UNIVERSITÉ D'EVRY-VAL D'ESSONNE

U.F.R de Sciences Fondamentales et Appliquées

pour obtenir le titre de **Docteur en Sciences**

spécialité **Bioinformatique**

présentée par

Laure Vescovo

Outils et méthodes pour la classification pyramidale de données biologiques

---

Soutenue le
devant le jury composé de :

M.	Antoine	DE DARUVAR	Rapporteur
M.	Eric	COISSAC	Rapporteur
M.	Bernard	PRUM	Examinateur
M.	Patrice	BERTRAND	Examinateur
Mme	Géraldine	AUDE-POLAILLON	Examinateur
M.	Jean-Christophe	AUDE	Examinateur
M.	Jean-Loup	RISLER	Directeur

Table des matières

Introduction

Les premières approches de classifications systématiques sont apparues au XVIII^{eme} siècle, avec l'établissement des premières taxonomies animales et végétales. Aujourd'hui, étant donnée l'ampleur des programmes de séquençage de génomes complets, le rythme d'acquisition des données en biologie croît de manière spectaculaire. L'objectif est d'extraire l'information de ces séquences (nucléiques et protéiques) grâce aux outils bioinformatiques permettant notamment la comparaison de séquences, l'alignement multiple, la classification en famille ou encore la prédiction de structures. Depuis les premières classifications, les méthodes statistiques ont une place prépondérante dans ces analyses car elles sont particulièrement adaptées aux données biologiques complexes. Elles permettent d'organiser, de gérer et traiter la masse et la diversité de ces données. La statistique, née au début du vingtième siècle, a manipulé des données pendant un demi-siècle sans véritables outils de calcul. Depuis, l'essor des calculateurs électroniques a permis le développement de nouvelles théories et d'outils. Ainsi, des méthodes propres à l'analyse biologique ont été créées.

Parmi les approches statistiques, les méthodes de classification ont pour but de construire une partition ou une suite de partition emboîtées et ainsi d'obtenir une représentation graphique simple des objets étudiés. L'idée des méthodes de recouvrement, c'est à dire des méthodes faisant apparaître des classes non-disjointes est présente dans [Sneath and Sokal, 1973]. En 1968, Jardine et Sibson avaient déja proposé une généralisation du modèle hiérarchique dans le but de conserver plus d'information sur les relations entre les objets que ne le permet ce type de classification. [Hubert, 1974] a aussi considéré des classifications conduisant à des classes recouvrantes dans le cadre de la théorie des graphes. Cependant, peu de personnes se sont intéressées aux méthodes de recouvrement, du fait de la complexité du problème. L'introduction de la notion d'ordre en classification est issue d'un problème d'archéologie, celui de la tendance évolutive sur les données. [Robinson, 1951] modélise le problème par la recherche d'un ordre sur l'ensemble des données traduisant "au mieux" une évolution, dans ce cas chronologique, en se basant sur le fait que des dépôts proches dans le temps auraient des distributions similaires des différents composants. Les pyramides [Diday, 1984a, Bertrand and Diday, 1990a] et les pseudo-hiérarchies [Fichet, 1984, Durand and Fichet, 1988] sont des méthodes de classification qui généralisent les hiérarchies en permettant la représentation de classes empiétantes.

Dans ce travail, nous souhaitons appliquer les propriétés inhérentes de la classification pyramidale à l'analyse de données biologiques. L'intérêt des classes empiétantes et de l'ordre partiel induit sur les données sera montré à plusieurs reprises dans ce document. Après avoir présenté les méthodes "classiques" de construction d'arbres, nous introduirons la classification pyramidale, ainsi qu'un algorithme permettant de calculer une pyramide : la classification ascendante pyramidale. Il s'agit d'une adaptation de son homologue hiérarchique au modèle pyramidal. Les algorithmes agglomératifs ont pour principe de fusionner itérativement des objets ou des classes jusqu'à obtenir une classe unique. A chaque étape de l'algorithme, les nouvelles classes créées sont de moins en moins homogènes. Cependant, certaines conditions particulières peuvent induire des inversions dans la représentation pyramidale. Une inversion est une classe, agrégeant deux autres classes, indiquée comme plus homogène que chacune de ses deux sous-classes. Ce biais perturbe la lisibilité. Il existe une bijection entre les pyramides et les matrices de Robinson. Or, lors de la présence d'inversion, la matrice de dissimilarité induite équivalente à la pyramide n'est pas de Robinson. La correction de ce biais est la première partie du travail réalisé : nous proposons d'utiliser une nouvelle méthode de construction de pyramide. Nous verrons les deux types d'approches que nous avons développées pour consolider les pyramides : une étape de filtrage (local ou global) de la matrice afin de la rendre robinsonienne ; un algorithme de reconstruction de la pyramide à partir de celle-ci.

La classification pyramidale a déja été utilisée pour l'étude de protéomes des végétaux [Louis et al., 2001] ou la comparaison systématique de 70 protéomes (soit 240000 séquences) pour le projet Teraprot [Teraprot, 2002]. Les domaines d'application de cette approche en biologie sont à étendre, par exemple dans l'analyse du transcriptome. Méthodologiquement, nous avons décidé de l'appliquer à l'alignement multiple de séquences, et plus particulièrement à l'approche progressive. Celui-ci est à la base de nombreuses analyses de séquences et couvre un vaste champ d'applications : annotation de séquences, prédiction fonctionnelle et structurale (structure secondaire et tertiaire), caractérisation de famille de protéines, étude phylogénétique... L'approche progressive, apparue dans les annés 80, utilise une structure de guidage pour déterminer l'ordre des séquences à aligner. C'est lors de cette étape que nous souhaitons appliquer la classification pyramidale et ainsi montrer l'intérêt de ses propriétés complémentaires pour l'analyse de données biologiques.

Le *premier chapitre* présente un état de l'art des méthodes de construction d'arbres utilisées pour l'analyse de données biologiques.

Dans un premier temps, nous définirons les contextes biologique dans lesquels les différentes méthodes de construction d'arbre s'appliquent. Afin d'étudier l'expression des gènes, deux approches complémentaires peuvent être réalisées : l'analyse du transcriptome, ainsi que l'analyse du protéome. Deux techniques d'étude à grande échelle ont rendu possible ces analyses, les puces à ADN pour le premier et le séquençage systématique de génomes dans le cas du protéome. Leur

finalité commune est la prédiction, l'identification de fonctions, de structures, grâce à la comparaison entre différents états biologiques ou différents organismes. Après la description de ces deux approches, nous présenterons la phylogénie qui est l'étude de la formation et de l'évolution des organismes vivants en vue d'établir leur parenté.

Après avoir défini la forme de représentation des données biologiques que nous utilisons, la suite de ce chapitre est consacrée à deux types de méthodes de construction d'arbre : les méthodes de classification et les méthodes basées sur le principe d'évolution minimum. La plupart des méthodes de classification visent à étudier un ensemble de données afin d'établir l'existence de similarité ou non entre les données. Elles permettent de représenter de façon synthétique les ensembles de données. En outre, elles permettent également un traitement automatique de l'information des ensembles de données volumineux. Les méthodes de classification mettent en évidence des relations de proximité dans un ensemble d'objets et permettent de les représenter graphiquement. Par construction, les objets ainsi réunis tendent à former des classes homogènes. Dans ce document, nous nous plaçons dans le cas de la classification automatique non-supervisée et plus particulièrement de la classification hiérarchique [Gordon, 1996a]. Cette méthode est une des plus fréquemment utilisées du fait de son efficacité et de la simplicité d'interprétation des résultats obtenus.

Le principe d'évolution minimum a été proposé comme principe de base de l'inférence phylogénétique. Etant donné une matrice de distances, ce principe implique d'abord d'estimer la longueur d'une topologie donnée et alors de déterminer la topologie de plus faible longueur. De nombreuses variantes de ce principe existent, dépendant de la façon d'estimer la longueur des branches et de celle dont l'arbre est calculé à partir de ces branches.

Le *deuxième chapitre* est dédié à la classification pyramidale, problématique principale de ce mémoire. Les pyramides [Diday, 1984a, Bertrand and Diday, 1990a] et les pseudo-hiérarchies [Fichet, 1984, Durand and Fichet, 1988] sont des méthodes de classification qui généralisent les hiérarchies en permettant la représentation de classes empiétantes. Nous utiliserons le terme pyramide pour caractériser indifféremment ces méthodes. Dans ce chapitre, nous introduisons de façon formelle les pyramides afin de mettre en évidence leurs similitudes et leurs différences par rapport aux hiérarchies. Nous détaillerons ensuite l'algorithme que nous utilisons, ainsi que le biais inhérent à celui-ci. La correction de ce biais est l'objet de ce chapitre, nous souhaitons fournir une méthode robuste de construction de pyramide. Pour cela, nous présentons plusieurs solutions afin de consolider l'algorithme et ainsi supprimer les inversions. Nous présentons d'abord une approche heuristique pour obtenir une matrice de Robinson. Puis, nous présentons les régressions isotones comme une solution optimale pour obtenir une matrice robinsonienne. Pour chaque solution, nous proposons un algorithme de consolidation de matrice. Nous proposons également un algorithme reconstruisant une pyramide à partir d'une matrice de Robinson. Nous comparerons ces solutions sous plusieurs aspects, notamment à l'aide de simulations. De

plus, nous montrerons que la pyramide corrigée est plus proche des données initiales au sens du critère des moindres carrés.

Dans le *troisième chapitre*, nous abordons la problématique des alignements multiples de séquences, une thématique importante en bioinformatique. Lorsque l'on étudie un gène (ou une protéine) de fonction inconnue, l'une des premières questions que l'on se pose à propos d'un gène (ou d'une protéine) est si celui-ci (ou celle-ci) est lié(e) ou non avec un autre gène (ou une autre protéine). Une ressemblance au niveau de la séquence suggère tout d'abord que deux protéines sont homologues mais aussi qu'elles peuvent partager des fonctions identiques. Le nombre de séquences (protéiques ou nucléiques) étant très important, il est possible d'identifier des domaines ou motifs communs à un groupe de molécules. Ces analyses sont réalisées grâce à l'alignement de séquences. Après avoir présenté les notions nécessaires à la compréhension de l'alignement de séquences (matrices de substitution, présence de brèches, score d'alignement), ainsi que l'alignement de deux séquences et ses applications, nous détaillerons dans cette partie l'alignement multiple de séquences.

L'alignement multiple de séquences est à la base de nombreuses analyses de séquences et couvre un vaste champ d'applications : annotation de séquences ; prédiction fonctionnelle et structurale (structure secondaire et tertiaire) ; caractérisation de famille de protéines ; étude phylogénétique... L'importance d'un bon alignement a été soulignée notamment pour la détermination de la structure secondaire des protéines par [Jones, 1999] et dans le cas d'études phylogénétiques [Phillips and Wheeler, 2000]. C'est pourquoi l'amélioration, ne serait ce qu'infime, des méthodes automatiques permettant de générer un alignement multiple est primordiale. Le calcul de l'alignement multiple est donc un thème de recherche particulièrement important.

Un soin tout particulier sera apporté à la description de l'approche progressive [Feng and Doolittle, 1987], une des méthodes les plus utilisées pour réaliser un alignement multiple. Cette stratégie repose sur un algorithme en trois étapes : un alignement par paire de toutes les séquences est calculé ; puis les scores attribués à ces alignements permettent de représenter les distances entre séquences ; enfin, les séquences sont alignées progressivement en suivant l'ordre donné par les branches de l'arbre. Cette approche présente de nombreux avantages, notamment une meilleure précision et une rapidité de calcul.

Dans le *quatrième chapitre*, nous appliquons la classification ascendante pyramidale à l'alignement multiple de séquences, plus particulièrement à l'approche progressive. Apparue dans les années 80, cette stratégie n'a été améliorée que dans la façon d'aligner les paires de séquences et/ou l'alignement progressif des séquences. L'influence de la seconde étape n'a jamais été étudiée même si dans ClustalW [Thompson et al., 1994], le programme de référence, les auteurs ont modifié la méthode utilisée initialement. La classification pyramidale étant une méthode généralisant la classification hiérarchique, nous avons souhaité utiliser ses propriétés supplémentaires.

Dans un premier temps, afin de montrer l'importance d'une bonne structure de guidage pour la qualité d'un alignement multiple, nous avons réalisé une étude comparative. Nous avons sélectionné un ensemble de méthodes de construction d'arbre et nous les avons implémenté dans ClustalW en plus de la méthode par défaut, le Neighbor-Joining [Saitou and Nei, 1987] : BioNJ [Gascuel, 1997] et Weighbor [Bruno et al., 2000], deux méthodes dérivées du NJ ; d'autres méthodes également basées sur le principe d'évolution minimum [Desper and Gascuel, 2002] ; la classification ascendante hiérarchique [Gordon, 1996b] et la classification pyramidale. La première partie de ce chapitre sera consacrée à la description d'un algorithme permettant de construire une hiérarchie ordonnée à partir d'une pyramide. Afin d'étudier uniquement l'impact de l'ordre induit par la pyramide, nous avons défini un algorithme permettant de calculer une hiérarchie ordonnée à partir de la matrice de Robinson équivalente à la pyramide. Ensuite, nous présenterons la comparaison de ces méthodes, grâce à une étude sur différentes bases d'alignements de référence. Même si ces méthodes sont habituellement classées sous différentes catégories (reconstruction phylogénétique, classification...), nous les regroupons sous une seule et même appellation : méthodes de construction d'arbres ou méthodes de distance.

L'approche proposée pour utiliser l'ordre induit par la pyramide est une approximation. C'est pourquoi nous présenterons ensuite une approche mixte, basée sur les stratégies d'alignement local et global. Les propriétés des pyramides seront ainsi exploitées au mieux. Après avoir présenté l'algorithme général, nous décrirons plusieurs perspectives ainsi qu'un exemple détaillé montrant la pertinence de cette approche.

Chapitre 1

Analyse de données biologiques

En France, on présente traditionnellement l'analyse de données comme une discipline scientifique couvrant deux classes de méthodes (*cf.* figure 1.1 page suivante) : "les analyses factorielles" qui relèvent de la géométrie euclidienne et qui cherchent à représenter, au mieux, dans un espace réduit les corrélations entre n variables aléatoires ; "la classification automatique" qui cherche à construire une partition ou un arbre de classification à partir d'un indice de proximité et d'un critère d'aggrégation.

Les fondements de l'analyse factorielle sont très certainement à mettre au crédit de [Spearman, 1904] quand il a cherché à exprimer sous forme d'équation sa vision unidimensionnelle de l'intelligence. Cependant, son travail a très largement profité des avancées faites par l'anglais Francis Galton (1822-1911) à qui l'on doit la notion de corrélation, même si plus tard on retiendra surtout les travaux de Karl Pearson (1857-1936) dans ce domaine. La contribution majeure de Spearman va être d'expliciter la notion de variables dans un modèle mathématique permettant de les mesurer et de calculer leurs liaisons avec les observations. Mais l'importance de la contribution de Spearman ne réside pas que dans cette découverte. Elle réside dans la démarche qu'il a utilisée pour l'obtenir : écrire une équation, un modèle dirons-nous plus tard, dont certaines des variables puissent être estimées par le calcul alors qu'elles ne peuvent pas faire l'objet d'observations directes [Reuchlin, 2003]. En France, J.P. Benzecri [Benzecri, 1973, Benzecri, 1976] a quand a lui marqué de son empreinte ce qu'on appellera la nouvelle école française d'analyse factorielle [Pages et al., 1979].

Les premières classifications d'objets sont apparues au XVIII^{eme} siècle, avec l'établissement des premières taxonomies animales et végétales. Les données récoltées ont été classées en fonction de leurs rapports ou filiation et de leurs ressemblances par les naturalistes. Afin de comprendre la complexité du réel, notre entendement nous contraint à classifier des objets que nous devons traiter en catégories. En philosophie, celles-ci ont été considérées par deux approches :

– Les catégories pré-existent aux études. Elles ne sont découvertes et décrites que plus ou moins parfaitement. Cette démarche est la plus ancienne (Moyen Âge) et est désignée sous le nom : *réalisme*.
– Les catégories sont estimées comme des regroupements ad hoc, comme par exemple des classifications fonctionnelles. Dans l'opposition « fruits comestibles » *versus* « fruits vénéneux », c'est l'effet observé qui conduit à la classification. Cette démarche opposée au réalisme du Moyen Âge fut nommée *nominalisme*.

Bertrand Russell [Russell, 1914] fait remarquer dans ses ouvrages que si l'on devait les nommer aujourd'hui, on permuterait les deux appellations.

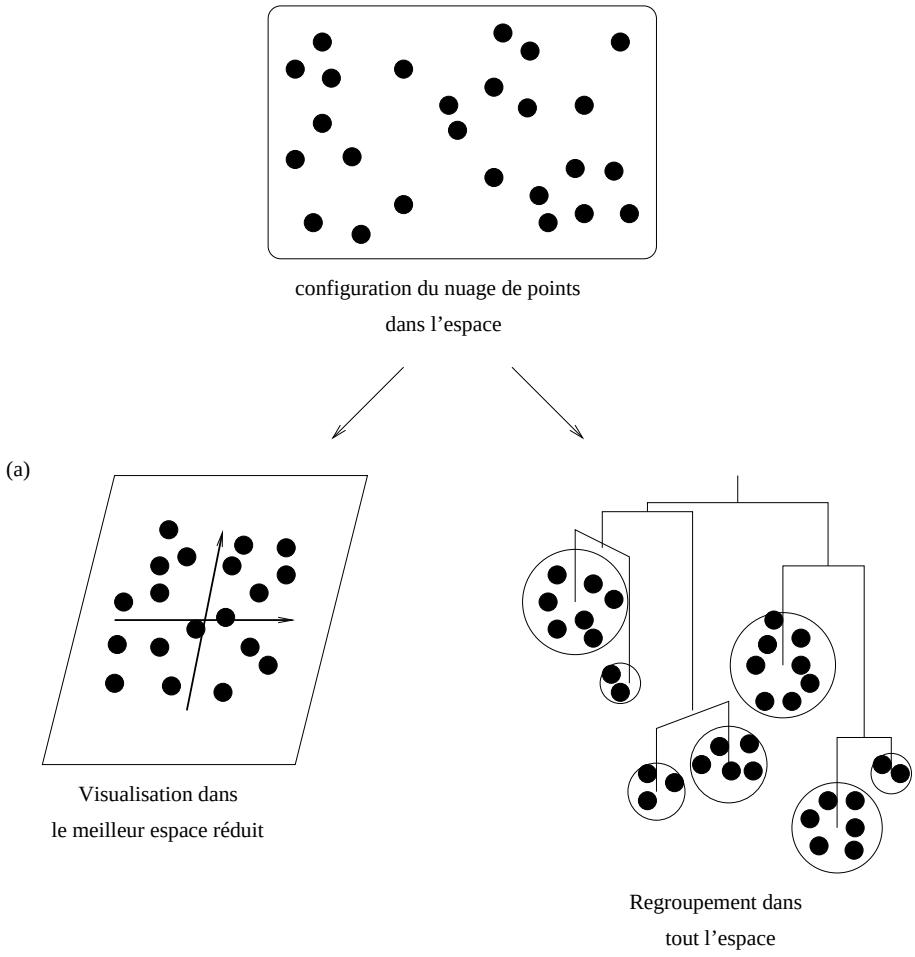


Fig. 1.1: Les deux types de méthodes statistiques – (a) Méthodes factorielles et (b) méthodes de classification. Figure tirée de [Lebart et al., 1995]

La statistique descriptive permet de représenter de façon vivante et assimilable des informations statistiques en les simplifiant et en les schématisant. La statistique multidimensionnelle en est la généralisation naturelle lorsque ces informations concernent plusieurs variables ou dimensions. Mais le passage au multidimensionnel induit un changement qualitatif important.

La réalité multidimensionnelle n'est pas seulement simplifiée parce que complexe, mais aussi explorée parce que cachée.

Née au début du vingtième siècle, la statistique a manipulé des données pendant un demi-siècle sans véritables outils de calcul. A défaut d'être repensée, la statistique s'est cependant enrîchie. Jusqu'à nos jours, des changements tout à fait notables ont eu lieu : les outils existants ont été améliorés, de nouveaux outils sont apparus, de nouveaux domaines d'application ont été explorés. Il n'est pas rare maintenant de traiter des tableaux correspondant à des milliers d'observations et des centaines de variables. Le changement d'échelle du volume des données a rapidement conduit à modifier les outils eux-mêmes et à imaginer de nouveaux outils dans le cadre de nouvelles approches.

La levée de l'obstacle de calcul a eu pour effet de diffuser l'emploi des techniques de type algorithmique, au premier rang desquelles se trouvent les méthodes de classification automatique, les méthodes impliquant des algorithmes coûteux, les méthodes d'estimation par maximum de vraisemblance ou encore de programmation dynamique. La classification automatique vise à créer ces catégories à partir de procédés ne faisant intervenir que les données et pas la subjectivité de l'expérimentateur. Il serait d'ailleurs plus exact de dire : ne faisant pas intervenir la subjectivité de l'expérimentateur par autre chose que le choix des représentations qu'il utilise : si l'on classifie des objets en considérant leur plus grande dimension, on n'obtiendra pas en général le même classement qu'en les classifiant par leurs poids. Le résultat d'une classification peut être soit une partition soit une hiérarchie (mathématiques).

Les méthodes statistiques sont particulièrement adaptées aux données biologiques complexes. La biologie moléculaire est apparue au XXe siècle, à la suite de l'élaboration des lois de la génétique, la découverte des chromosomes et l'identification de l'ADN comme support chimique de l'information génétique. Ces découvertes ont entraîné un accroissement important des connaissances et des domaines de recherche vers lesquels se tourner. Depuis une vingtaine d'années, le rythme d'acquisition des données en biologie croît de manière spectaculaire ; notamment avec les grands programmes de séquençage et leurs applications (protéome, transcriptome). La statistique et l'informatique offrent une panoplie riche et complexe de méthodes pour organiser, gérer et traiter la masse et la diversité de ces données. Des méthodes propres à l'analyse de biologie ont été créées, par exemple pour retracer l'évolution des espèces. Dans ce chapitre, nous nous intéressons aux méthodes permettant une représentation visuelle des données, c'est à dire un arbre.

Après avoir présenté le dogme central de la biologie moléculaire ainsi que la génomique, nous détaillerons différents algorithmes qui sont utilisés pour analyser les données génomiques. Nous décrirons d'abord les méthodes de classification, et plus particulièrement la classification ascendante hiérarchique. Ensuite, nous présenterons plusieurs méthodes basées sur le principe d'évolution minimum. Nous montrerons que ces deux types d'approches ont comme points com-

mun de débuter d'une matrice de distances pour construire un arbre et une forte ressemblance au point de vue algorithmique.

1.1 La biologie moléculaire et la génomique comparative

Cette partie permet de définir le cadre biologique dans lequel les différentes méthodes de construction d'arbre s'appliquent.

1.1.1 Le dogme central de la biologie moléculaire

Au XXème siècle, d'importantes découvertes en biologie telles que l'élaboration des lois de la génétique, la découverte des chromosomes, l'identification de l'ADN comme support chimique, ainsi que sa structure en double hélice (par Watson et Crick en 1953), ont mené à l'apparition d'une nouvelle discipline : la biologie moléculaire. Celle-ci est définie ainsi par l'encyclopédie en ligne wikipédia : "Au croisement de la génétique, de la biochimie et de la physique, la biologie moléculaire est une discipline scientifique dont l'objet est la compréhension des mécanismes de fonctionnement de la cellule au niveau moléculaire". D'une façon élargie, ce terme désigne également l'ensemble de techniques mises en place pour l'étude de ce fonctionnement. A la fin des années 50, Francis Crick a énoncé le "dogme central de la biologie moléculaire" : chez tous les êtres vivants, l'information est transmise dans un seul sens : de l'ADN (contenant l'information) qui sera transcrit en ARN (une structure transitoire) à son tour traduit en protéines. Ce principe est illustré par la figure 1.2 page suivante.

