W. L., 2001. Details of the adjusted rand index and clustering algorithms, supplement to the paper an empirical study on principal component analysis for clustering gene expression data. Bioinformatics, 17(9), 763-774.

Yujian, L. et Bo, L., 2007. A normalized Levenshtein distance metric. IEEE transactions on pattern analysis and machine intelligence, 29(6), 1091-1095.

Zhang, Q. and Couloigner, I., 2005. A new et efficient k-medoid algorithm for spatial clustering. Computational Science and Its Applications–ICCSA 2005, 207-224.

Zhi-Xiong, J., Bao-Sheng, W., Zhi-Feng, S., Yong, T. et Shuo-Wei, T., 2015. A Method Constructing the Phylogenetic Tree of Malware Based on UPGMA. Information Security and Technology, 3, 006.