Chez les êtres vivants, l'information génétique est stockée dans des polymères connus sous le nom de nucléotides ou acides nucléiques. Il en existe deux types : l'Acide DésoxyriboNucléique (ADN) et l'Acide RiboNucléique (ARN). L'ADN est une molécule formée de l'assemblage linéaire de quatre nucléotides. Chacun est composé d'une base azotée (adénine, cytosine, guanine, thymine), d'un sucre (désoxyribose) et d'un groupement phosphate. L'usage a consacré l'utilisation des simples lettres A, C, G et T pour décrire la séquence de l'ADN. L'ARN est lui aussi une molécule comprenant quatre types de nucléotides, où le sucre est le ribose et où l'uracile (U) remplace la thymine. Les nucléotides sont à l'origine de la transmission de l'information entre les éléments du système et entre les générations grâce à leur capacité à former des paires complémentaires (A-T(U), G-C) de façon stable et spécifique. Cette stabilité et spécificité sont inhérentes à la géométrie des bases et à leur possibilité de former des liaisons hydrogène. Comme chaque base ne peut s'apparier qu'avec une seule autre, le code qu'elle porte est univoque.

Le dogme central repose sur deux postulats permettant le passage unidirectionnel de l'in-

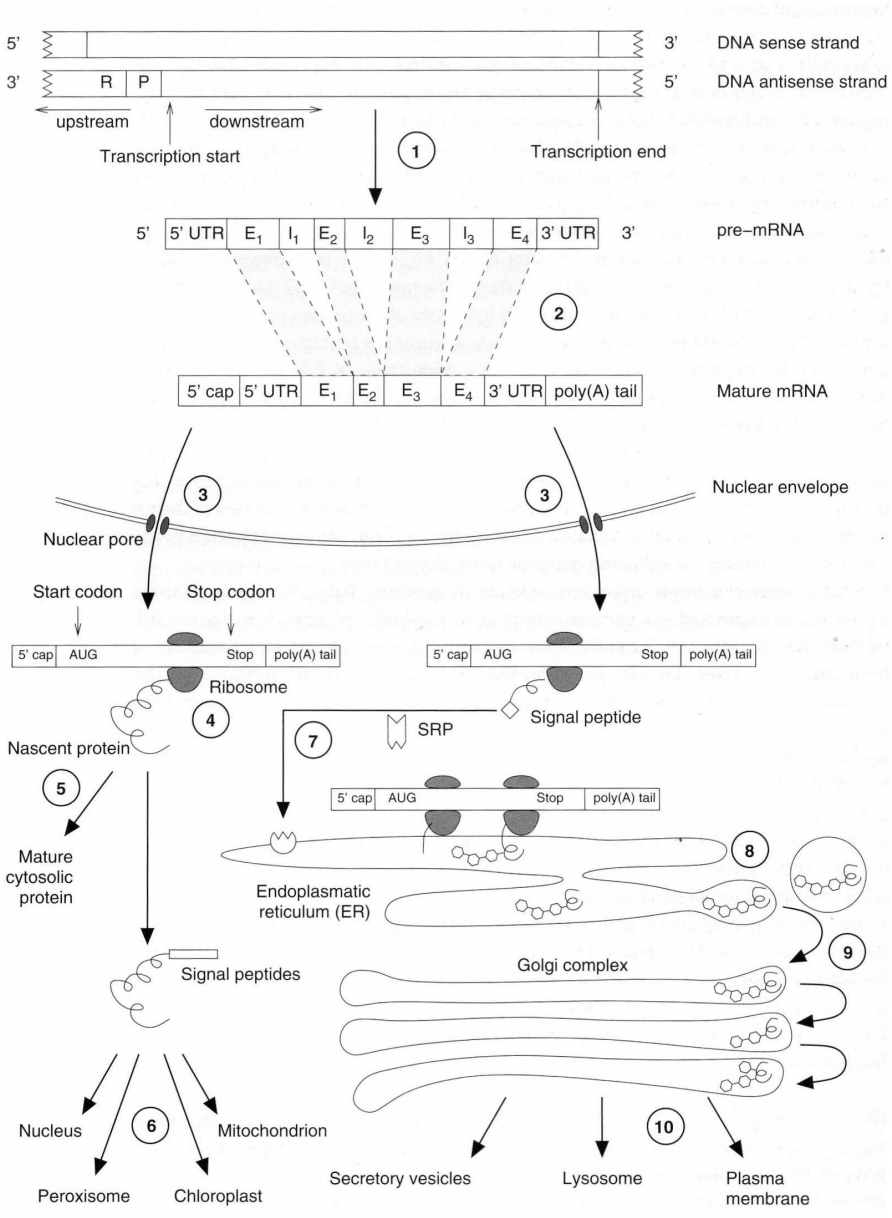


Fig. 1.2: Dogme central de la biologie moléculaire – L'ADN est d'abord transcrit en plusieurs ARNm. Ces molécules seront alors traduites pour former les différentes protéines. Tirée de [Klipp et al., 2005]

formation depuis l'ADN aux protéines. Il faut tout d'abord que l'ADN soit capable d'auto-réplication. Cette condition est le premier postulat du dogme et repose sur la déduction du caractère complémentaire du double brin d'ADN par Watson et Crick. On a ensuite démontré le caractère semi-conservatif de la réplication de l'ADN. En effet, chacune des molécules filles hérite d'un brin de l'ADN parental. Une réplication semi-conservative à l'avantage de permettre la réparation des erreurs commises lors de la réplication.

Le dogme central établit ensuite que l'ARN est l'intermédiaire entre l'ADN, porteur de l'information, et les protéines, les agents fonctionnels. La transcription est la synthèse d'une molécule d'ARN complémentaire à une séquences d'ADN. Les ANR messagers sont destinés à être traduits en protéines dans le cytoplasme, les ARN de transfert et les ARN ribosomiques sont impliqués dans le phénomène de traduction (défini ci-dessous). L'ARN subit une série de modifications avant de devenir une molécule fonctionelle. Nous nous concentrerons ici sur les ARN messagers.

La traduction des ARN messagers en protéines fait intervenir une machinerie complexe comprenant de nombreuses protéines, ARN de tranfert et ribosomiques. La traduction s'effectue en fonction du code génétique, chaque triplet de nucléotides (codon) est traduit en un unique acide aminé. Les acides aminés codés sont au nombre de 20. L'ARN messager est traduit entre un codon d'initiation et un codon de terminaison, cet intervalle définissant la phase ouverte de lecture. La protéine finalement synthétisée est strictement colinéaire à l'ARN messager.

Le dogme central dans sa version la plus dogmatique (ADN → ARN → protéine) correspond à la formulation originale de Francis Crick. Depuis, un certain nombre de phénomènes contredisant ce formalisme ont été décrits. Parmi ces phénomènes, on peut citer :

- **La transcription inverse**. Le génome des rétrovirus est constitué d'ARN, qui est reverse-transcrit en ADN après infection et va s'intégrer dans le génome de l'hôte.
- **La réplication de l'ARN**. Ceci est une caractéristique de la plupart des virus à ARN, qui produisent leurs propres enzymes pour répliquer leurs génomes.
- **La pseudo-réplication des prions**. Les prions sont des agents infectieux avec des caractéristiques similaires aux virus, mais qui ne contiennent pas d'acides nucléiques.

Pour plus de compléments sur ces informations, ces concepts ont été longuement décrits dans de nombreux manuels ou cours, parmi lesquels un cours intitulé "Réseaux cellulaires" du professeur Daniel MANGE duquel ont été extraites ces notions.

Les techniques les plus utilisées en biologie moléculaire sont : l'électrophorèse, l'extraction d'ADN d'une cellule, le clônage, la mutagénèse dirigée, l'amplification d'ADN et plus récemment le séquençage d'ADN. Ces techniques ne seront pas décrites dans ce manuscrit. Le séquençage est une technique permettant de définir l'ordre des nucléotides d'une molécule d'ADN (du simple fragment au génome complet). Grâce à cette méthodologie, un grand nombre de génomes complets d'organismes ont été séquencés. Cette avancée a permis un changement d'échelle de l'étude des gènes intra et inter-espèces. On parle maintenant de génomique et de génomique comparative.

1.1.2 La génomique comparative

Les projets de séquençage à grande échelle des génomes produisent de grandes quantités de données pour une multitude d'organismes. Cette accumulation de données implique aussi une augmentation des besoins de stockage, de gestion et d'analyse. Ces données sont stockées dans des banques accessibles à l'ensemble de la communauté scientifique. Genbank, maintenue par le National Center for Biotechnology Information (NCBI), a atteint en avril 200 la barre des 130 Giga nucléotides. On estime que le niveau d'un Tera nucléotides sera atteint avant 2010. On recense actuellement plus de 600 organismes entièrement séquencés et un grand nombre de génomes en cours de décryptage.

La génomique introduit donc de nouveaux problèmes à résoudre, tout d'abord au niveau intra-organisme : le flux d'information contenue dans l'ADN est utilisé par la cellule pour fabriquer ces protéines. Une des problématique est de déterminer la régulation impliquée afin d'obtenir un phénotype bien défini. Autrement dit, comment est régulée l'expression des gènes ? La question de comparaison se pose également au niveau inter-espèces. Les espèces actuelles partagent des ancêtres communs. En étudiant, les génomes d'espèces différentes, on a pu observer une conservation de leur séquence, voire de l'ordre des gènes. Grâce à la génomique comparative, les connaissances acquises pour un organisme peuvent donner des pistes d'études ou être appliquées à d'autres organismes.

Les données de séquençage produites n'ont que peu d'intérêt en tant que telles ; l'important est d'en extraire les connaissances biologiques. Le rôle principal de la génomique est de stocker, gérer, analyser ces données hétérogènes. Nous avons choisi de présenter ici trois approches de génomique comparative permettant d'analyser des données de séquences ainsi que des données d'expression :

- l'analyse du transcriptome. Celui-ci est constitué par l'ensemble des ARN messagers présents dans une cellule dans une situation donnée ;
- l'analyse du protéome, qui est contitué de l'ensemble des protéines, produits de la traduction des ARNm.
- l'analyse phylogénétique basée l'étude de la formation et de l'évolution des organismes vivants en vue d'établir leur parenté.

Leur finalité commune est la prédiction, l'identification de fonctions, de structures, grâce à la comparaison entre différents états biologiques ou différents organismes.

Analyse de transcriptome

L'analyse du transcriptome a pour but d'identifier et quantifier la sur- ou sous-expression d'un ensemble de gènes dans une situation biologique donnée par rapport à la situation "norma-

le". Cette étude est possible grâce à la technique des puces à ADN (ou microarrays). L'analyse d'une importante quantité de données d'expériences sur puces peut permettre d'identifier des familles et des réseaux fonctionnels de gènes mis en jeu sous l'effet de la condition étudiée. Elles permettent la mesure simultanée du niveau d'expression de plusieurs milliers de gènes voire d'un génome entier.

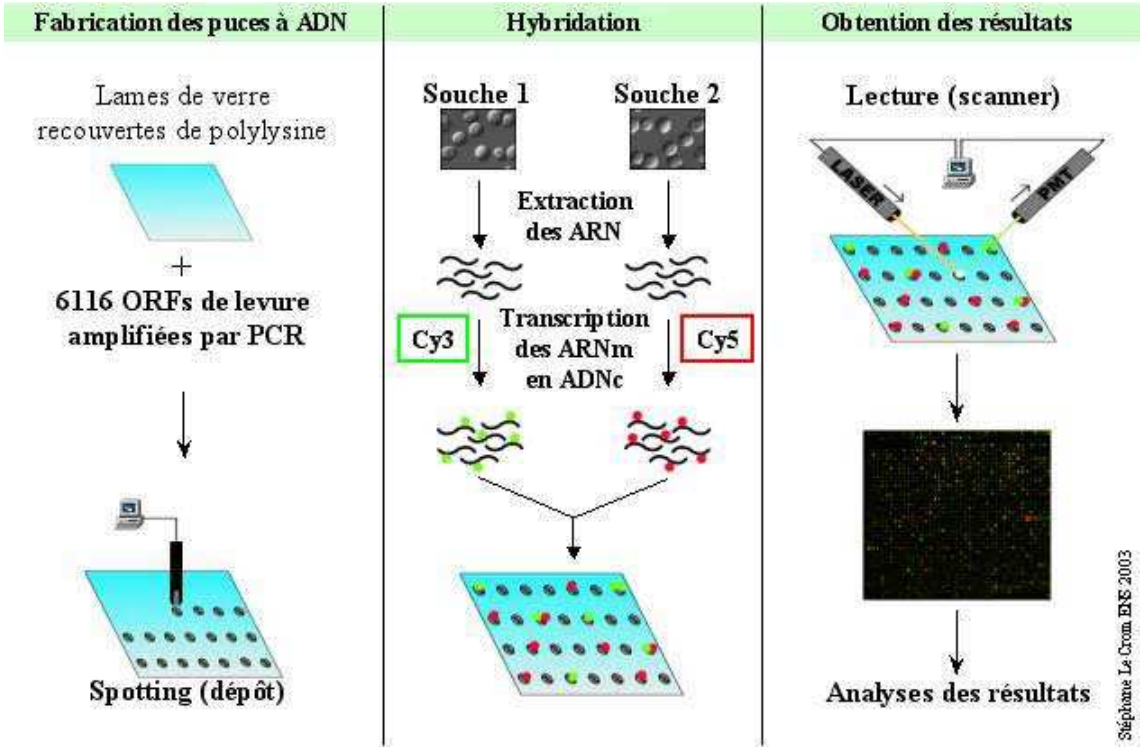


Fig. 1.3: Schéma représentant les différentes étapes de l'analyse de transcriptome par puce à ADN. – Figure tirée d'un transparent de cours de l'ENS, 2003.

Elles consistent, par exemple, en un support solide (lame de verre) sur lequel des milliers de fragments d'ADN sont déposés (sondes). Après extraction des ARNm des deux types de cellules à étudier, les ADN complémentaires (ADNc) sont obtenus par transcription inverse et sont marqués à l'aide de fluorochromes ; chacune des conditions étant marquées différemment. Les ADNc sont alors déposés sur la puce et il y a hybridation en cas de complémentarité. La fixation des gènes des cellules sur les sondes est étudiée. L'intensité du signal va permettre de quantifier l'expression des gènes. Toutes ces étapes sont schématisées sur la figure 1.3. On obtient alors une image telle que la figure 1.4 page précédente. Pour permettre d'analyser une telle quantité d'information, les méthodes de classification telles que la classification hiérarchique sont utilisées. Il est à noter que de nombreuses autres techniques statistiques sont appliquées lors de l'analyse de transcriptome (normalisation, réduction du bruit...).

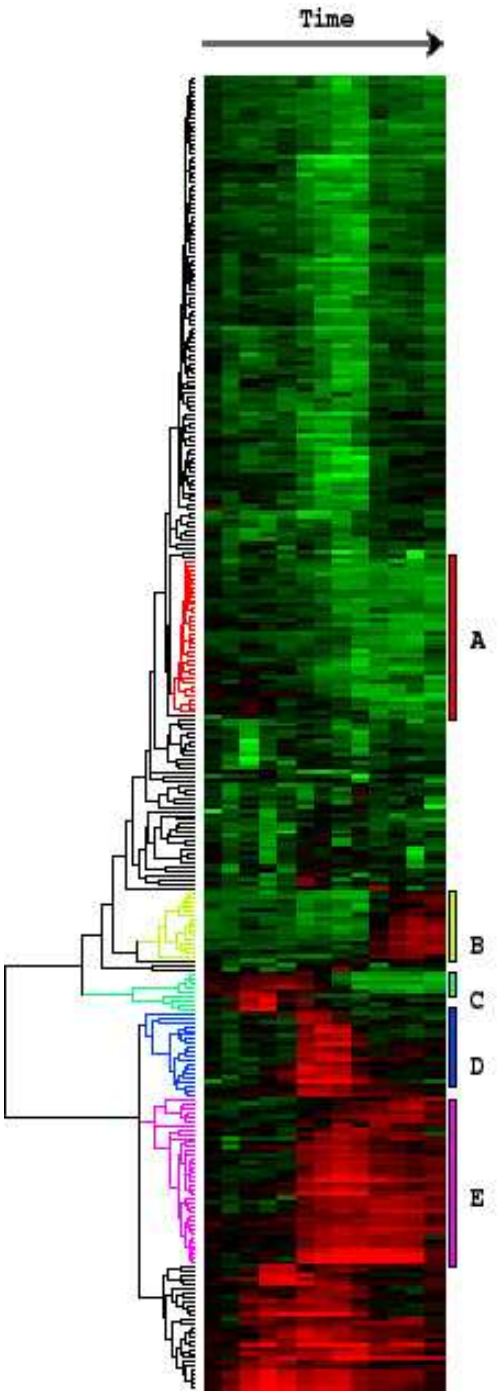


Fig. 1.4: Résultat d'une analyse de transcriptome par puce à ADN. – Verticalement, les différents gènes ont été ordonnés en fonction de la similarité de leurs profils d'expression. Horizontalement, les différentes conditions d'analyse (temps) sont représentées. A gauche, se trouve la hiérarchie ayant permis de regrouper les profils d'expression en fonction de leur ressemblance. Figure tirée de [Eisen et al., 1998]

Analyse de protéome

Le séquençage systématique de génomes complets a permis la comparaison de protéomes (produit fonctionnel du génome) de plusieurs organismes. Ainsi est apparue "l'analyse comparative de génomes" [Koonin et al., 1997, Park and Teichmann, 1998, Codani et al., 1999] dont le but est l'étude des relations fonctionnelles entre les gènes de différentes espèces. [Codani et al., 1999, Aude, 1999] ont proposé une méthode en trois étapes :

- la comparaison de toutes les paires de séquences par l'algorithme du Z-value [Comet et al., 1999] pour obtenir un indice de dissimilarité sans biais
- le calcul d'une partition par la méthode du lien simple
- l'étude des effets de chaîne par la classification pyramidale (décrite au chapitre suivant)

Ce dernier point est fondamental puisqu'il permet de comprendre la structure en domaine (sous-unité fonctionnelle) de certains gènes [Aude et al., 1999, Comet et al., 1999]. Cette approche a depuis été utilisée pour l'étude de protéomes des végétaux [Louis et al., 2001] ou la comparaison systématique de 70 protéomes (soit 240000 séquences) pour le projet Teraprot [Teraprot, 2002].

Sur la figure 1.5 sont décrites les différentes étapes menant à la construction des familles d'homologues.

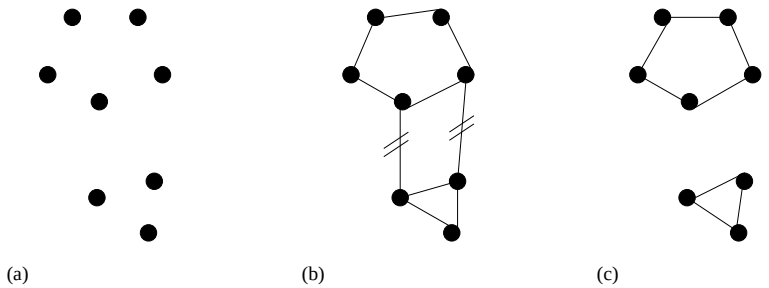


Fig. 1.5: Analyse comparative de protéome. – (a) Les séquences protéiques sont représentées par des points. (b) A l'issue de la comparaison des séquences, on peut représenter les distances entre les séquences par des arcs valués entre celles-ci. (c) Construction des familles de séquences homologues, le seuil α ayant été appliqué.

La première étape de l'analyse comparative est d'obtenir une mesure de la similarité, de la ressemblance entre chaque paire de séquences à analyser. Pour cela, la méthode la plus utilisée est de réaliser un alignement de ces séquences (décrit chapitre 3 page 95). Deux techniques utilisées Smith & Waterman [Smith and Waterman, 1981] et BLAST sont décrites en détail dans ce chapitre. La Z-value est moins sensible au biais de longueur et de composition, par rapport aux autres méthodes de comparaison de séquences. Elle est donc plus adaptée à la constitution d'un indice de dissimilarité global entre des séquences.

Ensuite, afin de construire une famille d'homologues, on définit un seuil α de similarité. Si l'indice de similarité (la Z-value, par exemple) entre deux séquences est inférieur à ce seuil, celles-ci ne sont pas considérées comme homologues.

Sur la figure 1.6, sont représentées des protéines multi-domaines formant une famille de séquences homologues. Cet exemple, représentant un cas complexe de données, sera utilisé à plusieurs reprises dans ce mémoire. Toutes les méthodes de classification menant à une représentation sous forme d'arbre ne permettent pas de traiter de telles données. En effet, les séquences peuvent appartenir à plusieurs classes, lorsqu'elles possèdent plusieurs domaines 1.7 page suivante. Nous verrons en détail, dans le chapitre 2 page 43, l'apport des pyramides pour interpréter ces données par rapport aux hiérarchies. Puis, une étude approfondie de cette famille sera réalisée dans le dernier chapitre de ce manuscrit.

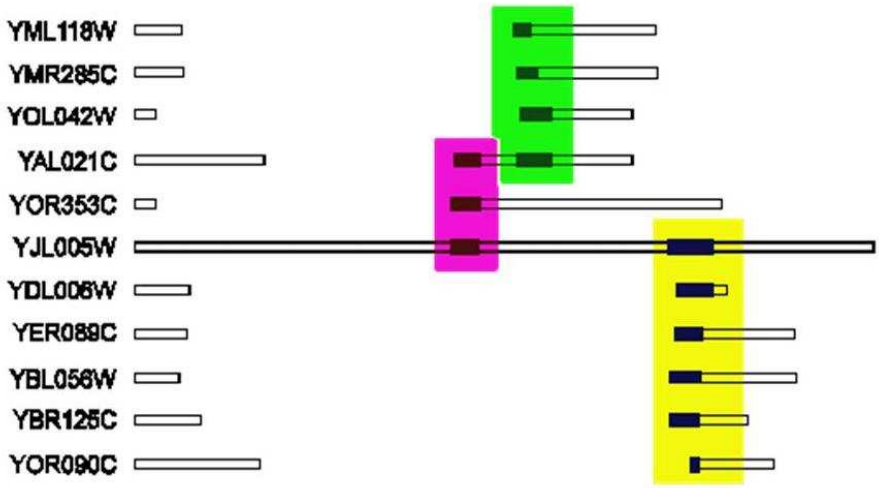


Fig. 1.6: Alignement d'une famille de protéines homologues multi-domaines. 3 types de domaines protéiques sont identifiables

Analyse phylogénétique

L'analyse phylogénétique a pour objectif de reconstruire des liens de parenté entre les organismes et d'estimer leurs temps de divergence. Depuis les travaux de Darwin, l'arbre est le support formel de la représentation du processus dynamique de la diversification des espèces : les feuilles représentent les espèces, les noeuds internes les ancêtres ayant divergé, et les arêtes les liens de filiation. Le problème posé est de retrouver l'arbre phylogénétique : c'est à dire de reconstruire l'histoire des spéciations, reconstruire l'histoire des duplications de séquences... Pour réaliser une phylogénie entre plusieurs espèces, il faut choisir une séquence ayant un homologue dans chacune des espèces étudiées. Ensuite, l'étape préliminaire de toute étude phylogénétique est l'alignement de ces séquences. L'arbre est alors reconstruit à l'aide d'une des méthodes que

nous allons décrire ci-dessous : parcimonie, maximum de vraisemblance ou encore de distances. Un arbre non-enraciné est une représentation intemporelle des relations phylogénétiques alors qu'un arbre enraciné spécifie où se situe l'ancêtre commun de tous les taxons présents dans l'arbre.

Les méthodes de parcimonie consistent en rechercher l'arbre le plus court en terme du nombre de mutations. Ces méthodes reposent sur le principe du même nom : le principe de parcimonie découle de la philosophie voulant que la nature soit parcimonieuse et que l'explication la plus simple soit souvent la meilleure. La phylogénie la plus vraisemblable est donc celle qui nécessite le plus petit nombre de changement évolutifs.

La vraisemblance est la probabilité d'observer les données sous une hypothèse donnée. La démarche consiste donc à rechercher la vraisemblance des données sous différentes hypothèses évolutives d'un modèle donné et à retenir les hypothèses qui rendent cette vraisemblance maximale. Dans ce cas, les données sont les séquences comparées et l'hypothèse est l'arbre. On recherche donc, étant donné un modèle d'évolution, l'arbre maximisant la probabilité d'obtenir les séquences actuelles. Le maximum de vraisemblance est très utilisée pour calculer les arbre phylogénétiques, car elle est fiable. En contre-partie, du fait des calculs combinatoires, elle est beaucoup plus coûteuse en temps. Cette méthode est moins sensible que la parcimonie à l'attraction des longues branches. Ce phénomène est particulièrement problématique et révèle l'inconsistance des méthodes de reconstruction. Plus il y a de données à analyser, plus la méthode de reconstruction va converger vers l'arbre faux. En règle générale, les espèces qui émergent tôt dans un arbre phylogénétique sont souvent des espèces qui sont mal placées en raison de l'artéfact

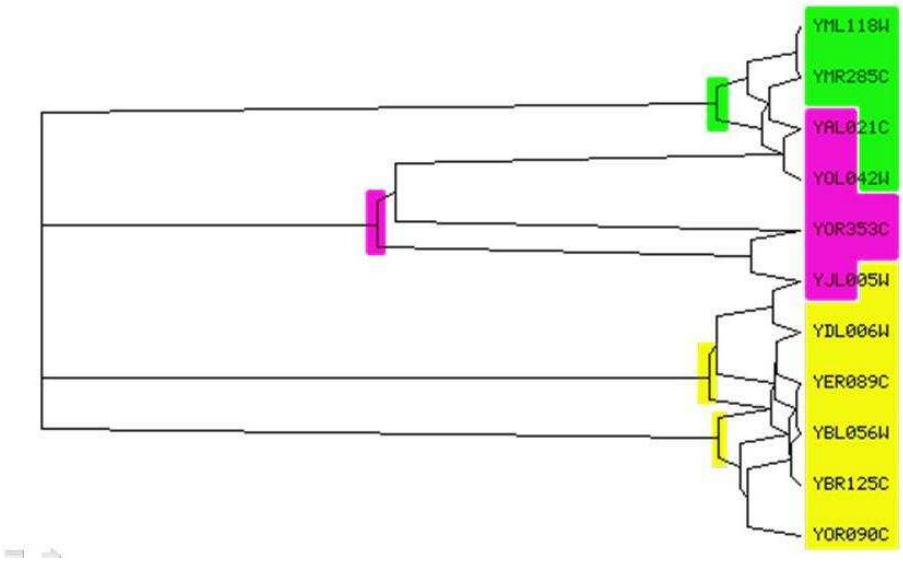


Fig. 1.7: Pyramide représentant une famille de protéines homologues.
Les trois domaines sont mis en évidence par la classification pyramidale

de l'attraction des longues branches.

Les méthodes de distance se proposent de reconstruire des arbres en partant des ressemblances observées entre chaque paire de séquences étudiées. Une étape critique pour ce type de méthodes est le calcul de la distance entre les séquences étudiées. De nombreux modèles existent : une fois la matrice de distance construite, il faut alors trouver l'arbre (la topologie et la longueur des branches) représentant le mieux cette matrice.

La figure 1.8 représente un arbre phylogénétique non-enraciné, issu d'une analyse phylogénétique que nous avons réalisé pour le CEA. Le projet avait pour but de montrer (ou non) un lien entre la distance entre les génomes étudiés et la proximité géographique des cavernes. Les données réalisées pour obtenir cet arbre sont des séquences d'ADN mitochondrial d'ours des cavernes. Les noms donnés à chaque séquence correspondent à la grotte dans laquelle a été prélevé l'échantillon.

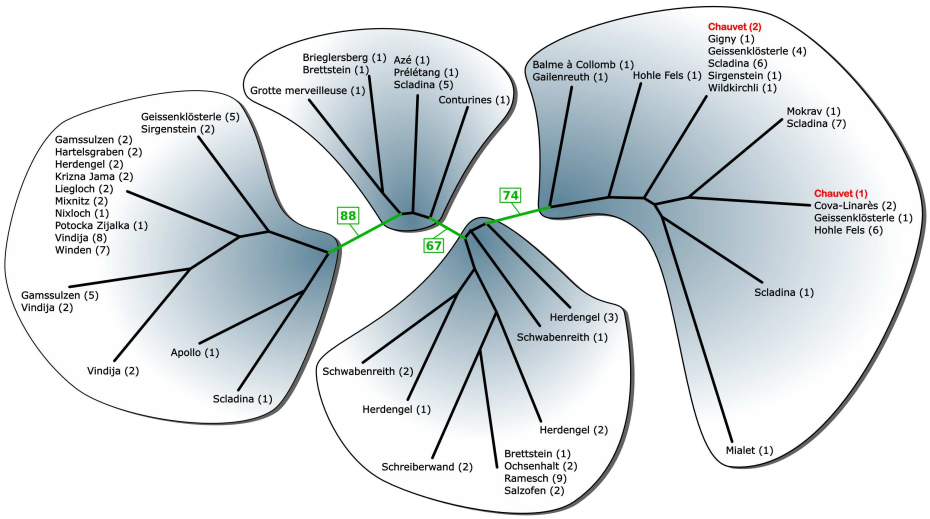


Fig. 1.8: Exemple d'arbre phylogénétique non-enraciné Etude réalisée à partir d'ADN mitochondrial d'ours des cavernes

Ces méthodes ont des propriétés différentes (rapidité, simplicité, précision), elles sont donc à utiliser conjointement. De plus, afin d'évaluer la robustesse de l'arbre, il existe plusieurs techniques comme le bootstrap : un arbre qui varie lorsque l'on introduit un peu de bruit ne peut être considéré comme fiable.

La génomique comparative utilise les méthodes de phylogénie moléculaire appliquées aux génomes complets pour mettre en évidence les liens de parenté entre différentes espèces.

1.1.3 Les disciplines dérivées

Nous avons vu précedemment que, suite à la biologie moléculaire et à l'appartion de la technique de séquençage, la gén"-omique" est apparue. Avec l'utilisation des techniques globales d'étude des génomes, les "-omiques" se multiplient. On citera notamment : la génomique, la protéomique, la transcriptomique, la pharmacogénomique... Ces technologies "omiques" génèrent, stockent et analysent d'énormes quantités de données biologiques, ce qui serait impossible sans méthodes informatiques et mathématiques. Devant l'importance des données, chaque nouvelle discipline a développé des méthodes d'analyses propres.

A présent, en plus de ces analyses ciblées, on retourne vers un traitement global des données grâce à l'accroissement de moyens de calcul importants et au développement des méthodes de traitement de données hétérogènes et volumineuses. La discipline de recherche émergente est la biologie intégrative, dont le but est de traiter un des données de sources hétérogènes (génome, transcriptome...). Dans un article de synthèse consacré à la bioéthique, Anne Cambon-Thomsen recense et explique qu' "Une revue scientifique consacrée à la biologie intégrative "omics" et des glossaires, s'attachant particulièrement aux termes en "ome" et "omique", ont fait leur apparition. Ce changement de vocabulaire marque la distinction faite entre la visée globale de l'approche scientifique désignée et de ses technologies afférentes et les approches ponctuelles précédentes."

Nous venons de présenter différents domaines dans lesquels l'analyse de données biologiques nécessite une représentation sous forme d'arbres. Les méthodes que nous allons développer dans la suite de ce chapitre peuvent traiter tous ces types de données et des applications seront vues tout au long de ce document. Auparavant, nous définissons les données et notions nécessaires à la compréhension des méthodes de construction d'arbre.

1.2 Définitions sur les distances et matrices

La représentation des données observées est la première étape de l'analyse de ces données. Celle-ci doit être conforme et la plus proche possible des données de départ et peut être obtenue après un pré-traitement (élimination de bruit, centrage...). Pour procéder à la classification d'un ensemble de données, nous devons au préalable évaluer la "distance" qui sépare ces données. Nous rappelons ici des définitions qui seront utilisées ensuite.

Définition 1.1 (Dissimilarité) *Une mesure de dissimilarité D, définie sur Ω, est une appli-*

cation de $\Omega \times \Omega$ dans $\mathbb{R}^+$ satifaisant les conditions suivantes :

- $\forall x \in \Omega \ : D(x,x) = 0$
- $\forall x, y \in \Omega \ : D(x,y) = D(y,x)$

A partir de données observées, il existe différentes méthodes pour calculer des mesures de dissimilarité. Ces méthodes ne seront pas présentées ici. Nous allons définir à présent la notion de distance qui est un cas particulier de dissimilarité.

Définition 1.2 (Distance) *Une distance sur un ensemble Ω est une application d de $\Omega \times \Omega$ sur $\mathbb{R}^+$ telle que :*

- $\forall x \in \Omega \ : d(x,x) = 0$
- $\forall x, y \in \Omega \ : d(x,y) = d(y,x)$
- $\forall x, y, z \in \Omega : d(x,z) \leq d(x,y) + d(y,z)$ *(inégalité triangulaire)*

Un ensemble muni d'une distance s'appelle un espace métrique.

La distance est dite *ultramétrique* si elle vérifie la condition suivante, plus forte que l'inégalité triangulaire : $\forall x, y, z \in \Omega : d(x,z) \leq max(d(x,y), d(y,z))$.

Dans le cas ultramétrique, trois objets x, y, z forment toujours un triangle isocèle, les deux côté égaux étant plus grands que le troisième ; une distance ultramétrique est forcément métrique.

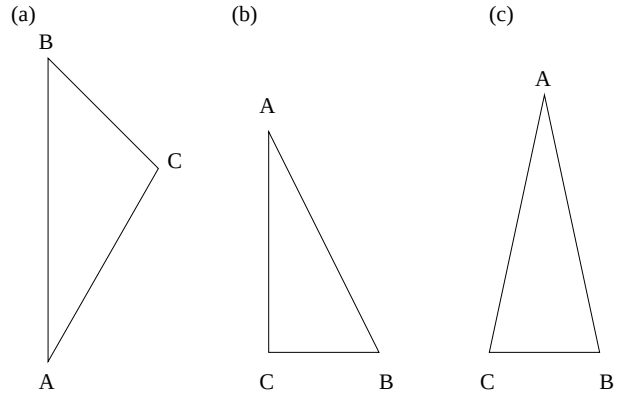


Fig. 1.9: Situations non-ultramétriques et ultramétrique

La situation décrite par la figure 1.9 (a) n'est pas ultramétrique : $d(A,B) > max(d(A,C), d(B,C)) = d(A,C)$. La situation décrite par (b) n'est pas ultramétrique non plus : $d(A,B) > max(d(A,C), d(B,C)) = d(A,C)$. La figure (c) réalise une situation ultramétrique : $d(A,B) \leq max(d(A,C), d(B,C)) = d(A,C)$, ainsi que $d(A,C) \leq max(d(A,B), d(B,C)) = d(A,C)$ et $d(B,C) \leq max(d(A,C), d(A,B))$.

On dit qu'une distance vérifie la **condition des quatres points** si pour tout x, y, z, t deux des trois grandeurs suivantes sont égales ou supérieures à la troisième :

- $d(x,y) + d(z,t)$

- $d(x,t) + d(z,y)$
- $d(x,z) + d(t,y)$

Définition 1.3 (Dissimilarité de Robinson) *Une dissimilarité d est dite de Robinson, si et seulement s'il existe un ordre total Θ sur Ω tel que pour tout $x, y, z \in \Omega$*

$$x\Theta y\Theta z \Rightarrow max\{d(x,y), d(y,z)\} \leq d(x,z)$$

Un ordre qui satisfait la condition ci-dessus est dit compatible avec d.

Afin de stocker la globalité des informations sur les données observés, le format matriciel est utilisé. Les données à analyser par une méthode de construction d'arbre sont généralement décrites par un de ces deux formats :

1. La matrice des profils ou des modalités, $X = (x_{ik})$. Cette matrice est de taille $n \times p$ où x_{ik} correspond à la valeur prise par la $k^{\text{ième}}$ variable $(k = 1, \ldots p)$ pour le $i^{\text{ième}}$ individu $(i = 1, \ldots n)$. Chaque ligne de la matrice constitue ainsi le profil, ou l'ensemble des modalités du $i^{\text{ième}}$ individu.
2. La matrice de dissimilarité $D = (d_{ij})$, ou la matrice de similarité $S = (s_{ij})$. Ces matrices de taille $n \times n$ sont à valeurs dans $\mathbb{R}^+$. Chaque valeur d_{ij} (*resp.* s_{ij}) de la matrice caractérise le degré de dissimilarité (*resp.* similarité) entre les deux individus i et j, où $(i, j = 1, \ldots n)$.

Après avoir défini une dissimilarité de Robinson, ainsi que la représentation sous forme de matrice des données, nous introduisons le terme de "Matrice de Robinson" qui sera très utilisé dans la suite de ce mansucrit.

Définition 1.4 (Matrice de Robinson) *On appelle matrice de Robinson, une matrice dont les éléments sont croissants en ligne et en colonne depuis la diagonale principale.*

Ces définitions donnent le point de départ et le cadre nécessaires aux méthodes traitant des données de distance entre objets. Dans la partie suivante, nous allons décrire certaines de ces méthodes aboutissant à une représentation arborée des relations entre ces objets.

1.3 Les méthodes de construction d'arbre

Nous allons exposer de façon formelle et algorithmique une sélection de méthodes de construction d'arbre (classification, reconstruction phylogénétique). Ces techniques, bien que partant toutes d'une matrice de distance entre les objets étudiés, donnnent des résultats de nature différente et offrent diverses interprétations. Nous allons mettre en avant les différences entre ces approches. Pour cela, nous présentons d'abord les méthodes de classification destinées à

établir les relations de proximité dans un ensemble d'objets avant de détailler plusieurs techniques de reconstruction phylogénétique, basées sur le principe d'évolution minimum.

1.3.1 Les méthodes de classification

La plupart des méthodes de classification visent à étudier un ensemble de données afin d'établir l'existence de similarité ou non entre les données. Elles permettent de représenter de façon synthétique les ensembles de données. En outre, elles permettent également un traitement automatique de l'information des ensembles de données volumineux. Ces méthodes ont été appliquées avec succès dans plusieurs domaines autres que la biologie : psychologie, physique, sciences cognitives... Les méthodes de classification découvrent des relations de proximité dans un ensemble d'objets et les représentent graphiquement. Par construction, les objets ainsi réunis tendent à former des classes homogènes.

La classification automatique peut être :
- supervisée : les classes sont connues a priori, leurs propriétés associées également.
- non-supervisée : les classes sont fondées sur la nature des objets. Les propriétés associées aux classes sont plus difficiles à déterminer.

Dans ce document, nous nous plaçons dans le cas de la classification automatique non-supervisée qui regroupe deux méthodes opposés : hiérarchique et non-hiérarchique. La classification hiérarchique [Gordon, 1996a] est une des plus fréquemment utilisées du fait de son efficacité et de la simplicité d'interprétation des résultats obtenus. Elle est décrite dans la section suivante.

La classification hiérarchique [Gordon, 1996a]

De nombreuses méthodes de classification sont basées sur des algorithmes agglomératifs dont l'enjeu est d'obtenir par étapes successives une seule partition à partir d'un ensemble d'objets.

La représentation d'une classification hiérarchique est un dendrogramme (figure 1.10 page suivante). Les objets ou individus, A, B, C, D, E, sont les éléments terminaux de l'arbre. Les classes F, G, H, I sont les noeuds de l'arbre : ce sont des classes issues de regroupements de deux éléments (objet ou classe) dont chacun détermine une nouvelle partition. La hiérarchie de la figure 1.10 page suivante est indicée par les valeurs des distances correspondant à chaque étape d'agrégation. L'indice est la distance déterminant le regroupement. Considérant une hiérarchie indicée, nous pouvons déduire un autre indice, appelé dissimilarité induite. Il définit le plus petit niveau de regroupement de deux objets dans la hiérarchie indicée et caractérise la dissimilarité entre ces objets déduite de la représentation hiérarchique.

Notations et définitions Les notations suivantes sont utilisées pour permettre une présentation formelle. Soient Ω un ensemble fini d'objets et S un sous-ensemble de $P(\Omega)\backslash\{\emptyset\}$. Quelques assertions utiles peuvent être formulées en utilisant le formalisme introduit par [Gaul and Schader, 1994] :

$$\Omega \in S \tag{1.1}$$

$$\forall i \in \Omega : \{i\} \in S \tag{1.2}$$

$$\forall K, L \in S : K \cap L \in \{\emptyset, K, L\} \tag{1.3}$$

$$\forall K, L \in S : K \cap L = \emptyset \text{ or } K \cap L \in S \tag{1.4}$$

$$\begin{array}{l}\text{Il existe un ordre total } \Theta \text{ sur } \Omega \\ \text{tel que chaque sous-ensemble de } S \text{ est un intervalle de cet ordre}\end{array} \tag{1.5}$$

Etant données ces conditions, nous allons pouvoir définir les structures hiérarchiques.

Définition 1.5 (Hiérarchie) *Un sous-ensemble H de $P(\Omega)\backslash\{\emptyset\}$ est une hiérarchie sur Ω s'il vérifie les conditions* (1.1)*,* (1.2)*, et* (1.3)*.*

Nous définissons une fonction indicée pour représenter l'homogénéité des classes. Celle-ci représente la proximité entre les classes et permet ainsi de valuer une hiérarchie. La caractérisation de cette fonction, notée f, nécessite les conditions suivantes :

$$\forall K \in S : f(K) = 0 \Leftrightarrow |K| = 1 \tag{1.6}$$

$$\forall (K, L) \in S^2 : K \subseteq L \Rightarrow f(K) \leq f(L) \tag{1.7}$$

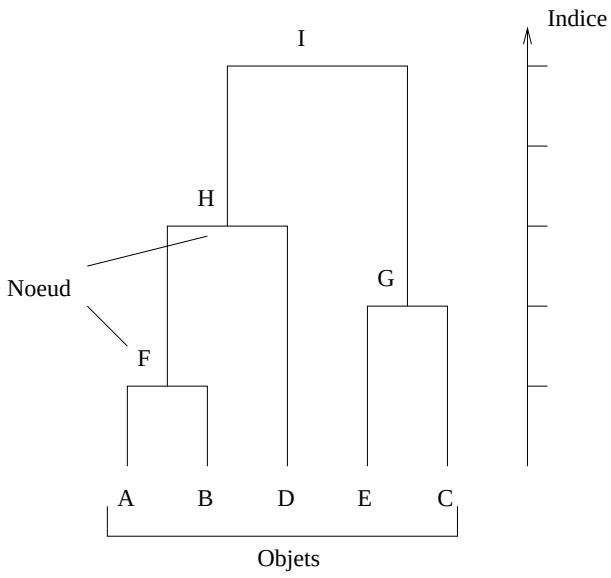


Fig. 1.10: Arbre hiérarchique ou dendrogramme – Eléments de vocabulaire.

En utilisant ce contexte, nous pouvons alors associer une fonction indicée à une hiérarchie comme défini ci-dessous.

Définition 1.6 (Hiérarchie indicée) (S, f) *est une hiérarchie indicée sur* Ω *si* S *est une hiérarchie et* f *satisfait les conditions* (1.6) *et* (1.7)*.*

Il est équivalent de munir un ensemble fini Ω d'une ultramétrique ou de définir une hiérarchie indicée de parties de cet ensemble. En effet, toute hiérarchie indicée permet de définir une distance entre objets ayant les propriétés requises, appelée dissimilarité induite. De nombreuses caractérisations d'une hiérarchie indicée ou dendrogramme ont été présentées.

Définition 1.7 (Dissimilarité induite) *Soit* (H, f) *est une hiérarchie indicée. On note* h *la dissimilarité induite par* H *la fonction à valeurs dans* $\mathbb{R}^+$ *satisfaisant les conditions suivantes :*

- $h_{mn} = 0 \Leftrightarrow m = n$
- $\forall m, n \in \Omega x \Omega, h_{mn} = f(p)$ *où il existe* $p \in H | m \in p, n \in p$ *et il n'existe pas* $p' \in H | p' \subseteq p, m \subseteq p', n \subseteq p'$

Algorithmes Il existe un grand nombre de façons d'obtenir un arbre à partir d'un ensemble d'objets décrit dans une matrice. Ces façons sont catégorisées par le type d'algorithmes mis en place pour calculer l'arbre : agglomératif, divisif, incrémental, optimisation directe, parallèle... Le principe des algorithmes divisifs est le contraire des algorithmes agglomératifs décrits dans la section suivante. Initialement, tous les objets appartiennent à une seule classe. A chaque étape, le nombre de classes augmente, une des classes existantes étant divisée en deux classes homogènes. L'algorithme s'arrête lorsque chaque objet constitue une classe. Les autres types d'algorithmes ne seront pas décrits ici.

La classification ascendante hiérarchique

La classification ascendante hiérarchique (CAH) a pour principe de créer à chaque étape une partition obtenue en agrégeant 2 à 2 les éléments (individus ou groupe d'individus) les plus proches. L'algorithme fournit une hiérarchie de partitions, c'est à dire un arbre contenant l'historique de la classification. Il est nécessaire de se munir d'une dissimilarité pour définir les dissemblances entre les objets et de fixer une règle pour agréger un individu et un groupe d'individus.

Critères d'agrégation Pour définir l'agrégation entre objets, nous allons avoir besoin de définir une dissimilarité entre classes d'objets. Les critères d'agrégation sont les règles de calcul de ces dissimilarités et sont à choisir pour dérouler l'algorithme.

Soient i, j, k trois objets de poids n_i, n_j, n_k et C une classe regroupant i et j.

Plusieurs critères existent :

- la *critère du lien simple* (ou du saut minimal) :
  $d(C,k) = min(d(i,k), d(j,k))$.
- la *critère du lien complet* (ou du saut maximal) :
  $d(C,k) = max(d(i,k), d(j,k))$.
- la *critère de la moyenne* :
  $d(C,k) = \frac{d(i,k)+d(j,k)}{2}$.
- la *critère de la moyenne pondérée* :
  $d(C,k) = \frac{n_i d(i,k)+n_j d(j,k)}{n_i+n_j}$.
- la *critère de Ward* :
  $d(C,k) = \frac{(n_i+n_k)*d(i,k)+(n_j+n_k)*d(j,k)-n_k*d(i,j)}{n_i+n_j+n_k}$.

Le critère du lien simple a tendance à agréger les classes qui ont une propriété quelconque en commun.

Le critère du lien complet minimise la distance à l'intérieur d'une classe.

A partir d'un nuage étiré, le critère du lien simple tendra à ajouter les individus un à un au cluster déjà formé, favorisant ainsi l'effet de chaîne. Au contraire, celui du lien complet tendra à former des sous-sous-classes de taille similaire qui seront regroupés dans des sous-classes de taille similaire, etc... Le dendrogramme sera homogène. Les deux critères de la moyenne offrent de bons compromis par rapport à ces deux critères.

Le critère de Ward est basé sur la décomposition inter/intra groupe de l'inertie. Cela revient à minimiser l'inertie intra-classe ou encore à maximiser l'inertie inter-classe.

Algorithme Soit Ω l'ensemble des individus, et μ un critère d'agrégation (voir paragraphe précédent). L'algorithme fondalemental de la CAH est le suivant :

- **Initialisation :** Les seuls paliers existants sont les singletons de Ω.
- **Agrégation :** Deux paliers p^* et q^* sont agrégés si l'indice d'agrégation $\mu(p^*, q^*)$ est minimum parmi l'ensemble des indices d'agrégation $\mu(p,q)$ où $(p,q) \in P \times P$.
- **Test d'arrêt :** L'algorithme prend fin quand le palier que l'on vient de créer, $(p^* \cup q^*)$, est égal à Ω. Dans le cas inverse on reprend l'exécution de l'algorithme à partir de la phase d'agrégation.

Remarque : La méthode UPGMA (Unweight Pair Group Method with Arithmetic mean) utilsée en phylogénie pour calculer un arbre est identique à la cah avec le critère de la moyenne pondérée. C'est pourquoi nous n'en parlerons pas spécifiquement.

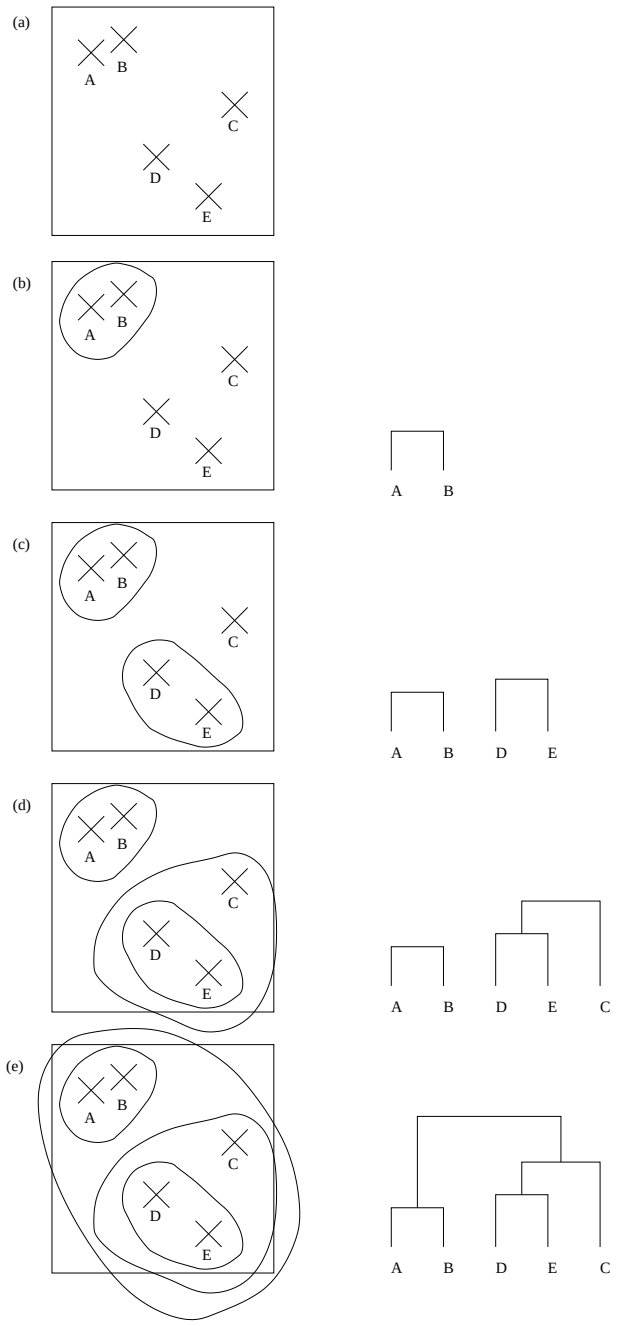


Fig. 1.11: Construction d'une hiérarchie – Explication de la cah. (Initialisation) Les dissimilarités entre les objets sont stockées dans une matrice carrée. (Agrégation) Les objets les plus proches (A et B) sont repérés et agrégés. Un nouveau noeud interne est créé : F. (Mise à jour) Le nouveau noeud remplace les objets qu'il contient. Les nouvelles dissimilarité sont calculées en fonction du critère d'agrégation (minimum, maximum, moyenne). (Fin) Hiérarchies.

Les méthodes décrites dans les sections suivantes reposent sur le principe d'évolution minimum, qui existe depuis le 19ème siècle dans ce que l'on appelle les arbres de Steiner.

1.3.2 Les méthodes de reconstruction phylogénétique

Le principe d'évolution minimum a été proposé comme principe de base de l'inférence phylogénétique. Etant donné une matrice de distances, ce principe implique d'abord d'estimer la longueur d'une topologie donnée et alors de choisir la topologie de plus faible longueur. De nombreuses variantes de ce principe existent, dépendant de la façon d'estimer la longueur des branches et de celle dont l'arbre est calculé à partir de ces branches. La solution la plus commune est de définir la longueur de l'arbre comme la somme des longueurs de toutes les branches (positives ou négatives). Les longueurs des branches sont généralement estimées à l'aide des moindres carrés. Ci-dessous, nous décrivons les principales méthodes avec lesquelles nous allons comparer dans la suite de notre travail.

Addtree [Corter, 1996]

La méthode Addtree est un algorithme agglomératif, très générique au point de vue algorithmique. Elle donne de bons résultats dans le domaine de la reconstruction phylogénétique. Elle consiste à agréger itérativement les espèces et à constituer ainsi des paires externes. A chaque paire constituée, les deux espèces concernées sont supprimées de la matrice et sont remplacées par leur barycentre. A chaque étape, il reste donc une espèce de moins. L'algorithme s'arrête lorsqu'il ne reste plus que 3 espèces. A chaque étape, le choix de la "meilleure" paire repose sur une application directe de la "condition des quatre points" (voir la définition).

Neighbor Joining [Saitou and Nei, 1987]

L'algorithme NJ introduit par [Saitou and Nei, 1987] est la méthode la plus populaire pour reconstruire un arbre phylogénétique à partir d'une matrice de distances évolutives. L'idée principale du Neighbor Joining (NJ) est de rechercher, parmi tous les arbres celui dont la longueur est minimale. D'un point de vue algorithmique, NJ est très proche de ADDTREE. Cet algorithme suit le procédé agglomératif qui, à chaque étape, sélectionne une paire d'objets à agréger. Lors de cette agglomération, le nouveau noeud créé remplace les deux noeuds sélectionnés (voir la figure 1.12 page ci-contre), et la matrice de distance est réduite en remplaçant les distances aux deux noeuds par celle du nouveau noeud créé.

Algorithme A chaque étape, NJ utilise une matrice de distance d_{ij}, où i (respectivement j) représente soit un des objets étudiés, soit un sous-arbre contenant plusieurs objets. A partir de ces distances, deux sous-arbres sont sélectionnés et sont regroupés dans un même ensemble. Ils perdent alors leur identité et constituent alors un seul et même sous-arbre. Au départ, chaque sous-arbre est constitué d'un seul objet, et la dimension de la matrice est égale au nombre n d'objets étudiés. A chaque agglomération, deux sous-arbres sont fusionnés, réduisant ainsi le nombre de sous-arbres restant et la dimension de la matrice de distance. Si l'on note Q_{ij} la valeur du critère correspondant à l'agglomération des sous-arbres i et j (éventuellement réduits à un seul objet), alors la paire à agglomérer est celle qui minimise : $Q_{ij} = S_i + S_j - (n-2) * d_{ij}$ avec $S_x = \sum_{i=1}^{n} d_{xk}$.

Une fois que la paire (i, j) à agréger est sélectionnée, NJ crée un nouveau noeud k qui représente la racine du nouveau sous-arbre. Puis, NJ estime la longueur des branches (i,k) et (j,k) en utilisant les formules suivantes : $d_{ik} = \frac{1}{2}(d_{ij} + \frac{S_i - S_j}{n-2})$ et $d_{jk} = \frac{1}{2}(d_{ij} + \frac{S_j - S_i}{n-2})$ Enfin, NJ réduit la matrice de distances en enlevant toutes les distances associées à i ou à j, et en estimant les distances entre le noeud k et tout autre noeud P, en utilisant la formule suivante : $d_{kP} = \frac{1}{2}(d_{ik} - d_{iP}) + \frac{1}{2}(d_{jk} - d_{jP})$. Le processus continue jusqu'à ce que la matrice ne soit plus que de dimension 2, la longueur de la dernière branche étant alors égale à la dernière valeur contenue dans la matrice de distance.

NJ a une faible complexité en temps de calcul ($\mathcal{O}(n^3)$), ce qui lui permet de traiter de grands jeux de données. De nombreuses simulations informatiques ont montré que NJ est une méthode de reconstruction phylogénétique relativement fiable. Des variantes de NJ ont été développées, notamment BIONJ [Gascuel, 1997] et Weighbor [Bruno et al., 2000]. D'autres méthodes de distances, utilisant une approche différente, ont été proposées. La plus connue est la méthode des moindres carrés pondérés qui est implémenté dans le programme FITCH. Néanmoins, la topologie des arbres inférés par ces méthodes de distance est moins fiable que celle des arbres reconstruits par maximum de vraisemblance. Les variantes de NJ utilisent d'autres manières d'estimer la paire à agréger et le critère d'agrégation. Toutes ces méthodes partagent cependant le même schéma agglomératif décrit ci-dessus et sont donc de même complexité en temps. Quand

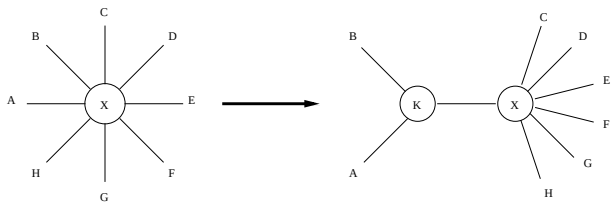


Fig. 1.12: Neighbor Joining – Les objets sont d'abord regroupés dans un arbre en étoile. Les deux objets les plus proches (A et B) sont regroupés et un nouveau noeud K est créé. Ce processus est répété jusqu'à ce que l'arbre entier soit construit

les sous-arbres i et j sont agglomérés en un nouveau sous-arbre k, les nouvelles distances d_{kP} peuvent être estimées par n'importe quelle combinaison convexe de $(d_{ik} - d_{iP})$ et de $(d_{jk} - d_{jP})$. NJ suppose implicitement que les deux estimations se valent et leur accorde le même poids $(1/2)$.

BioNJ[Gascuel, 1997] choisit ces poids de manière à obtenir l'estimateur de d_{ij} dont la variance est minimale. Les agglomérations suivantes se font donc en s'appuyant sur de meilleures estimations des distances. BioNJ est aussi rapide que NJ et améliore les performances de ce dernier. La différence entre les deux méthodes est surtout sensible lorsque l'horloge moléculaire n'est pas respectée.

Weighbor [Bruno et al., 2000] utilise un critère d'agrégation différent de celui de NJ. Ce critère prend en compte le fait que les distances les plus grandes sont les moins bien estimées. Pour cela, les estimations des distances sont modélisées par des variables normales. Cette modélisation est utilisée pour sélectionner la paire à agréger et pour réduire la matrice de distance. Weighbor est moins sensible au phénomène d'attraction des longues branches que NJ et BioNJ.

Le tableau 1.1 page suivante regroupe les différentes façons de choisir les paires d'objets à agréger et la façon de réduire la matrice de distance. La première étape consiste à déterminer l'élément i de poids m_i et j de poids m_j à joindre pour former la paire P. Une fois ces éléments choisis, il faut alors calculer la distance entre cette paire et les éléments k présents dans la matrice.

Comme nous pouvons le constater dans l'énoncé des algorithmes de la cah et de NJ, ceux-ci reposent sur le même principe agglomératif et diffèrent uniquement par les étapes présentées dans le tableau 1.1 page ci-contre. Les méthodes présentées dans la section suivante reposent comme NJ sur le principe d'évolution minimum et utilisent les moindres carrés ordinaires pour estimer la longueur des branches de l'arbre. C'est au niveau de l'algorithmique qu'elles diffèrent.

FastMe [Desper and Gascuel, 2002]

[Desper and Gascuel, 2002] ont montré que le principe d'évolution minimum était statistiquement cohérent avec l'utilisation des moindres carrés ordinaires. Et ceci malgré le fait que les estimations des distances, obtenues par exemple à partir de données de séquences, n'aient pas la même variance, les grandes distances étant plus variables que les courtes. [Desper and Gascuel, 2002] ont donc choisi d'optimiser l'utilisation des moindres carrés ordinaires avec le principe d'évolution minimume. Ils proposent donc deux algorithmes de construction d'arbre :

– un algorithme combinant l'utilisation des moindres carrés ordinaires avec le principe d'évolution minimum et ayant l'avantage d'être plus rapide que NJ et ses variantes.

	Choix de la paire	Réduction de la matrice	
CAH	Valeur minimale de la matrice	Simple	$Min(d_{ik}, d_{jk})$
		Complet	$Max(d_{ik}, d_{jk})$
		Moyenne	$\frac{d_{ik}+d_{jk}}{2}$
		Moy. pondérée	$\frac{m_i*d_{ik}+m_j*d_{jk}}{m_i+m_j}$
		Ward	$\frac{(m_i+m_k)*d_{ik}+(m_j+m_k)*d_{jk}-m_k*d_{ij}}{m_i+m_j+m_k}$
Neighbor-Joining	Paire qui maximise : $S_i + S_j - (r-2) * d_{ij}$ où $S_x = \Sigma_{i=1}^{r} d_{xk}$	$\frac{1}{2}(d_{ik} - d_{iP}) + \frac{1}{2}(d_{jk} - d_{jP})$	
BioNJ	Même étape que NJ	$\lambda(d_{ik} - d_{iP}) + (1-\lambda)(d_{jk} - d_{jP})$	
Weighbor	$g * Add(i,j) + Pos(i,j)$	$\frac{(d_{ik}-d_{iP})/\sigma^2_{moy}(iP)+(d_{jk}-d_{jP}/\sigma^2_{moy}(jP))}{1/\sigma^2_{moy}(iP)+1/\sigma^2_{moy}(jP)}$	

Tab. 1.1: Comparaison de deux étapes des algorithmes des méthodes de construction d'arbre. La première étape consiste à déterminer i de poids m_i et j de poids m_j, les objets de la paire P. La seconde étape consiste à calculer la distance entre P et les autres éléments de la matrice k

– une version "pondérée" de l'algortithme précédent. Dans cette nouvelle version, chacun des deux sous-arbres considérés (lors d'une étape décrite ci-dessous) ont le même poids alors que dans l'algorithme standard chaque sous-arbre a un poids égal au nombre d'objets qu'il contient.

Algorithme basé sur les moindres carrés ordinaires L'algorithme, appelé GME (pour Greedy Mimimum Evolution), est le suivant :

– **Initialisation :** Chaque objet est considéré comme un sous-arbre et les distances entre ces sous-arbres sont stockées dans une matrice de distance. Un arbre tertiaire T_3 est construit à partir de 3 objets.

– **Insertion :** Un objet k est sélectionné. Chaque point d'insertion possible va être testé et l'objet k sera inséré lorsque la longueur de l'arbre T sera minimale, c'est à dire en minimisant :

$$L + \frac{1}{2}((\lambda - \lambda') * (d_{kA} + d_{BC}) + (\lambda' - 1) * (d_{AB} + d_{kC}) + (1-\lambda) * (d_{AC} + d_{kB}))$$

avec L la longueur de l'arbre composé de A, B et C, $d_{A,B}$ la distance entre A et B et où

$$\lambda = \frac{n_A + n_B * n_C}{(n_A + n_B) * (n_C + 1)}$$

et

$$\lambda' = \frac{n_A + n_B * n_C}{(n_A + n_C) * (n_B + 1)}$$

. La matrice de distance ainsi que le nombre d'objets de chaque sous-arbre sont mis à jour. Cette étape est illustrée par la figure 1.13

– **Test d'arrêt :** L'algorithme prend fin lorsque le dernier objet formant un sous-arbre a été inséré dans l'arbre. Dans le cas inverse on reprend l'exécution de l'algorithme à partir de l'étape d'insertion.

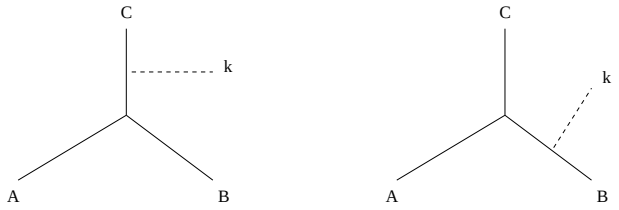


Fig. 1.13: FastMe – L'arbre est composé de trois objets. L'objet k sera inséré au point permettant de minimiser la longueur de l'arbre. Ce processus est répété jusqu'à ce que tous les points d'insertion aient été testés

Cet algorithme a une complexité en temps de calcul en $\mathcal{O}(n^2)$, ce qui est inférieur à NJ et ses variantes qui est en $\mathcal{O}(n^3)$.

Version pondérée de l'algorithme précédent Cet algorithme, appelé BME (pour Balanced Mimimum Evolution) est similaire au précédent. La différence principale a lieu dans l'étape d'insertion lors du la mise à jour de la matrice de distance. La distance entre le sous-arbre formé par A et k et le sous-arbre B va dépendre de la position de k dans A.

De plus, FastMe propose d'améliorer les arbres obtenus à l'aide de deux méthodes de type NNi, les méthodes BNNi et FastNNi. Ces méthodes sont basées sur des réarrangements locaux de la topographie de l'arbre, consistant à échanger deux branches de l'arbre et a vérifier si l'arbre est meilleur, en terme de représentation des données, que le précédent. FastNNi (*resp.* BNNi) est basé sur l'algorithme GME (*resp.* BME).

Dans la suite de ce manuscrit, nous noterons ainsi les différents algorithmes lors de leur utilisation :

– GME_o (*resp.* BME_o) l'algorithme seul
– GME_b (*resp.* BME_b) l'algorithme utilisé avec l'amélioration de l'arbre obtenu avec la méthode BNNi
– GME_n (*resp.* BME_n) l'algorithme utilisé avec l'amélioration de l'arbre obtenu avec la méthode FastNNi

Toutes les méthodes que nous avons présentées ci-dessus permettent de construire des arbres à partir d'une matrice de distance. Dans la prochaine section, nous présenterons un moyen de

mesurer l'adéquation entre la représentation arborée et les données initiales.

1.3.3 Validation

Il est important de pouvoir mesurer la qualité d'une représentation obtenue à partir d'un jeu de données. Des méthodes ont donc été développées afin de mesurer la distorsion entre la matrice de dissimilarité initiale et la matrice de dissimilarité induite associée à la structure arborée. Comme nous l'avons vu pour la classification hiérarchique, une fois l'arbre (ou la hiérarchie) obtenu on peut obtenir une matrice de dissimilarité induite liée à la topologie et à l'indiçage de l'arbre. Il est également possible de calculer la dissimilarité induite à partir d'un arbre obtenu par la méthode du Neighbor-Joining lorsque celui-ci est enraciné. Ainsi, on peut mesurer quelle méthode représente au mieux les données initiales.

Plusieurs auteurs se sont intéressés à mesurer la distorsion de la dissimilarité initiale d_{ij} par rapport à la dissimilarité induite h_{ij}. Les principaux coefficients mesurant cette distorsion sont les suivants :

- *le coefficient de corrélation "cophenetic" [Sokal and Rohlf, 1962] :*

$$D_1 = \frac{\sum(d_ij - \bar{d})(h_ij - \bar{h})}{\sqrt{\sum(d_ij - \bar{d})^2 \sum(h_ij - \bar{h})^2}}$$

  Le coefficient de corrélation "cophenetic" mesure le degré de linéarité entre d_{ij} et h_{ij}. Cet indicateur n'est pas très informatif en raison du nombre important de valeurs identiques dues à la condition d'ultramétrique.
- *la distance euclidienne [Hartigan, 1967] :*

$$D_2 = \sum \omega_{ij}(d_{ij} - h_{ij})^2 \tag{1.8}$$

  où w_{ij} est un facteur de pondération associé à chaque couple (i, j). C'est souvent ce coefficient qui est minimisé dans le cas des algorithmes *directement optimaux*.

1.4 Conclusion

Dans ce chapitre, nous avons présenté la biologie moléculaire et son dogme principal, introduit par Francis Crick : chez tous les êtres vivants, l'information n'est transmise que dans un sens : de l'ADN à l'ARN aux protéines, les constituants de base qui font fonctionner la cellule et l'organisme entier. Depuis une vingtaine d'année, le volume des données biologiques a cru de façon importante, entrainant ainsi une hausse des besoins en stockage, gestion et analyses de ces données. Parmi ces analyses, nous avons choisi de mettre en avant l'approche comparative qui est appliquée au transcriptome, au protéome ou encore à la reconstruction phylogénétique.

Ces méthodologies requièrent des représentations des données sous forme d'arbre. En effet, celles-ci permettent une interprétation et une analyse facilitée d'importants volumes de données. Nous avons sélectionné un ensemble de ces méthodes permettant de construire un arbre à partir d'une matrice de distance. Un des points importants pour ces techniques est la capacité à gérer un grand volume de données et donc une algorithmique simple. Les algorithmes agglomératifs comme la classification ascendante hiérarchique et le Neighbor-Joining (et ses variantes) répondent en effet à ce critère. Il en est de même pour FastMe. Nous utiliserons ces différentes approches dans le chapitre "Influence de la structure de guidage".

Dans le chapitre suivant, nous présentons une méthode de classification généralisant les hiérarchies : la classification pyramidale qui est le sujet principal de cette thèse. De part ses propriétés que nous détaillerons, cette approche permet une représentation simple des données tout en apportant des informations supplémentaires aux méthodes présentées précédemment.

Chapitre 2

Consolidation de la classification pyramidale

Les méthodes de classification ont pour but de mettre en évidence une structuration en groupes ou en classes homogènes d'individus au sein d'une population. Le résultat obtenu par une méthode de classification est une partition ou une suite de partitions emboîtées. Généralement, les méthodes de classification permettent aussi d'obtenir une représentation graphique simple des objets étudiés. La classification hiérarchique, décrite dans le chapitre précédent, est une des méthodes de classification les plus fréquemment utilisées. Elle permet d'obtenir une séquence de partitions emboitées de la plus fine à la plus grossière représentée par un dendogramme ou arbre hiérarchique. Les classes sont organisées par inclusion et sont disjointes. Cependant, dans de nombreuses situations réelles, les données appartiennent rarement à une classe unique. Par exemple, les animaux peuvent appartenir à plusieurs catégories, les livres à plusieurs thèmes, les films à plusieurs genres... Le modèle hiérarchique ne permet pas de représenter le recouvrement entre classes, ce qui peut engendrer une perte d'information notamment sur le lien entre les classes. Ce lien peut être visualisé par des recouvrements de classe et/ou par l'ordonnancement des classes.

L'idée des méthodes de recouvrement ou de la généralisation du modèle hiérarchique avec des classes recouvrantes est déjà présente dans les travaux de [Jardine and Sibson, 1968], puis dans ceux de [Sneath and Sokal, 1973]. L'idée est de généraliser le modèle hiérarchique en conservant plus d'information sur les relations entre les objets. Dans le cadre de la théorie des graphes, [Hubert, 1974] a aussi considéré des classifications conduisant à des classes recouvrantes.

L'introduction de la notion d'ordre en classification est issue d'un problème d'archéologie, celui de la tendance évolutive sur les données. [Robinson, 1951] modélise le problème par la recherche d'un ordre sur l'ensemble des données, traduisant "au mieux" une évolution (ici

chronologique), en se basant sur le fait que des dépôts proches dans le temps auraient des distributions similaires des différents composants. L'ordre est ainsi à la base d'une dissimilarité de Robinson. Le problème de sériation a été étudié par d'autres auteurs, notamment par [Guénoche, 1974]. Par ailleurs, [Brossier, 1980] remarque le fait qu'un arbre de classification hiérarchique admet plusieurs représentations possibles liées à l'ordre des objets sur l'axe horizontal. Toutes ces représentations étant équivalentes, le problème qui se pose est de choisir parmi toutes les représentations possibles celle qui, au sens de la relation d'ordre qu'elle induit sur l'axe horizontal, est la meilleure par rapport aux données initiales.

En 1984, [Diday, 1984a] introduit une nouvelle structure : la classification pyramidale. Les pyramides constituent une généralisation des hiérarchies en permettant la représentation des recouvrements emboités au lieu de partitions. Elles ont été ensuite étudiées par [Bertrand, 1986, Bertrand and Diday, 1990a, Brito, 1991, Mfoumoune, 1998, Aude, 1999]... Les pseudo-hiérarchies, introduites par [Fichet, 1984, Durand and Fichet, 1988] et généralisées par [Brucker, 2001], ont aussi pour principe de généraliser les hiérarchies. Les principes de ces méthodes étant très similaires, et notre objectif étant d'utiliser les propriétés de ces méthodes pour l'étude de données biologiques, nous utiliserons le terme pyramide pour caractériser indifféremment ces méthodes. Les pyramides font appaître des recouvrements emboîtés (*i.e.* deux classes ne sont pas nécessairement disjointes) au lieu de partitions emboîtées comme c'est le cas pour les hiérarchies. Cette propriété réduit le nombre d'ordres compatibles sur les objets de la classification et permet de représenter plus fidèlement les liens entre les objets classés. Une des propriétés importante des pyramides pour l'analyse de données est de pouvoir associer un ordre partiel à chaque pyramide. Sur la figure 2.1, on montre un exemple de hiérarchie où les objets peuvent être permutés. Les deux représentations sont équivalentes et on ne peut pas déterminer d'ordre sur les objets. Tandis que sur la figure du milieu qui représente les mêmes données par une pyramide, l'ordre des objets est fixé et indique que l'objet B fait le lien entre les classses (A,B) et (B,C). Cette information peut avoir une importance pour l'interprétation des données, comme nous le verrons par exemple sur des données biologiques

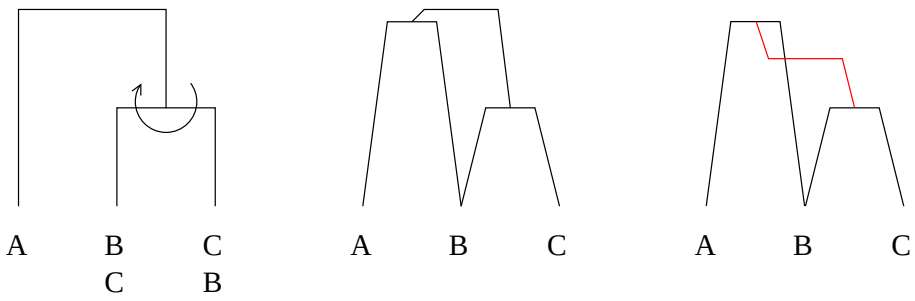


Fig. 2.1: Hiérarchie et pyramide – A gauche : hiérarchie (classes disjointes, B et C peuvent être permutés). Au centre : pyramide (classes recouvrantes, objets partiellement ordonnés). A droite : pyramide avec inversion (inversion entre les classes (AB) et (BC))

L'algorithme le plus utilisé pour calculer une pyramide est la Classification Ascendante Pyramidale [Bertrand and Diday, 1990a]. C'est une adaptation de son homologue hiérarchique au modèle pyramidal. Son principe est celui d'un algorithme agglomératif qui procède à la construction des classes par fusion itérative des objets ou des classes jusqu'à l'obtention d'une classe unique. A chaque étape de l'algorithme, les nouvelles classes créées sont de moins en moins homogènes. L'algorithme proposé par [Bertrand and Diday, 1990a] présente malheureusement dans certains cas un biais qui perturbe la lisibilité des résultats et peut nuire à l'interprétation (figure 2.1 page précédente, droite). En effet, l'algorithme peut produire des *inversions* dans la représentation pyramidale. Une inversion est une classe, agrégeant deux autres classes, indiquée comme plus homogène que l'une de ses deux sous-classes. De part ses propriétés, une inversion ne respecte pas la définition des pyramides.

L'objectif de ce chapitre est de consolider la construction des pyramides pour supprimer les inversions et rendre leur représentation plus robuste et plus proche des données initiales. Nous proposons de corriger ce biais a posteriori. Dans ce but, nous avons défini plusieurs méthodes de *filtrage* que nous allons présenter et comparer. Nous allons aussi montrer que la pyramide corrigée est plus proche des données initiales au sens du critère des moindres carrés.

Nous commencons d'abord par définir de façon formelle les pyramides et nous illustrons leur apport sur une famille de séquences protéiques. Ensuite, nous présentons l'algorithme de classification ascendant pyramidal et le biais introduisant des inversions dans la représentation. Puis, nous définissons plusieurs méthodes de filtrage a posteriori permettant de supprimer les inversions. Pour finir, nous comparons ces solutions de façon théorique et expérimentale et nous les appliquons à l'étude d'une famille de protéines.

2.1 Présentation de la classification pyramidale

Dans cette partie, nous donnons les définitions et propriétés des pyramides, à partir de celles des hiérarchies, afin de mettre en avant les points communs et les différences entre ces méthodes. Nous présenterons également l'algorithme de la Classification Ascendante Pyramidale ainsi que son biais inhérent.

Une pyramide est représentée sous la forme d'un graphe planaire (*cf.* figure 2.2 page suivante). A l'image des hiérarchies, les noeuds terminaux de ce graphe représentent les objets. Chaque noeud interne, que l'on appelle *classe* ou *palier* et que nous noterons P, représente un sous-ensemble d'objets de Ω. Sur cette figure nous pouvons observer que l'objet `D` est inclus dans deux paliers distincts que l'on a noté $P1$ et $P2$. On a donc un recouvrement des sous-ensembles engendrés par les deux paliers $P1$ et $P2$. L'objet `D` est à l'intersection des deux sous-ensembles. Cette caractéristique est une des conséquences de la généralisation du modèle hiérarchique en

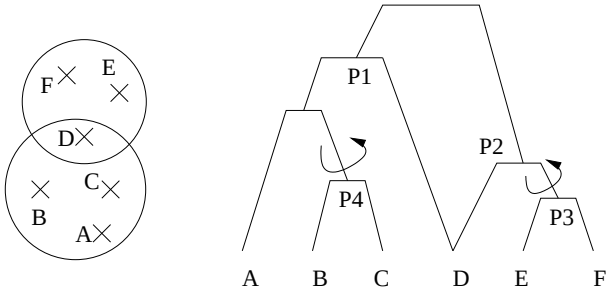


Fig. 2.2: Exemple de pyramide – A gauche : données regroupées dans un plan. A droite : données représentées par une pyramide qui autorise les recouvrements et ordonne partiellement les objets. L'objet D fait le lien entre les deux classes.

modèle pyramidal. On remarque que certains objets peuvent encore être permutés (B et C par exemple).

Nous allons maintenant présenter les définitions formelles des pyramides et leurs propriétés.

2.1.1 Définition d'une structure pyramidale

Pour définir les pyramides, nous nous basons sur les définitions données dans la partie sur la classification hiérarchique.

Soient Ω un ensemble fini d'objets et S un sous-ensemble de $P(\Omega)$.

Une pyramide est caractérisée par la définition suivante :

Définition 2.1 (Pyramide) *Un sous-ensemble P de $P(\Omega)\backslash\{\emptyset\}$ est une pyramide sur Ω s'il vérifie les conditions* (1.1), (1.2), (1.4) *et* (1.5).

Les pyramides et les hiérarchies sont liées les unes aux autres par la proposition suivante établie par [Diday, 1984a] :

Proposition 2.1 *L'ensemble des hiérarchies sur Ω est inclus dans l'ensemble des pyramides sur Ω.*

Nous définissons une fonction indicée pour représenter l'homogénéité des classes. Celle-ci représente la proximité entre les classes agrégées et permet ainsi de valuer une pyramide. La caractérisation de cette fonction, notée f, nécessite les conditions suivantes :

$$\forall K \in S : f(K) = 0 \Leftrightarrow |K| = 1 \tag{2.1}$$

$$\forall (K, L) \in S^2 : K \subseteq L \Rightarrow f(K) \leq f(L) \tag{2.2}$$

Nous pouvons maintenant associer une fonction indicée à une pyramide comme défini ci-dessous.

Définition 2.2 (Pyramide indicée) *(S, f) est une pyramide indicée sur Ω si S est une pyramide et f satisfait les conditions* (1.6) *et* (1.7).

La dissimilarité induite par une hiérarchie indicée est une ultramétrique (définition 1.3.2 page 36). A partir d'une pyramide indicée, nous pouvons aussi inférer une dissimilarité entre les éléments de Ω.

Définition 2.3 (Indice pyramidal) *La dissimilarité induite d par une pyramide indicée (P, f), appelée indice pyramidal, est une dissimilarité satisfaisant les conditions suivantes :*

1. $d(i, j) = 0 \Rightarrow i = j$
2. *Il existe un ordre Θ défini sur Ω tel que tout triplet i, j, k éléments de Ω vérifiant $i\Theta j\Theta k$ satisfait l'inégalité suivante :*

$$d(i, k) \geq \max(d(i, j), d(j, k))$$

De même, la proposition suivante caractérise la dissimilarité induite par une pyramide indicée.

Proposition 2.2 *La dissimilarité (d) induite par une pyramide est robinsonienne :*
$\forall i, j, k \in \Omega, \exists \Theta$ *un ordre total sur* $\Omega : i\Theta j\Theta k \Rightarrow d(i, k) \geq max(d(i, j), d(j, k))$.

L'auteur [Diday, 1984a] a également étendu le théorème classique de Johnson-Benzecri au modèle pyramidal. Ce théorème établit l'existence d'une bijection entre une pyramide indicée et une dissimilarité robinsonienne.

Nous venons de définir de façon formelle les pyramides. Nous allons ensuite présenter les propriétés qui découlent de ces définitions.

2.1.2 Propriétés et description des pyramides

L'apport des pyramides par rapport aux hiérarchies pour analyser des données se fait grâce à ses deux propriétés caractéristiques :

- le recouvrement possible des classes
- l'obtention d'un ordre partiel sur les données

Nous donnons ensuite des définitions relatives à la description des pyramides.

Recouvrement

Deux paliers d'une pyramide peuvent avoir une intersection non-vide, sans être inclus l'un dans l'autre. Les classes ne sont pas nécessairement disjointes. Les pyramides permettent donc de visualiser les *recouvrements*, contrairement aux hiérarchies.

Sur la figure 2.2 page 46, nous pouvons voir que l'objet D appartient à deux classes qui ne sont pas incluses l'une dans l'autre : ce sont des classes *empiétantes*. Pour éviter des chevauchements qui rendraient la lecture difficile, on représente chaque coté d'un palier de manière oblique, orienté vers l'intérieur du palier.

Ordre partiel

La notion d'ordre dans le modèle pyramidal est fondamentale. La contrainte d'ordre empêche la permutation systématique des éléments d'un palier. Il existe non pas un ordre, mais un ensemble d'ordres compatibles engendrés par une pyramide. Sur la figure 2.2 page 46, on peut dire qu'il y a au moins deux ordres compatibles : {A, B, C, D, E, F} et {A, B, C, D, F, E}.

Un palier est dit *"réordonnable"* si l'on peut permuter les éléments qui le composent. Ainsi, le passage de l'ordre {`A, B, C, D, E, F`} vers {`A, C, B, D, F, E`} est possible par permutations des éléments des paliers `P3` et `P4`, comme l'indiquent les flèches. L'objet `D` empêche aux paliers `P1` et `P2` d'être réordonnables. La rotation de l'un de ces paliers induirait un croisement, nuisant à la lisibilité de la représentation.

[Bertrand, 1986] a donné une condition pour déterminer si un palier est *"réordonnable"* ainsi que le nombre total d'ordres compatibles engendrés par une pyramide.

Description des pyramides

Ci-dessous nous donnons un certain nombre de définitions relatives à la description des pyramides. Ces définitions serviront pour la compréhension de l'algorithme de construction d'une pyramide.

Définition 2.4 (Successeur et Prédécesseur) *Soit P une pyramide, et $(p, q) \in P \times P$, alors p est un* successeur *de q si et seulement si :*

- $p \subset q$
- $\nexists\, h \in P \mid p \subset h \subset q$

Si p est un successeur *de q, alors q est le* prédécesseur *de p.*

Dans une pyramide, tout palier a deux successeurs, et a un ou deux prédécesseurs (à l'exception du palier contenant Ω). Par exemple sur la figure 2.2 page 46, les successeurs du palier $P2$ sont D et $P3$, et le prédécesseur de $P3$ est $P2$.

Soit la pyramide P, et '$<$' l'ordre total induit par une de ces représentations. Pour chaque palier, on peut définir un min et un max et calculer l'union de plusieurs paliers.

Définition 2.5 (Minimum et maximum d'un palier) *Pour chaque palier $p \in P$, le minimum* $\min(p)$ *et le maximum* $\max(p)$ *sont définis par :*

- $\min(p)$ *correspond au singleton le plus à gauche, inclus dans p.*
- $\max(p)$ *correspond au singleton le plus à droite, inclus dans p.*

On appelle bordure de p, son minimum ou son maximum.

On a par exemple, sur la figure de gauche 2.2, $\min(P2) = D$ et $\max(P2) = F$.

Définition 2.6 (Union) *Soit H un ensemble de paliers d'une pyramide P, alors leur union est l'ensemble des singletons inclus dans tous les paliers de H.*

L'union des paliers $P1$ et $P2$ est égale à {`A, B, C, D, E, F`}, sur la figure 2.2.

Nous venons de présenter les définitions et les propriétés des pyramides pour pouvoir les utiliser dans la suite du manuscrit.

2.1.3 Exemple de pyramide de familles de séquences protéiques

Afin de montrer l'intérêt de la classification pyramidale pour l'analyse de données biologiques, nous présentons un exemple d'application sur une famille de 11 séquences protéiques.

Le jeu de données suivant permet de mettre en évidence l'apport des recouvrements pour analyser un ensemble de séquences. Nous allons étudier une famille de 11 séquences protéiques, toutes issues du génome de *S.cerevisiae*. L'alignement de ces séquences multi-domaines est représenté sur la figure 2.3 page suivante. Cette famille est composée de 11 protéines caractérisées par la présence de 3 domaines : un domaine "endonucléase MG^{2+}-dépendante", un domaine "Leucin-rich repeats" (LRR) et un domaine "Protéine phosphatase 2C". Ces données sont regroupées dans le tableau 2.1 page suivante. Les protéines identifiées par leur locus sont associées au nom du gène correspondant et au(x) domaine(s) qu'elles possèdent.

Les trois gènes NGL présents chez *S.cerevisiae* sont trois gènes non-essentiels qui codent pour des protéines contenant un domaine de forte similarité avec le motif endonucléase MG_{2+}-

Locus	Gène	Fonction/Domaine
YML118W	*ngl3*	endonucléase Mg^{2+}-dépendante
YMR285C	*ngl2*	endonucléase Mg^{2+}-dépendante
YOL042W	*ngl1*	endonucléase Mg^{2+}-dépendante
YAL021C	*ccr4*	endonucléase Mg^{2+}-dépendante et LRR
YOR353C	*sog2*	LRR
YJL005W	*cyaa*	LRR et Protéine phosphatase 2C
YDL006W	*ptc1*	Protéine phosphatase 2C
YER089C	*ptc2*	Protéine phosphatase 2C
YBL056W	*ptc3*	Protéine phosphatase 2C
YBR125C	*ptc4*	Protéine phosphatase 2C
YOR090C	*ptc5*	Protéine phosphatase 2C

Tab. 2.1: Tableau récapitulant le locus des protéines, le gène correspondant et les domaines caractéristiques

dépendant de la mRNA-desadenylase Ccr4P. Leur rôle physiologique est pour le moment mal connu.

Les domaines LRR (Leucin-rich repeats) sont formés de 2 à 45 motifs d'une longueur de 20 à 30 acides aminés qui se replient généralement en forme d'arc ou de fer à cheval. Les LRR sont présents dans les protéines, des virus aux eucaryotes, et semblent fournir un cadre structural pour la formation des interactions protéines-protéines. Parmi les protéines contenant ces motifs, on trouve des récepteurs tyrosine kinase, des molécules d'adhésion cellulaire, des facteurs de virulence, ... et sont impliquées dans une grande variété de processus biologiques :

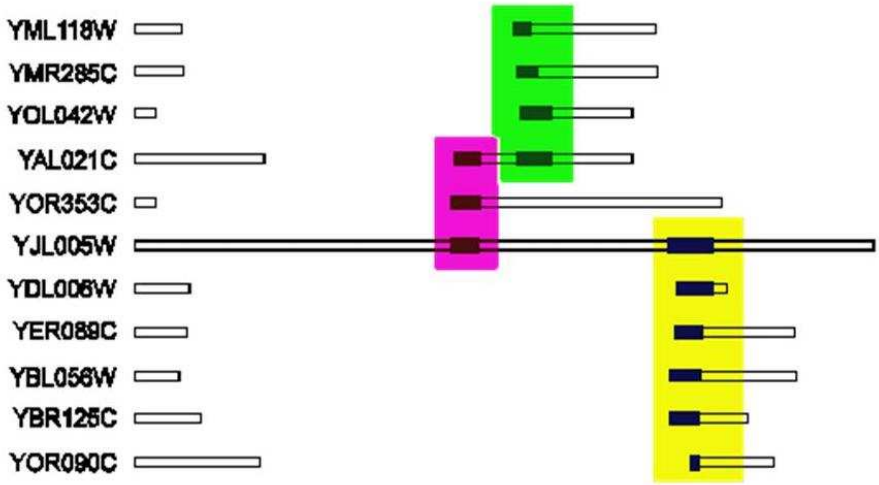


Fig. 2.3: Famille de protéines homologues multi-domaines. – Alignement

transduction du signal, adhésion cellulaire, réparation de l'ADN, recombinaison, transcription, apoptose, réponse immunitaire...

Les protéines phosphatases 2C (PP2C) appartiennent à une des quatre classes majeures de phosphatases mammifères spécifiques des sérines/thréonines. La PP2C est une enzyme monomérique d'environ 42 kDa montrant une grande variété de substrats et dépendant des cations bivalents (principalement manganèse et magnésium) pour son activité. Son rôle physiologique exact n'est pas clair.

La figure 2.4, montre l'arbre obtenu en appliquant l'algorithme UPGMA. Nous pouvons observer deux classes respectivement composées des séquences YAL021C, YOL042W, YML118W, YMR256C et la seconde du reste des séquences.

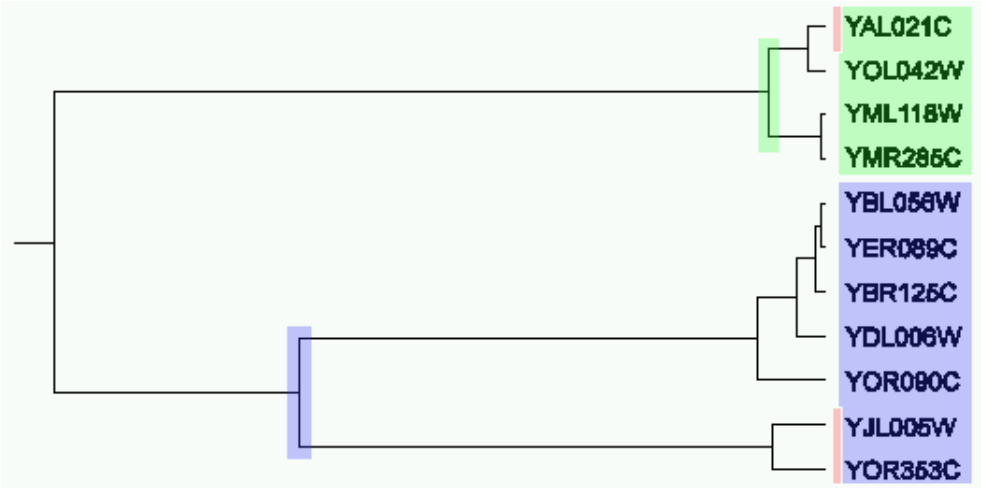


Fig. 2.4: Arbre UPGMA des séquences de levure

L'algorithme de la Classification Ascendante Pyramidale a ensuite été appliqué pour obtenir la représentation 1.7 page 26. Nous pouvons observer trois classes recouvrantes :

1. YML118W, YMR285C, *YAL021C*, *YOL042W*
2. *YAL021C*, *YOL042W*, YOR353C, *YJL005W*
3. *YJL005W*, YDL006W, YER089C, YBL056W, YBR125C, YOR090C

Nous avons *mis en avant* les séquences appartenant à deux classes.

Sur la pyramide, on observe trois classes principales (en vert, rose et jaune sur la figure 1.7 page 26), alors que sur l'arbre, on en observe deux principales (en vert et bleu sur la figure 2.4). En comparant les classes, on observe que :

- La classe verte de la pyramide correspond à la première classe verte de l'arbre.
- La classe jaune de la pyramide correspond, à la séquence `YOR353C` près, à la classe bleu de l'arbre.

– Dans la pyramide, on distingue une troisième classe (en rose) qui permet de faire le *lien* entre les deux autres classes. Sur l'arbre, cette classe n'apparait pas et les séquences concernées sont même positionnées aux extrémités de l'arbre.

Nous connaissons la composition en domaine des séquences (représentée sur la figure 1.7 page 26), et donc nous pouvons en déduire les relations entre les séquences a priori. Il y a 3 domaines présents et donc trois classes de séquences à détecter. De plus, les séquences multi-domaines `YJL005W` et l'une des séquences `YAL021C` et `YOL042W` apparaissent clairement comme faisant le lien entre les différentes classes. La classification pyramidale représente correctement les trois classes et indique clairement la classe faisant le lien avec les deux autres grâce aux séquences multi-domaines. Par contre, l'arbre hiérarchique n'a pas réussi à trouver la troisième classe de séquence.

Dans cet exemple, la classification pyramidale nous a guidé dans l'analyse des relations entre les séquences de cette famille. Sans les propriétes de recouvrement et d'ordre propres aux pyramides, l'interprétation auraient été difficile et incomplète.

Maintenant, nous allons présenter l'algorithme de Classification Ascendante Pyramidale.

2.2 La Classification Ascendante Pyramidale et inversions

L'objectif de ce chapitre est la correction du biais des inversions engendré par l'algorithme de Classification Ascendante Pyramidale. Nous allons donc présenter cet algorithme et ses spécificités, pour ensuite formaliser le biais.

Les algorithmes de classification pyramidale, sont dans la majorité des cas, des adaptations d'algorithmes utilisés pour le modèle hiérarchique. De la même manière, il existe différents principes de construction de pyramides : *(i)* les algorithmes de type agglomératif ; *(ii)* les algorithmes de type divisif ; *(iii)* les algorithmes directement optimaux. D'autres algorithmes ont été proposés par [Fichet, 1984, Durand and Fichet, 1988]. Nous avons choisi d'utiliser l'algorithme de la Classification Ascendante Pyramidale (CAP) pour son bon compromis entre efficacité et simplicité.

La CAP est un algorithme de type agglomératif, qui reprend dans son ensemble les principes de la CAH (voir le rappel de cet algorithme 1.3.1 page 34). Les algorithmes agglomératifs ont pour principe de fusionner itérativement des objets ou des classes jusqu'à obtenir une classe unique. A chaque étape de l'algorithme, les nouvelles classes créées sont de moins en moins homogènes. [Diday, 1984a, Bertrand, 1986] décrivent l'algorithme de la CAP. De nombreuses variations de cet algorithme ont depuis été développées, comme par exemple : [Brito, 1991] a étendu la CAP au traitement de tableaux individus × variables contenant des données complexes ; *CAPII* : [Gaul and Schader, 1994] propose une extension de la CAP qui permet de

traiter le cas de données incomplètes.

2.2.1 Algorithme de la Classification Ascendante Pyramidale

Nous donnons ici la description de l'algorithme CAP tel qu'il a été décrit par [Bertrand, 1986, Bertrand and Diday, 1990a]. Cet algorithme présente un biais dont la correction est l'objectif de ce chapitre.

Le principe de l'algorithme est qu'à chaque itération, on construit un graphe G dont les noeuds sont les paliers que l'on vient de créer, et les arcs les relations de successeurs et prédécesseurs entre les paliers.

Soit Ω l'ensemble des individus, et μ un indice d'agrégation.

L'algorithme est le suivant :

1. **Initialisation :** Les seuls paliers existants sont les singletons de Ω. Un ordre arbitraire sur Ω est choisi.
2. **Agrégation :** Deux paliers p^* et q^* sont agrégés si l'indice d'agrégation $\mu(p^*, q^*)$ est minimum parmi l'ensemble des indices d'agrégation $\mu(p, q)$ où $(p, q) \in P \times P$. De plus p^* et q^* doivent satisfaire les conditions suivantes (illustrées ci-dessous) :
   - *Condition 1.* p^* est avant q^*, c'est à dire $min(p^*) < min(q^*)$
   - *Condition 2.* il n'existe pas de palier x tel que p^* ou q^* soient à l'intérieur de ce palier, c'est à dire qu'il n'existe pas de x tel que $succ(x) = p^*$ et $succ(x) = q^*$
   - *Condition 3.* si p^* et q^* appartiennent à la même composante connexe, alors tout palier x de cette composante vérifie :

     $$\max(p^*) < \max(x) \Rightarrow \min(q^*) \leq \min(x)$$

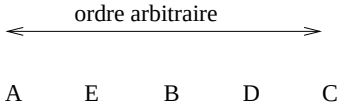
   - *Condition 4.* si p^* et q^* n'appartiennent pas à la même composante, alors le palier p^* contient une borne de $C(p^*)$ et le palier q^* contient une borne de $C(q^*)$.
3. **Mise à jour de l'ordre compatible :** Si au cours de l'étape d'agrégation, p^* et q^* n'appartiennent pas à la même composante connexe, alors les éléments de $C(q^*)$ sont positionnés après ceux de $C(p^*)$. La composante $C(q^*)$ peut être inversée de manière à ce que $\min(q^*)$ soit le premier élément à droite de $\max(p^*)$.
4. **Test d'arrêt :** L'algorithme prend fin quand le palier que l'on vient de créer, $(p^* \cup q^*)$, est égal à Ω. Dans le cas inverse on reprend l'exécution de l'algorithme à partir de la phase d'agrégation.

Un exemple des différentes étapes de l'algorithme est donné par la figure 2.5 page 55. Lors de l'étape d'initialisation, seuls les singletons A,B,C,D,E existent et sont placés dans un ordre arbitraire. Ensuite, lors de l'étape d'agrégation, les paliers A et B sont identifiés comme agrégables

car $d(A, B)$ est la distance minimale de la matrice et ces deux paliers vérifient les quatre conditions décrites précédemment. L'ordre compatible des paliers est alors mis à jour à l'étape suivante et le palier F est créé dans la matrice et la pyramide. Ces étapes sont répétées jusqu'à la construction du palier J qui contient alors l'ensemble des objets, soit Ω.

Etape 1 : Initialisation

A	0	3	6	10	10
B		0	3	10	10
C			0	10	10
D				0	3
E					0



Etape 2 : Agrégation

A	0	(3)	6	10	10
B		0	3	10	10
C			0	10	10
D				0	3
E					0

d(A,B) = min(matrice)

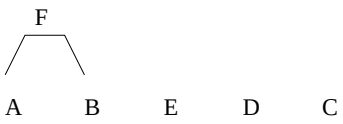
Vérification des 4 conditions :
− A est avant B
− A et B ne sont pas à l'intérieur de classes de P
Cette condition ne s'applique pas à des singletons
− A et B sont sur des bordures

=> A et B peuvent etre agrégés

Etape 3 : Mise a jour de l'ordre compatible

A	0	3	6	10	10	3
B		0	3	10	10	3
C			0	10	10	4,5
D				0	3	10
E					0	10
F						0

d(F,C) = (d(A,C) + d(B,C)) /2
= (6 + 3) / 2 = 4,5
avec le critère de la moyenne



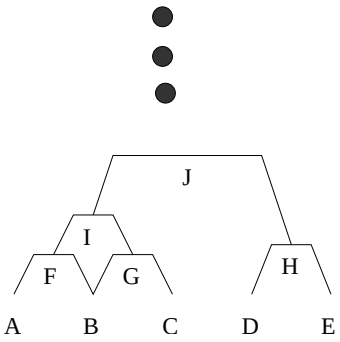


Fig. 2.5: Déroulement des étapes de la classification ascendante pyramidale – (initialisation) seuls les singletons ordonnés arbitrairement existent. (agrégation) $d(A, B)$ est la valeur minimume de la matrice. Les 4 conditions étant vérifiées, A et B sont agrégés. (mise à jour) Les distances des paliers existants et du nouveau palier F sont calculées. (fin) Tous les paliers sont agrégés, nous obtenons une pyramide

Conditions d'agrégation

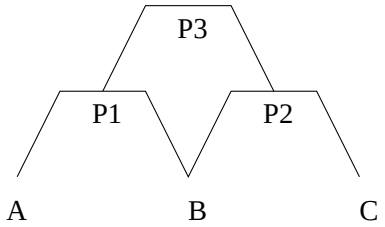
Comme pour la Classification Ascendante Hiérarchique, il y a plusieurs façons de calculer la distance entre un palier nouvellement créé et ceux déja existants. Il existe donc plusieurs critères d'agrégation : lien simple, lien complet, moyenne...

L'algorithme (déroulé sur la figure 2.5 page précédente) utilise le critère d'agrégation de la moyenne.

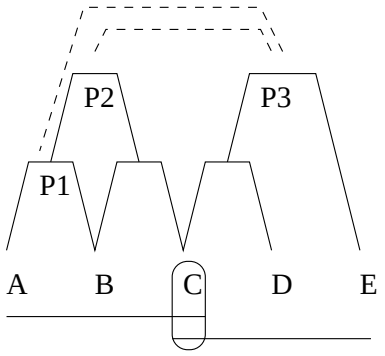
Les conditions que doivent satisfaire les paliers p^* et q^* pour être agrégés décrites dans l'algorithme précédent sont illustrées sur les exemples suivants [Diday, 1984a] :

Condition 1 : Sur la figure ci-contre, on a :
- P1 est avant P2, car :
$\min(P1) = A < \min(P2) = B$
et $\max(P1) = B < \max(P2) = C$
- P1 n'est pas avant P3, car :
$\min(P1) = A = \min(P3) = A$

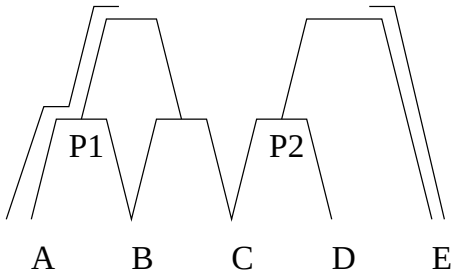
Condition 2 : Sur la figure ci-contre, on a :
A et B sont à l'intérieur de P1, car :
$succ(P1) = A$ et $succ(P1) = B$



Condition 3 : Sur la figure ci-contre, on a :
- P2 peut s'agréger avec P3, car :
$P2 \cap P3 = C$
- P1 ne peut pas s'agréger avec P3, car :
$P1 \cap P3 \neq C$



Condition 4 : Sur la figure ci-contre, on a :
- P1 contient une bordure de C, car :
$\min(P1) = A = \min(C)$
- P2 ne contient pas de bordure de C, car :
$\min(P2) = C \neq \min(C) = A$
et $\max(P2) = D \neq \max(C) = E$

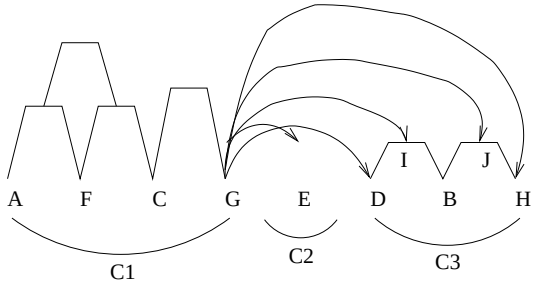


Nous avons utilisé l'algorithme optimisé de la CAP dont nous détaillons ci-après quelques spécificités.

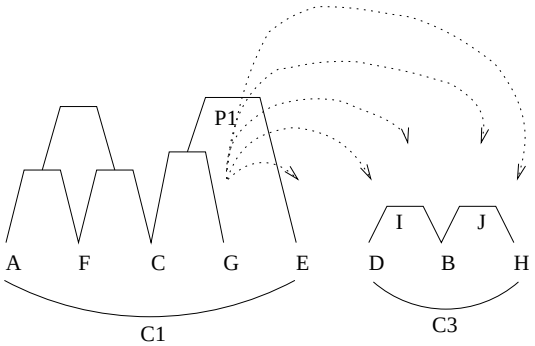
Spécificité de l'algorithme utilisé

La complexité de l'algorithme est au pire en $\mathcal{O}(n^4)$. Aussi, [Mfoumoune, 1998] s'est intéressé à l'optimisation de cet algorithme pour pouvoir traiter de grands volumes de données. Pour cela il part du constat que lors de la phase d'agrégation des paliers, on a un effet de bord qui fait que certains couples de paliers ne peuvent plus être candidats à la phase d'agrégation. L'auteur définit alors la notion d'accessibilité mutuelle entre paliers, qui permet d'identifier les couples de paliers agrégeables, et donc de réduire la complexité de la phase de recherche du couple de paliers satisfaisant les conditions d'agrégation. Sur la figure 2.6 page suivante, nous pouvons voir que les paliers des composantes C2 et C3 sont accessibles depuis G. Lorsque le palier P1 est créé, les paliers candidats rendus inaccesibles sont supprimés : GE, GD, GI, GB, GJ, GH et seuls les nouveaux paliers candidats sont ajoutés (par exemple : {P1K} et {P1L}). La matrice est alors mise à jour. Nous obtenons donc une matrice creuse contrairement à la CAP où les paliers rendus inaccesibles sont conservés et tous les paliers candidats possibles sont créés.

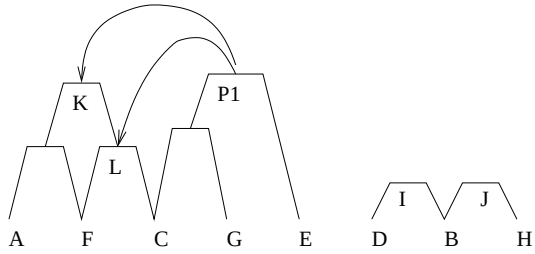
L'implémentation de cet algorithme, *QuickCAP*, a une complexité en $\mathcal{O}(n^2)$ dans le cas numérique. Les autres spécificités de *QuickCAP* sont de permettre la création de pyramides d'objets symboliques [Brito, 1991], et de permettre la construction incrémentale (par ajout progressif d'individus) de pyramides. On notera toutefois que dans ce dernier cas, le résultat peut ne pas être une pyramide au sens de la définition 2.1 page 46.



Les paliers des composantes C2 {E} et C3 {D,B,H,I,J}
sont accessibles depuis G.



Après la création de P1, ils sont supprimés de la matrice d'agrégation



Après la création de P1, on ajoute les paliers candidats à la matrice
d'agrégation

Fig. 2.6: QuickCAP – Notion d'accessibilité entre paliers pour optimisation de l'algorithme

2.2.2 Les inversions

Les algorithmes de classification ascendante pyramidale introduits précédemment ont un biais qui introduit des paliers inversés dans la représentation pyramidale.

Il s'agit d'un biais obtenu lors du processus de construction d'une pyramide et observé dans l'indice pyramidal. Ce biais peut être obtenu avec l'utilisation de tous les critères d'agrégation sauf celui du lien complet.

Une inversion est une classe, agrégeant deux autres classes, indiquée comme plus homogène

que l'une de ses deux sous-classes. Comme le montre l'exemple de pyramide obtenu par l'algoritme de la CAP (figure 2.7), le palier {ABC ∪ D} est inversé. Il est plus homogène que son successeur gauche : son indice est inférieur à celui du palier ABC. Ceci est en contradiction avec la seconde condition de la définition de l'indice pyramidal 2.2 page 47.

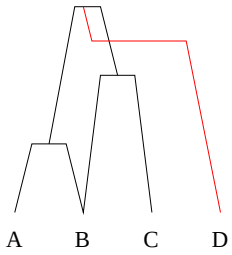


Fig. 2.7: Pyramide – exemple de représentation pyramidale avec palier inversé

Condition d'existence d'une inversion

[Lasch, 1996] a explicité une condition de l'existence de paliers inversés avec le critère du lien simple.

Proposition 2.3 *A l'étape μ de l'algorithme pyramidal avec le lien simple, soient J_μ l'ensemble de classes pas encore agrégées à l'étape μ, $d^*_\mu = \min d(K, L)\ \forall (K, L) \in J_\mu$ la distance minimale entre deux classes et f_μ le niveau de l'indice correspondant. Alors :*

$$\textit{Existence d'une inversion} \Leftrightarrow \exists \mu : d^*_\mu < f_\mu$$

En conséquence, avec l'utilisation de l'algorithme de la CAP, la matrice de dissimilarité induite n'est nécessairement robinsonnienne et la proposition 2.2 page 47 n'est pas vérifiée.

Exemple d'obtention d'une inversion

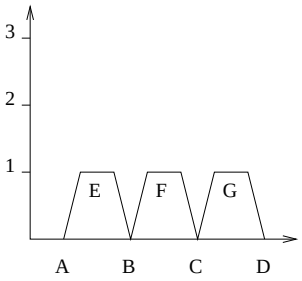
Nous présentons un exemple, sur la figure 2.8 page suivante, où une inversion apparait lors de la construction d'une pyramide.

Matrice de dissimilarité initiale

A	0	1	3	2
B		0	1	5
C			0	1
D				0

Les 3 paliers E = {AB}, F = {BC} et G = {CD} ont été créés et ont permis de déterminer un ordre compatible :

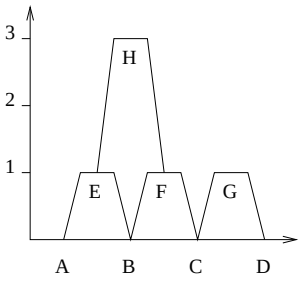
O : A < B < C < D



min(d) = d(A,D) = d(E,G)= 2
Ce palier ne peut etre créé :
la condition 3 n'est pas vérifiée
(incompatible avec O)

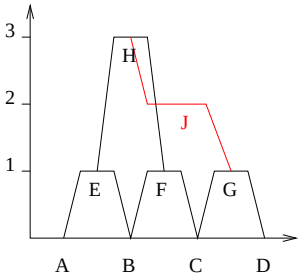
Matrice modifiée : les paliers A, B, C, D ne sont plus agrégeables.

E	**0**	**3**	**2**
F		**0**	**5**
G			**0**



La valeur minimale suivante est :
d(E,F) = d(A,C) = 3
Le palier H est créé :
H = E U F ={A,B,C}

E	0	**3**	2
F		0	5
G			0



min(d) = d(H,G)= d(A,D) = 2
Ce palier vérifie toutes les conditions d'agrégation et est donc créé.
Or, on a :

J< H

Matrice modifiée : les paliers E, F ne sont plus agrégeables.

G	0	**2**
H		0

La matrice de dissimilarité induite n'est pas robinsonienne :

A	0	1	3	2
B		0	1	2
C			0	1
D				0

Fig. 2.8: Construction d'une pyramide avec inversion – Déroulement de l'algorithme

Différentes inversions

Afin de distinguer les différents types d'inversion, nous avons caractérisé trois types de configuration possibles dans la matrice de dissimilarité induite. Elles correspondent à trois cas

où la matrice n'est pas robinsonnienne.

Les trois configurations possibles sont :

- *inversion droite* : la matrice contient deux éléments qui ne sont pas croissants en colonne
- *inversion gauche* : la matrice contient deux éléments qui ne sont pas croissants en ligne qq
- *inversion double* : la matrice contient simultanément deux éléments qui ne sont pas croissants en ligne et deux éléments qui ne sont pas croissants en colonne

Les configurations sont illustrées sur la figure 2.9 avec pour chacune la pyramide et la matrice de dissimilarité induite correspondante.

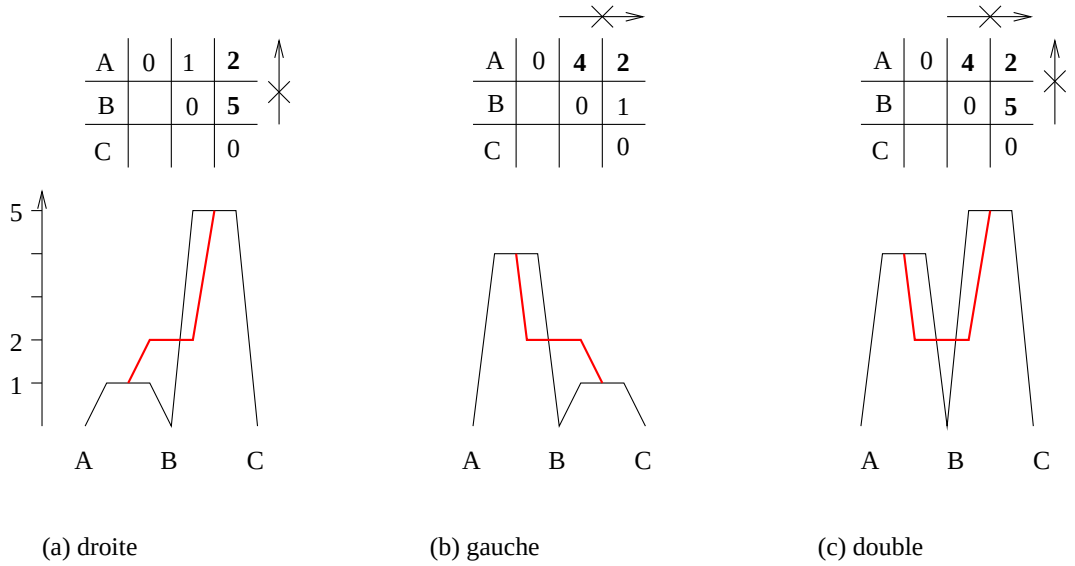


Fig. 2.9: Types d'inversion – (a) inversion droite : la matrice n'est pas croissante en colonne. (b) inversion gauche : la matrice n'est pas croissante en ligne. (c) inversion double : la matrice n'est pas croissante en ligne et colonne.

Dans la prochaine partie, nous introduisons deux approches pour corriger ce biais afin d'obtenir une matrice de dissimilarité induite robinsonnienne.

2.3 Consolidation par suppression des inversions

L'algorithme de la CAP introduit un biais dans l'indice pyramidal : des inversions peuvent apparaître lors de la construction de la pyramide. Cette partie présente deux approches que nous proposons afin de corriger ce biais. Le point de départ de ces méthodes est la matrice de dissimilarité induite calculée à partir de la pyramide (après l'application de l'algorithme de la

CAP). Notre idée est de la transformer en une matrice robinsonienne puis de reconstruire alors la pyramide ; tout en maintenant la contrainte d'ordre sur les données.

A notre connaissance, seuls les auteurs [Gil A.J. and A., 1998] ont proposé une solution pour supprimer les paliers inversés. Ils modifient l'algorithme de construction en ajoutant un critère empêchant la sélection de groupes générant des inversions. En reprenant, l'algorithme de la CAP, nous avons alors :

1. **Initialisation :** Inchangée.
2. **Agrégation :** Deux paliers p^* et q^* sont agrégés si l'indice d'agrégation $\mu(p^*, q^*)$ est minimum parmi l'ensemble des indices d'agrégation $\mu(p, q)$ où $(p, q) \in P \times P$. De plus p^* et q^* doivent satisfaire les conditions suivantes :
   - p^* est avant q^*
   - il n'existe pas de palier tel que p^* ou q^* soient à l'intérieur de ce palier
   - si p^* et q^* appartiennent à la même composante connexe, alors tout palier x de cette composante vérifie :

$$\max(p) < \max(x) \Rightarrow \min(q) \leq \min(x)$$

   - si p^* et q^* n'appartiennent pas à la même composante, alors le palier p^* contient une borne de $C(p^*)$ et le palier q^* contient une borne de $C(q^*)$.
   - si le palier résultat de l'agrégation de p^* et q^* est moins homogène que ces deux paliers
3. **Mise à jour de l'ordre compatible :** Inchangé.
4. **Test d'arrêt :** Inchangé.

Ce nouveau critère est contradictoire avec le principe inhérent de l'algorithme : l'agrégation des groupes les plus proches.

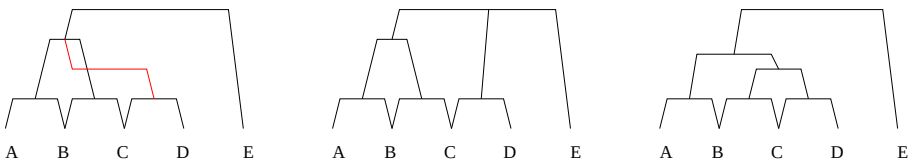


Fig. 2.10: Pyramide initiale et corrections – A gauche, la pyramide initiale avec des inversions. Au centre, la pyramide obtenue avec l'algorithme de [Gil A.J. and A., 1998]. A droite, notre solution.

Notre approche est différente : nous ne modifions pas l'algorithme de calcul de la pyramide. Notre point de départ est la matrice induite, correspondant à la pyramide avec des inversions, afin de conserver la structure originale de la pyramide. Nous supprimons les paliers inversés en appliquant un filtre global sur la matrice pour la rendre robinsonienne. Sur la figure 2.10, nous

pouvons comparer la pyramide initiale avec les inversions (à gauche), la pyramide obtenue avec l'algorithme de [Gil A.J. and A., 1998] (au centre), et la pyramide obtenue avec notre solution (à droite). Notre pyramide est plus en adéquation avec la structure de la pyramide initiale que celle de [Gil A.J. and A., 1998]. La pyramide de [Gil A.J. and A., 1998] a perdu l'information de proximité entre les groupes (C,D) et (A, B, C).

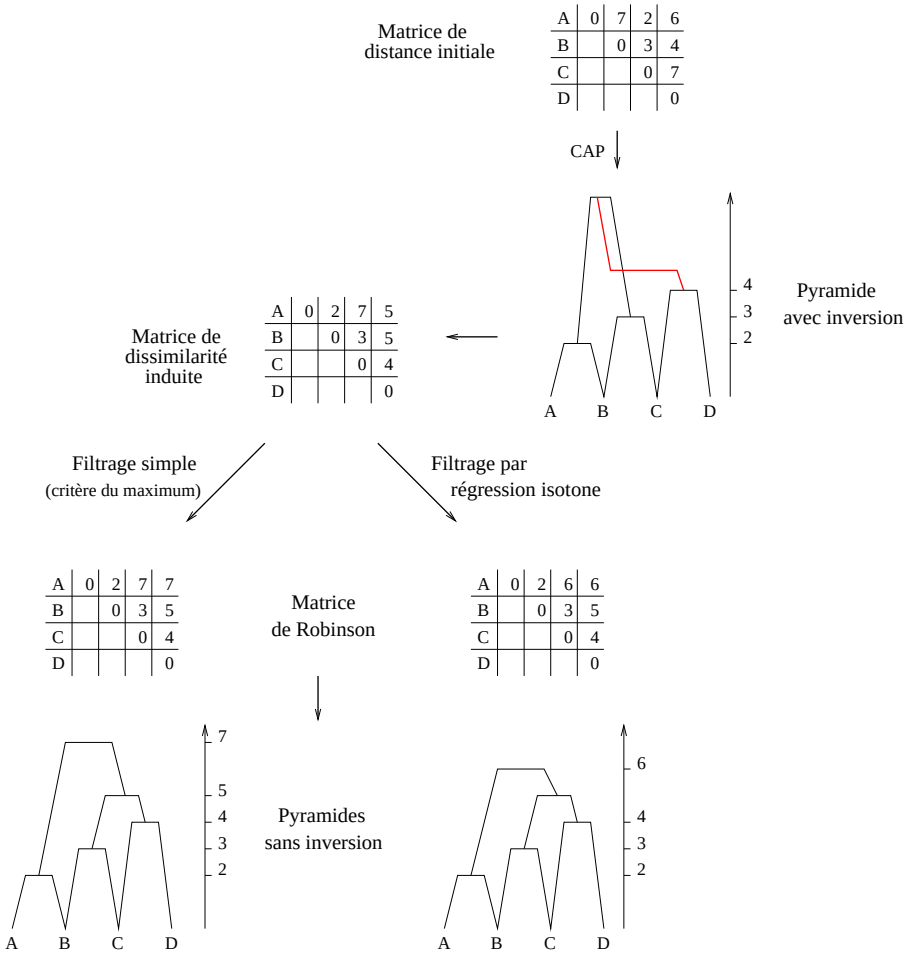


Fig. 2.11: Processus global – Filtrages simple et par régression isotone

Dans cette partie, notre présentons un filtrage simple par seuillage local qui est une approche heuristique pour obtenir une matrice de Robinson et nous proposons un algorithme pour l'implémenter. Puis, nous présentons un filtrage par les régressions isotones comme une solution optimale pour obtenir une matrice robinsonienne et nous proposons l'algorithme associé. Ensuite, nous proposons un algorithme reconstruisant une pyramide à partir d'une matrice de Robinson (*cf* figure 2.11). Enfin, nous comparons théoriquement et expérimentalement les solutions proposées et nous les appliquons à des données biologiques.

2.3.1 Filtrage simple par seuillage local

Suite à la construction d'une pyramide avec des inversions, on observe que la matrice de dissimilarité induite n'est pas robinsonienne. Cela signifie que les valeurs en lignes et colonnes ne sont pas croissantes à partir de la première diagonale. Le principe de cette approche heuristique est le suivant : i) trouver un élément de valeur erronée par rapport à son voisinage ; ii) remplacer celui-ci par une nouvelle valeur. Si nous appliquons cette idée à tous les éléments qui ne respectent pas l'ordre croissant, nous obtenons une matrice dont les lignes et colonnes sont croissantes.

Nous avons déterminé plusieurs critères pour le seuillage local : maximum, minimum, moyenne et moyenne pondérée. Dans le cas du maximum (*resp.* minimum), on seuillera en utilisant le maximum (*resp.* minimum) local du voisinage de l'élément. Dans les deux cas de moyenne, le seuillage est plus complexe et sera détaillé ultérieurement. Sur la figure 2.12, nous présentons une pyramide avec inversion corrigée avec chacun des critères. Nous allons à présent décrire plus précisément le filtrage par seuillage local avec chaque critère.

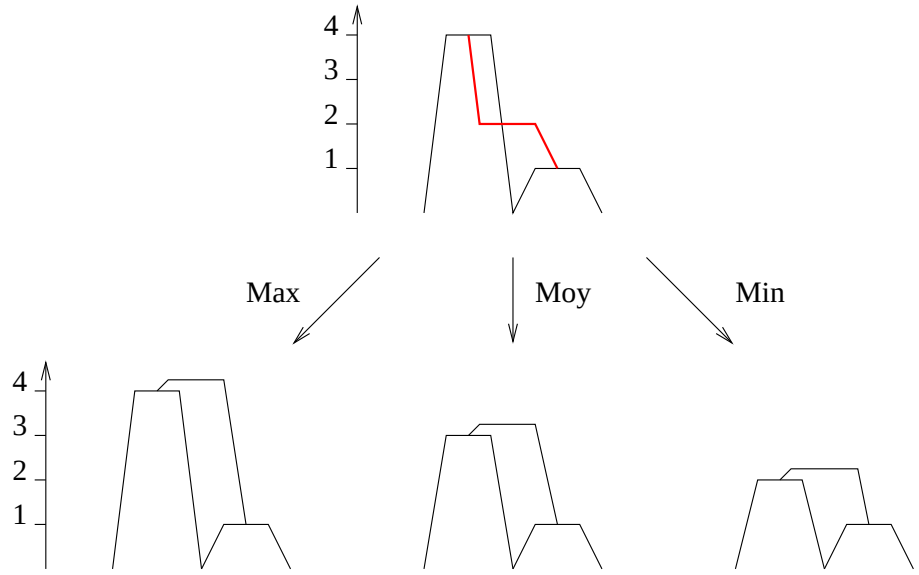


Fig. 2.12: Filtrage par seuillage local – Critères du maximum, du minimum et de la moyenne

Filtrage local avec le critère du maximum

Soient D la matrice triangulaire supérieure de taille n, i et j les indices de ligne et colonne de D. On notera $d_{i,j}$ la valeur de l'élément repéré par i et j. Sur la figure 2.13 page suivante, cet élément est représenté avec les éléments voisins dans la matrice. A droite, sont représentés les éléments pris en compte pour le seuillage dans le cas du critère du maximum. On peut alors

définir :

$$d_{i,j} = max(d_{i,j}, d_{i,j-1}, d_{i+1,j})$$

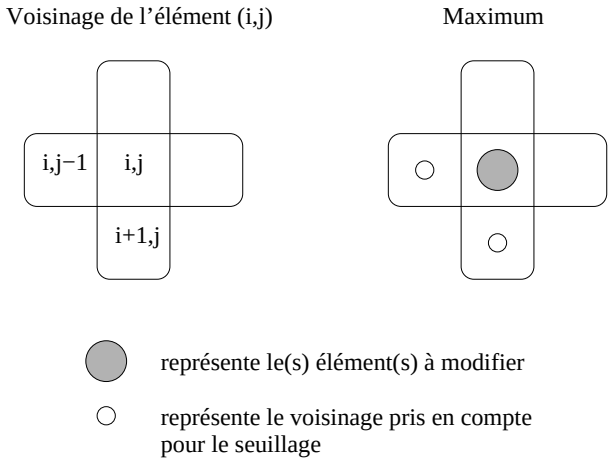


Fig. 2.13: Eléments voisins nécessaires au seuillage dans la matrice – Critère du maximum

Trois types d'inversion (droite, gauche ou double) ont été identifiés. Nous donnons ci-dessous un exemple de filtrage par le critère du maximum pour chaque type d'inversion.

La figure 2.14 présente une inversion droite et son filtrage. La matrice de gauche n'est pas robinsonienne ($2 < 5$). La matrice de droite est robinsonienne après le remplacement de la valeur concernée (2) par la valeur maximale (5).

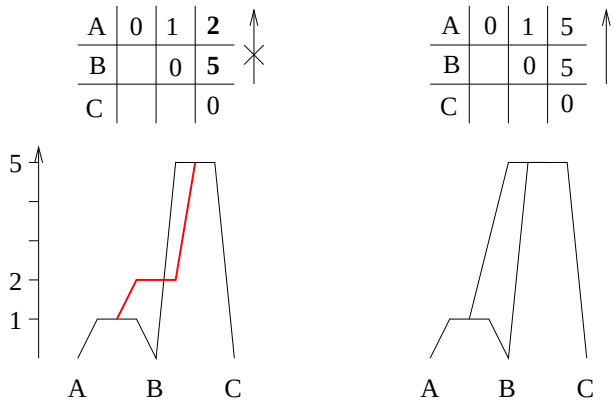


Fig. 2.14: Filtrage par seuillage local avec le critère du maximum – Inversion droite

La figure 2.15 page suivante présente une inversion gauche et son filtrage. La matrice de gauche n'est pas robinsonienne ($2 < 4$). La matrice de droite est robinsonienne après le remplacement de la valeur concernée (2) par la valeur maximale (4).

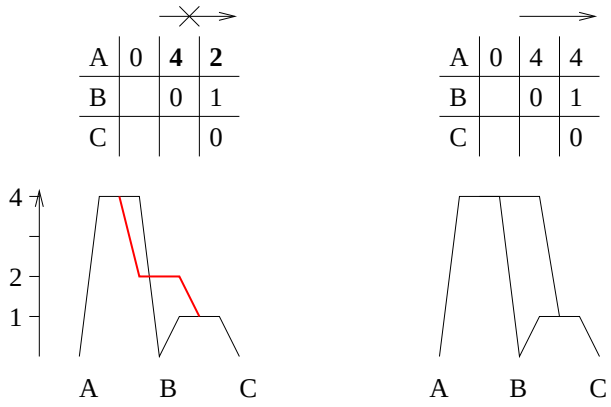


Fig. 2.15: Filtrage par seuillage local avec le critère du maximum – Inversion gauche

La figure 2.16 présente une inversion double et son filtrage. La matrice de gauche n'est pas robinsonienne ($2 < 4$ et $2 < 5$). La matrice de droite est robinsonienne après le remplacement de la valeur concernée (2) par la valeur maximale ($5 = \max(4,5)$).

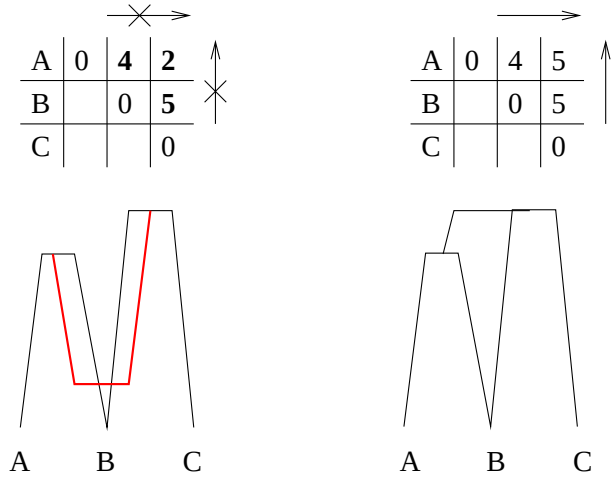


Fig. 2.16: Filtrage par seuillage local avec le critère du maximum – Double inversion

Algorithme de filtrage local avec le critère du maximum L'algorithme implémentant cette idée est le suivant : soit D la matrice triangulaire supérieure de taille n, $d_{i,j}$ un élément duquel i et j sont les indices de ligne et colonne. Les éléments de la matrice sont explorés diagonale par diagonale en commençant par la première diagonale supérieure et pour chaque élément, nous vérifions cette propriété : si $d_{i,j} \leq max(d_{i,j-1}, d_{i+1,j})$ alors $d_{i,j} = max(d_{i,j-1}, d_{i+1,j})$.

Listing 2.1 Seuillage local avec le critère du maximum

```
1: i = 1, size = Card(D) {initialisations}
2: while i < size − 1 do
3:    j = 0, k = i
4:    while i < size − 1 AND j < i do
5:       if i − j = k then {Parcours diagonale par diagonale}
6:          if D(i, j) < D(i, j − 1) OR D(i, j) < D(i + 1, j) then
7:             D(i, j) = max(D(i, j − 1), D(i + 1, j))
8:          end if
9:       end if
10:      j = j + 1, i = i + 1
11:   end while
12:   i = k, i = i + 1
13: end while
```

Preuve Nous allons prouver cet algorithme par récurrence. Soient i et j les indices de ligne et colonne. L'hypothèse de récurrence est la suivante : tous les éléments qui sont sur une diagonale inférieure à la diagonale I vérifient la propriété : $d_{i,j} \geq max(d_{i,j-1}, d_{i+1,j})$.

Premier cas : sur la diagonale I+1, tous les éléments vérifient la propriété. Alors l'hypothèse de récurrence est toujours vérifiée sur la diagonale I+1.

Second cas : sur la diagonale I+1, un élément $d_{i,j}$ change. Alors nous avons $d_{i,j} = max(d_{i,j-1}, d_{i+1,j})$. La nouvelle valeur de l'élément $d_{i,j}$ est obtenue à partir des valeurs des éléments de la diagonale I. Les valeurs des éléments de la diagonale I sont inchangées, donc il n'y a pas d'effet sur les autres éléments de la diagonale I+1. Après remplacement, $d_{i,j}$ vérifie la propriété. Finalement, plusieurs éléments de la diagonale I pourraient être changés dans un autre ordre et après toutes les modifications les éléments de la diagonale I+1 vérifient la propriété.

Pour conclure, nous démontrons que, avec l'exploration diagonale par diagonale de tous les éléments, si nous changeons la valeur d'éléments ne vérifiant pas la propriété, nous convergeons vers une matrice de Robinson.

Filtrage local avec le critère du minimum

Soient D la matrice triangulaire supérieure de taille n, i et j les indices de ligne et colonne de D. On notera $d_{i,j}$ la valeur de l'élément repéré par i et j. Sur la figure 2.17 page suivante, cet élément est représenté avec les éléments voisins dans la matrice. A droite, sont représentés les éléments pris en compte pour le seuillage dans le cas du critère du minimum. On peut alors

définir :

$$d_{i,j} = min(d_{i,j}, d_{i-1,j}, d_{i,j+1})$$

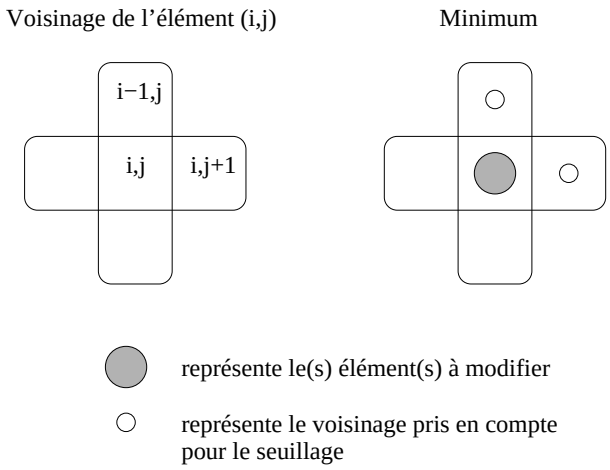


Fig. 2.17: Eléments voisins nécessaires au seuillage dans la matrice – Critère du minimum

Nous donnons ci-dessous un exemple de filtrage par le critère du minimum pour chaque type d'inversion.

La figure 2.18 présente une inversion droite et son filtrage. La matrice de gauche n'est pas robinsonienne ($2 < 5$). La matrice de droite est robinsonienne après le remplacement de la valeur concernée (5) par la valeur minimale (2).

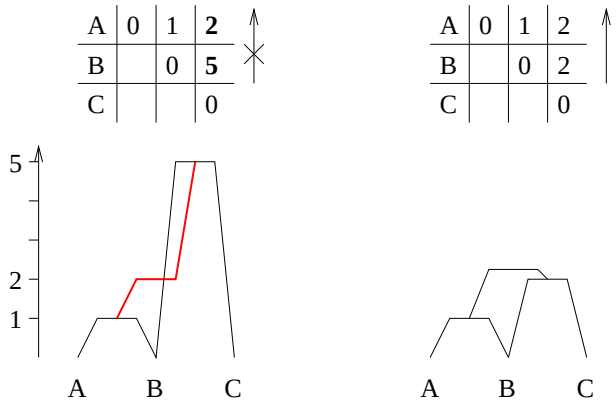


Fig. 2.18: Filtrage par seuillage local avec le critère du minimum – Inversion droite

La figure 2.19 page suivante présente une inversion gauche et son filtrage. La matrice de gauche n'est pas robinsonienne ($2 < 4$). La matrice de droite est robinsonienne après le remplacement de la valeur concernée (4) par la valeur minimale (2).

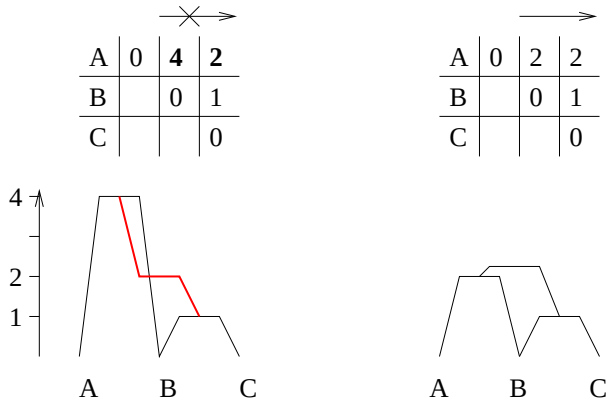


Fig. 2.19: Filtrage par seuillage local avec le critère du minimum – Inversion gauche

La figure 2.20 présente une inversion double et son filtrage. La matrice de gauche n'est pas robinsonienne ($2 < 4$ et $2 < 5$). La matrice de droite est robinsonienne après le remplacement des valeurs concernées (4 et 5) par la valeur minimale (2).

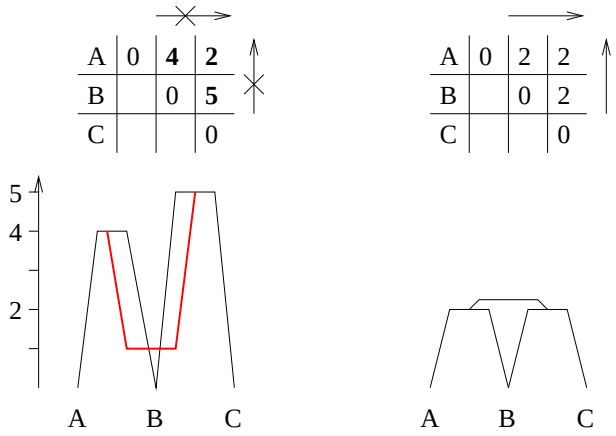


Fig. 2.20: Filtrage par seuillage local avec le critère du minimum – Double inversion

Algorithme de filtrage local avec le critère du minimum Au point de vue algorithmique, le changement le plus important est le parcours de la matrice : depuis le coin supérieur droit de la matrice, jusqu'à la première diagonale.

Listing 2.2 Seuillage local avec le critère du minimum

```
1: size = Card(D), i = size − 1 {initialisations}
2: while i > 1 do
3:    j = 0, k = i
4:    while i < size AND j < i do
5:       if i − j = k then {Parcours diagonale par diagonale}
6:          if D(i, j) > D(i − 1, j) OU D(i, j) > D(i − 1, j) then
7:             D(i, j) = min(D(i, j), D(i − 1, j), D(i, j + 1))
8:          end if
9:       end if
10:      j = j + 1, i = i + 1
11:   end while
12:   i = k, i = i − 1
13: end while
```

L'algorithme implémentant cette idée est donc le suivant : soit D la matrice triangulaire supérieure de taille n, $d_{i,j}$ l'élément pour lequel i et j sont les indices de ligne et colonne. Les éléments de la matrice sont explorés diagonale par diagonale en commençant par le coin supérieur droit et pour chaque élément, nous vérifions cette propriété : si $d_{i,j} \geq d_{i,j-1}$ ou $d_{i,j} \geq d_{i+1,j}$ alors $d_{i,j} = min(d_{i,j-1}, d_{i+1,j}, d_{i,j})$.

La preuve par récurrence de cet algorithme est similaire à celle énoncée pour le principe du maximum et par conséquent ne sera pas donnée.

Filtrage local avec les critères de la moyenne

Nous proposons de prendre en compte deux types de moyenne : la moyenne simple et la moyenne pondérée.

Soient D la matrice triangulaire supérieure de taille n, i et j les indices de ligne et colonne de D. On notera $d_{i,j}$ la valeur de l'élément repéré par i et j. Sur la figure 2.21 page suivante, cet élément est représenté avec les éléments voisins dans la matrice. Nous pouvons observer que dans le cas des moyennes, plusieurs éléments seront modifiés. Soit K la valeur prise par ces éléments. A droite, sont représentés les éléments pris en compte pour le seuillage.

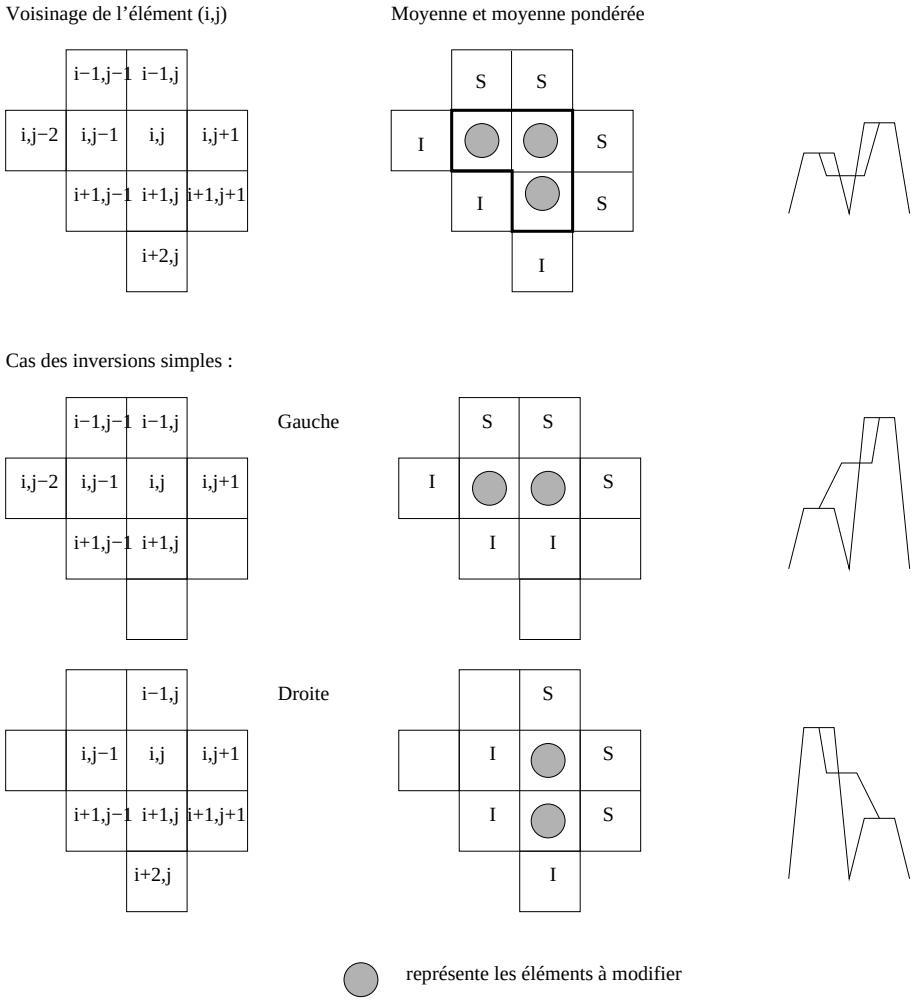


Fig. 2.21: Eléments voisins nécessaires au seuillage dans la matrice – Critères de moyenne et de la moyenne pondérée

On peut alors définir pour la moyenne :

$$K = moy(d_{i,j} + d_{i,j-1} + d_{i+1,j})$$

et pour la moyenne pondérée :

$$K = \frac{n_{i,j} * d_{i,j} + n_{i,j-1} * d_{i,j-1} + n_{i,j-1} * d_{i+1,j}}{(n_{i,j} + n_{i,j-1} + n_{i+1,j})}$$

en respectant les contraintes suivantes : $K \geq max(I)$ et $K \leq min(S)$. On notera $n_{i,j}$ le nombre d'éléments inclus dans l'élément (i, j).

Nous donnons ci-dessous un exemple de filtrage par le critère de la moyenne pour chaque type d'inversion.

La figure 2.22 présente une inversion droite et son filtrage. La matrice de gauche n'est pas robinsonienne ($2 < 5$). La matrice de droite est robinsonienne après le remplacement des valeurs concernées (2 et 5) par la valeur moyenne (3,5).

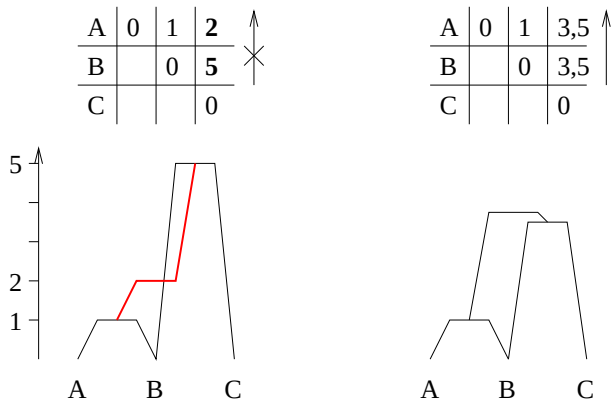


Fig. 2.22: Filtrage par seuillage local avec le critère de la moyenne – Inversion droite

La figure 2.23 présente une inversion gauche et son filtrage. La matrice de gauche n'est pas robinsonienne ($2 < 4$). La matrice de droite est robinsonienne après le remplacement des valeurs concernées (2 et 4) par la valeur moyenne (3).

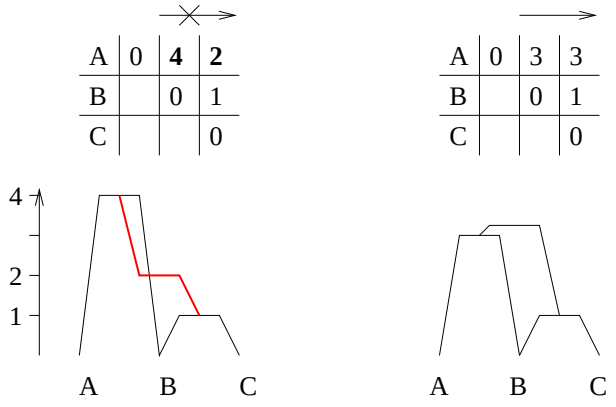


Fig. 2.23: Filtrage par seuillage local avec le critère de la moyenne – Inversion gauche

La figure 2.24 page ci-contre présente une inversion double et son filtrage. La matrice de gauche n'est pas robinsonienne ($2 < 4$ et $2 < 5$). La matrice de droite est robinsonienne après le remplacement des valeurs concernées (2, 4 et 5) par la valeur moyenne (3,6).

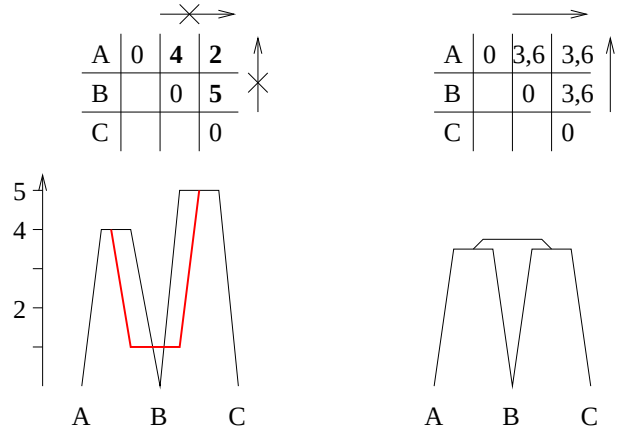


Fig. 2.24: Filtrage par seuillage local avec le critère de la moyenne – Double inversion

Cependant, en choisissant un exemple plus complexe, nous pouvons nous apercevoir que d'autres paramètres sont à prendre en compte pour définir les critères de la moyenne.

Pour rendre l'explication des différents paramètres plus claire, nous nous appuyons sur la pyramide avec inversion de la figure 2.25. Soit K la moyenne des valeurs des paliers A, B, C faisant partie de l'inversion traitée. Nous noterons d(A) la valeur du palier A. K ne peut pas prendre une valeur inférieure à celles de D, le successeur gauche de B, et du successeur droit de C sans créer une nouvelle inversion. De la même façon, K ne peut pas prendre une valeur supérieure à celles du prédecesseur gauche de B (s'il existait) et de E, le prédecesseur droit de C. Nous définissons alors deux ensembles I et S (*cf* figure 2.25), constituant le voisinage étendu des paliers A, B, C.

Deux cas de figure se présentent :

- Si $max(I) \leq moy \leq min(S)$, alors les paliers A, B et C prendront la valeur de la moyenne ;
- Sinon les paliers prendront la valeur de laquelle la moyenne s'approche le plus parmi : $max(I)$ et $min(S)$

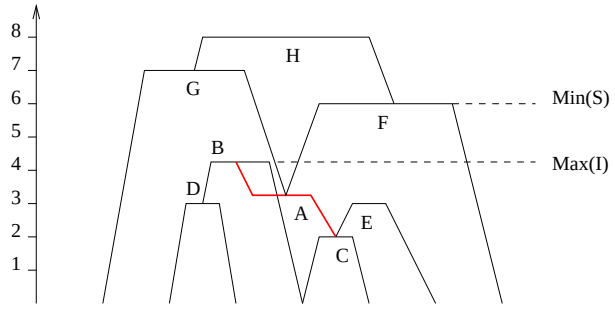


Fig. 2.25: Pyramide avec inversion complexe – Filtrage local avec le critère de la moyenne

Algorithme de filtrage local avec le critère de la moyenne Du point de vue algorithmique, le parcours de la matrice est le même que pour le critère du maximum, c'est à dire en débutant de la première diagonale. Les paramètres décrits ci-dessus garantissent de ne pas créer de nouvelles inversions par rapport aux diagonales préalablement traitées. Ainsi aucun back-tracking n'est nécessaire et la preuve par récurrence de l'algorithme est similaire à celle du critère du maximum. Elle ne sera donc pas redonnée ici. Les pages suivantes sont consacrées aux différents listings correspondant à ce critère, pour ne pas interférer avec la suite du manuscrit.

Listing 2.3 Seuillage local avec le critère de la moyenne

```
1: size = Card(D), i = 1 {initialisations}
2: while i < size − 1 do
3:    j = 0, k = i
4:    while i < size AND j < i do
5:       if i − j = k then {Parcours diagonale par diagonale}
6:          if D(i, j − 1) < D(i, j) AND D(i + 1, j) < D(i, j)  then {Double inversion}
7:             Fonction double_inversion(D, i, j). Algorithme 2.4 page suivante
8:          else if D(i + 1, j) < D(i, j) then {Inversion droite}
9:             Fonction inversion_droite(D, i, j). Algorithme 2.5 page 76
10:         else if D(i, j − 1) < D(i, j) then {Inversion gauche}
11:            Fonction inversion_gauche(D, i, j). Algorithme 2.6 page 77
12:         end if
13:      end if
14:      j = j + 1, i = i + 1
15:   end while
16:   i = k, i = i − 1
17: end while
```

Listing 2.4 Fonction Double_inversion(D,i,j)

```
1: moy = moy(D(i,j-1), D(i,j), D(i+1,j))
2: if i ≠ 0 AND j ≠ size − 2 then {Cas général}
3:    maxI = max(D(i,j-2), D(i+1,j-1), D(i+2,j))
4:    minS = min(D(i-1,j-1), D(i-1,j), D(i,j+1), D(i+1,j+1))
5: else if i = 0 AND j = size − 2 then {Coin supérieur}
6:    maxI = max(D(i,j-2), D(i+1,j-1), D(i+2,j))
7:    minS = moy
8: else if i = 0 then {Ligne supérieure}
9:    maxI = max(D(i,j-2), D(i+1,j-1), D(i+2,j))
10:   minS = min(D(i,j+1), D(i+1,j+1))
11: else {Colonne droite}
12:   maxI = max(D(i,j-2), D(i+1,j-1), D(i+2,j))
13:   minS = min(D(i-1,j-1), D(i-1,j))
14: end if
15: if moy ≤ minS AND moy ≥ maxI then
16:   D(i,j) = D(i,j-1) = D(i+1,j) = moy
17: else
18:   if |(moy − minS)| < |(moy − maxI)| then
19:      moy = minS
20:   else
21:      moy = maxI
22:   end if
23:   D(i,j) = D(i,j-1) = D(i+1,j) = moy
24: end if
```

Listing 2.5 Fonction Inversion_droite(D, i, j)

```
1: moy = moy(D(i,j), D(i+1,j))
2: if i ≠ 0 AND j ≠ size − 2 then {Cas général}
3:    maxI = max(D(i,j−1), D(i+1,j−1), D(i+2,j))
4:    minS = min(D(i−1,j), D(i,j+1), D(i+1,j+1))
5: else if i = 0 AND j = size − 2 then {Coin supérieur}
6:    maxI = max(D(i,j−1), D(i+1,j−1), D(i+2,j))
7:    minS = moy
8: else if i = 0 then {Ligne supérieure}
9:    maxI = max(D(i,j−1), D(i+1,j−1), D(i+2,j))
10:   minS = min(D(i,j+1), D(i+1,j+1))
11: else {Colonne droite}
12:   maxI = max(D(i,j−1), D(i+1,j−1), D(i+2,j))
13:   minS = D(i−1,j)
14: end if
15: if moy ≤ minS AND moy ≥ maxI then
16:   D(i,j) = D(i+1,j) = moy
17: else
18:   if |(moy − minS)| < |(moy − maxI)| then
19:      moy = minS
20:   else
21:      moy = maxI
22:   end if
23:   D(i,j) = D(i+1,j) = moy
24: end if
```

Le filtrage par seuillage local, décliné avec différents critères, est une approche heuristique efficace permettant de modifier localement une matrice pour la rendre robinsonienne. Nous allons à présent décrire une approche optimale de filtrage global utilisant un modèle de régression pour effectuer la même opération.

2.3.2 Filtrage par seuillage global par régression isotone

Dans cette partie, nous utilisons les régressions isotones afin d'obtenir une matrice robinsonienne. Les régressions isotones sont une extension du modèle classique de régressions par des fonctions isotones ou croissantes [Robertson et al., 1988]. Nous nous plaçons dans le cas d'un ensemble de données bivariées muni d'un ordre partiel puisque notre point de départ est une matrice de dissimilarité induite ordonnée d'après la pyramide associée.

Listing 2.6 Fonction Inversion_gauche(D, i, j)

```
1: moy = moy(D(i,j), D(i,j-1))
2: if i ≠ 0 AND j ≠ size − 2 then {Cas général}
3:    maxI = max(D(i,j−2), D(i+1,j−1), D(i+1,j))
4:    minS = min(D(i−1,j−1), D(i−1,j), D(i,j+1))
5: else if i = 0 AND j = size − 2 then {Coin supérieur}
6:    maxI = max(D(i,j−2), D(i+1,j−1), D(i+1,j))
7:    minS = moy
8: else if i = 0 then {Ligne supérieure}
9:    maxI = max(D(i,j−2), D(i+1,j−1), D(i+1,j))
10:   minS = D(i,j+1)
11: else {Colonne droite}
12:   maxI = max(D(i,j−2), D(i+1,j−1)D, (i+1,j))
13:   minS = min(D(i−1,j−1), D(i−1,j)
14: end if
15: if moy ≤ minS AND moy ≥ maxI then
16:   D(i,j−1) = D(i,j) = moy
17: else
18:   if |(moy − minS)| < |(moy − maxI)| then
19:      moy = minS
20:   else
21:      moy = maxI
22:   end if
23:   D(i,j−1) = D(i,j) = moy
24: end if
```

Régressions isotones sur un ensemble de données bivariées muni d'un ordre partiel

Cet ensemble est habituellement représenté par une matrice X de taille $a \times b$, dont les éléments sont indicés par i et j où $1 \leq i \leq a$ et $1 \leq j \leq b$. Un ordre partiel, noté $\lesssim$, associé à X est défini par :

$$(i,j) \lesssim (k,l) \Leftrightarrow \{i \leq k \; et \; j \leq l\}$$

Soit G la variable indépendante :

$$G = (g_{ij})$$

et w une fonction de pondération positive sur X.

Nous allons à présent définir l'ensemble des fonctions isotones sur X et les régressions isotones.

Définition 2.7 (Fonction isotone) *Une fonction f définie sur X est isotone si et seulement si :*

$$\forall (i,j) \in X \text{ et } \forall (k,l) \in X : (i,j) \lesssim (k,l) \Leftrightarrow f_{ij} \leq f_{kl}$$

Définition 2.8 (Régression isotone) *La régression isotone G^* de G est la solution du problème de minimisation suivant :*

$$\min_f \sum_{i=1}^{a} \sum_{j=1}^{b} [g_{ij} - f_{ij}]^2 w_{ij}$$

où f est une fonction isotone sur X.

Plusieurs algorithmes existent pour calculer une régression isotone dans ce cas. Pour des raisons de simplicité et d'efficacité, nous avons choisi SMOOTH [Dykstra and Robertson, 1982] : un algorithme itératif qui converge vers la solution optimale.

Algorithme de filtrage

Dans cette partie, nous montrons comment les régressions isotones nous permettent d'obtenir une matrice de Robinson à partir d'une matrice de dissimilarité ordonnée. Nous proposons tout d'abord un algorithme puis nous prouverons sa convergence.

Soit D la matrice de dissimilarité induite, une matrice triangulaire supérieure. Cette matrice est calculée à partir de l'algorithme de la CAP, les données sont ordonnées selon la pyramide. Nous avons modifié l'algorithme SMOOTH pour fonctionner sur une matrice de dissimilarité en respect d'un ordre partiel (sur les objets).

Ce nouvel algorithme, appelé SMOOTH[T] [Aude, 1999] , est décrit ci-dessous :

Etape 1 : Soit $\hat{G}^{(1)} = \hat{g}_{ij}^{(1)}$ la régression isotone de G sur les lignes, *i.e* $\hat{G}^{(1)}$ minimise

$$\sum_{i=1}^{n} \sum_{j=n-i}^{n} [g_{ij} - f_{ij}]^2 w_{ij}$$

avec $f_{(n-i)j} \leq \cdots \leq f_{nj}$ pour $j = 1 \ldots n$.

Soit $R^{(1)} = r_{ij}^{(1)} = (\hat{g}_{ij}^{(1)} - g_{ij})$ la première matrice "incrémentale des lignes".

Etape 2 : Soit $\tilde{G}^{(1)} = \tilde{g}_{ij}^{(1)}$ la régression isotone de $G + R^{(1)}$ sur les colonnes, *i.e* $\tilde{G}^{(1)}$ minimise

$$\sum_{i=1}^{n} \sum_{j=n-i}^{n} [g_{ij} + r_{ij}^{(1)} - f_{ij}]^2 w_{ij}$$

avec $f_{(n-i)j} \leq \cdots \leq f_{nj}$ pour $j = 1 \ldots n$.

Appelons $C^{(1)} = \tilde{G}^{(1)} - (G + R^{(1)})$ la première matrice "incrémentale des colonnes". On notera $\tilde{G}^{(1)} = G + R^{(1)} + C^{(1)}$.

Etape 3 : Au début de la k^{th} itérations, $\hat{G}^{(k)}$ est la régression isotone de $G+C^{(k-1)}$ sur les lignes. La k^{th} matrice "incrémentale des lignes" est définie par $R^{(k)} = \hat{G}^{(k)} - (G+C^{(k-1)})$ donc $\hat{G}^{(k)} = G + C^{(k-1)} + R^{(k)}$.

Ensuite, $\tilde{G}^{(k)}$ est la régression isotone de $G + R^{(k)}$ sur les colonnes. La k^{th} matrice "incrémentale des colonnes" est $C^{(k)} = \tilde{G}^{(k)} - (G + R^{(k)})$ ou, de façon équivalente, $\tilde{G}^{(k)} = G + R^{(k)} + C^{(k)}$.

Le lemme suivant permet de faire la preuve de la convergence vers la solution optimale.

Lemme 2.1 *Les deux suites $\hat{G}^{(k)}$ et $\tilde{G}^{(k)}$ convergent vers la régression isotone, G^*, en respectant l'ordre partiel des objets quand $k \to \infty$*

La définition suivante va être utilisée pour la démonstration.

Définition 2.9 (Matrices inscrites à droite et à gauche) *Soient $T = \{t_{ij}\}$ une matrice triangulaire de taille n et $G = \{g_{ij}\}$ une matrice carrée de taille $n \times n$. Alors T est "inscrite" dans la matrice G si l'une des conditions suivantes est vérifiée :*

$$j \geq (n+1-i) \Rightarrow g_{ij} = t_{ij} \qquad \textit{alors } T \textit{ est inscrite à droite de } G \ (T \trianglerighteq G)$$
$$j > (n+1-i) \Rightarrow g_{ij} = t_{ij} \qquad \textit{alors } T \textit{ est strictement inscrite à droite de } G \ (T \triangleright G)$$
$$j \leq (n+1-i) \Rightarrow g_{ij} = t_{ij} \qquad \textit{alors } T \textit{ est inscrite à gauche de } G \ (T \trianglelefteq G)$$
$$j < (n+1-i) \Rightarrow g_{ij} = t_{ij} \qquad \textit{alors } T \textit{ est strictement inscrite à gauche de } G \ (T \triangleleft G)$$

Preuve :

Soit D la matrice triangulaire de taille n, dont on veut calculer la régression isotone D^*. Nous allons montrer que la solution obtenue en appliquant l'algorithme SMOOTH est équivalente à celle obtenue avec cet algorithme.

1. On construit la matrice carré G, telle que $D \trianglerighteq G$ et $k \triangleleft G$, où $k = \min\{d_{ij}\}$.
2. Soit $\hat{G}^{(1)} = \{\hat{g}_{ij}^{(1)}\}$ (*resp.* $\hat{D}^{(1)}$) la matrice calculée à partir des régressions isotones des lignes de la matrice G (*resp.* D). On a alors $\hat{D}^{(1)} \trianglerighteq \hat{G}^{(1)}$.

   En effet, considérons la l^{e} ligne de la matrice G, l'ensemble de valeurs $\{g_{lj} \mid j < n-l+1\}$ est par définition isotone, et chacune de ces valeurs est inférieure ou égale à $g_{l,n-l+1}$. Par conséquent, pour tout $j < n-l+1$ on a $\hat{g}_{lj}^{(1)} = g_{lj} = k$, et pour $j \geq n-l+1$ on a $\hat{g}_{lj}^{(1)} = \hat{d}_{lj}^{(1)}$. Les régressions sur les lignes étant indépendantes, on a donc $\hat{D}^{(1)} \trianglerighteq \hat{G}^{(1)}$ et $k \triangleleft G$.
3. Sachant que $k \triangleleft G$ et $k \triangleleft \hat{G}^{(1)}$, on en déduit que dans la matrice incrémentale des lignes de G, $R^{(1)} = G - \hat{G}^{(1)}$, on a $0 \triangleleft R^{(1)}$.

4. On a $k \lhd G + R^{(1)}$. En appliquant le même raisonnement que précédement on montre que $\tilde{D}^{(1)} \trianglerighteq \tilde{G}^{(1)}$ et $k \lhd \tilde{G}^{(1)}$, et donc $0 \lhd C^{(1)}$ où $C^{(1)} = \tilde{G}^{(1)} - G + R^{(1)}$ est la matrice incrémentale des colonnes.
5. Par récurrence on obtient $\hat{D}^{(n)} \trianglerighteq \hat{G}^{(n)}$ et $\tilde{D}^{(n)} \trianglerighteq \tilde{G}^{(n)}$. Or, d'après le lemme 2.1 page précédente, les suites $\hat{G}^{(n)}$ et $\tilde{G}^{(n)}$ convergent vers G^* la régression isotone de G, donc $\hat{D}^{(n)}$ et $\tilde{D}^{(n)}$ convergent vers D^* la régression isotone de Ds. □

Grâce à l'emploi des régressions isotones, nous proposons un algorithme pour transformer une matrice de dissimilarité non robinsonienne en matrice de dissimilarité induite robinsonienne. La matrice obtenue après le fitrage global est plus proche au sens des moindres carrés de la matrice de distance initiale.

Les deux types de filtrage décrits dans cette section sont la première étape de la consolidation des pyramides. Ils permettent d'obtenir une matrice de Robinson à partir de la matrice de dissimilarité induite. Il faut ensuite recalculer la pyramide à partir de cette matrice.

2.3.3 Calcul d'une pyramide à partir d'une matrice robinsonienne

Dans cette partie, nous présentons un algorithme en deux étapes permettant de reconstruire une pyramide à partir d'une matrice de Robinson. Conformément au théorème de bijection [Diday, 1984a], nous pouvons calculer la pyramide associée à la matrice de Robinson. [Bertrand, 2000] a formalisé ce sujet par des définitions et théorèmes. L'idée principale repose sur le fait que la recherche de cliques maximales dans une matrice de Robinson conduit à obtenir les classes de la pyramide correspondante. Nous proposons un algorithme implémentant cette idée en utilisant les définitions et théorèmes établis par [Bertrand, 2000]. Nous les rappelons ici afin de définir ensuite notre algorithme.

Soit d la dissimilarité induite et Ω l'ensemble des individus.

Définition 2.10 *Soit $G(V, E)$, le graphe associé à la pyramide indicée où $V = \Omega$ est l'ensemble des noeuds et $E = \{(a, b) : a, b \in \Omega \times \Omega\}$ est l'ensemble des arcs. A chaque arête $e = (a, b) \in E$ est associé le poids $w(e) = d(a, b)$.*

Définition 2.11 *Le graphe-seuil $G_h(V, E_h)$ d'une dissimilarité d pour le seuil h admet V pour ensemble de noeuds et chacune de ses arêtes $e = (a, b)$ est tel que $d(a, b) \leq h$.*

La définition suivante caractérise un M_L-ensemble, qui peut être vu comme une clique maximale ou un sous-graphe complet dans un des graphes-seuil $G_h(V, E_h)$ associés à la pyramide.

Définition 2.12 *Un sous-ensemble lié maximal (M_L-ensemble) d'un graphe-seuil $G_h(V, E_h)$*

est un sous-ensemble U maximal (pour l'inclusion) de E_h possèdant la propriété que U est lié au niveau h au sens que $\forall (a,b) \in U$ implique $d(a,b) \leq h$. Soit V_U l'ensemble des noeuds de U alors les propriétés suivantes sont vérifiées :

- $\nexists (x,y) | x \in V_U, y \in V_U, (x,y) \notin U et d(x,y) \leq h$
- $\nexists z \in \{V/V_U\} | \forall x \in V_U, d(x,z) \leq h$

Soient $\widehat{M_L}(G_h(V,E_h)|h>0)$ la fermeture par intersections non-vides de l'ensemble $M_L(G_h(V,E_h)|h>0)$. De plus, nous définissons le diamètre de chaque sous-ensemble non-vide A (*resp.* $\widehat{A}$) de $M_L(G_h(V,E_h))$ (*resp.*$\widehat{M_L})(G_h(V,E_h))$ par $diam_d(A) = max\{d(a,b) : a,b \in A\}$ (*resp.* $\widehat{A}$) . Par exemple, sur la figure 2.26 page suivante, pour le graphe seuil G_3, les diamètres des cliques sont 2 et 3.

[Batbedat, 1990] a proposé le théorème suivant, écrit en utilisant la terminologie de [Bertrand, 2000] qui appelle les pyramides pré-pyramides fermées :

Théoreme 2.2 *Soit $\preceq$ un ordre linéaire, et d un coefficient de dissimilarité sur E. Les propriétés suivantes sont équivalentes :*

1. *$\preceq$ est compatible avec d ;*
2. *$P = (M_L(G_h(V,E_h)|h>0), diam_d)$ est une pré-pyramide indicée qui admet $\preceq$ comme ordre compatible ;*
3. *$\widehat{P} = (\widehat{M_L}(G_h(V,E_h)|h>0), diam_\delta)$ est une pré-pyramide fermée faiblement indicée qui admet $\preceq$ comme ordre compatible.*

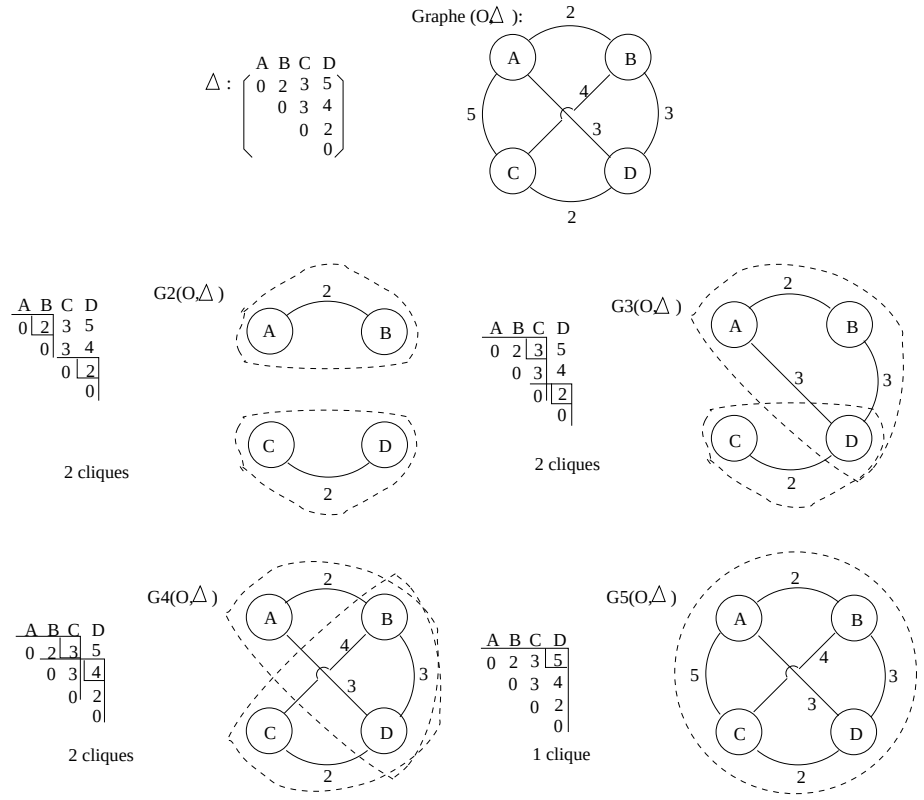


Fig. 2.26: Recherche des cliques maximales – Exemple pour un seuil h variant de 2 à 5

Les définitions et théorèmes permettent de définir formellement une pyramide à partir d'un graphe. Il existe une correspondance entre ce graphe et la matrice de Robinson à partir de laquelle nous voulons construire la pyramide. La matrice de Robinson est la matrice d'adjacence associée au graphe pondéré : les noeuds du graphe sont les individus et les poids associés aux arêtes sont les valeurs contenues par la matrice (*c.f.* figure 2.26 : la matrice Δ et le graphe $G(O, \Delta)$). Sur cette figure, les cliques sont représentées dans la matrice de Robinson par un triangle dont l'angle supérieur droit indique le niveau de la clique, inférieur ou égal au seuil (h).

L'algorithme (listing 2.7 page suivante) permettant de rechercher les cliques maximales dans des graphes-seuil est alors le suivant : soient D la matrice triangulaire supérieure de taille n et $d_{i,j}$ l'élément pour lequel i et j sont les indices de ligne et colonne. Les éléments de la matrice sont explorés par indices de ligne et colonne croissants et pour chaque élément, nous vérifions cette condition : si $d_{i,j} \neq d_{i,j-1}$ et $d_{i,j} \neq d_{i+1,j}$ alors nous sommes en présence d'une clique maximale (si $d_{i,j-1}$ et/ou $d_{i+1,j}$ n'existent pas, une seule inégalité est suffisante). En conséquence, la matrice est parcourue en une seule passe, tous seuils confondus.

D'après la figure 2.26, en suivant les itérations de l'algorithme énoncé ci-dessus, on obtient le tableau 2.2 page suivante.

$d_{i,j}$	$d_{i,j}$	$d_{i,j}$	clique	graphe-seuil
2	_	3	oui	G_2
3	_	5	oui	G_3,G_4
5	_	_	oui	G_5
3	3	4	non	
4	5	_	oui	G_4
2	4	_	oui	G_2,G_3

Tab. 2.2: Décomposition de l'algorithme de recherche de cliques maximales dans une matrice de Robinson (d'après la figure 2.26 page précédente). Chaque ligne représente une passe de l'algorithme.

Nous proposons donc un algorithme permettant de calculer la pyramide à partir de la matrice de Robinson en $\mathcal{O}(n^2)$, cette matrice étant triangulaire. En effet, la matrice de Robinson correspond à un graphe particulier. Nous ne nous situons donc pas dans le cas NP-complet de recherche de cliques maximales.

Listing 2.7 Recherche des cliques maximales

```
1: size = Card(D),k = size-1 {initialisation}
2: for i = size − 2 to 1 step −1 do {Cas général}
3:    for j = 1 to k − 1 step +1 do
4:       if D(i, j) ≠ D(i, j − 1) ET D(i, j) ≠ D(i + 1, j) then
5:          Création d'une clique
6:       end if
7:    end fork =k - 1
8: end for
9: for i = size − 2 to 1 step −1 do {Ligne supérieure}
10:   if D(0, i) ≠ D(0, i − 1) then
11:      Création d'une clique
12:   end if
13: end for
14: for i = 1 to size − 1 step +1 do {Colonne droite}
15:   if D(i, size − 1) ≠ D(i + 1, size − 1) then
16:      Création d'une clique
17:   end if
18: end for
```

Après cette première étape, nous obtenons une pyramide en recherchant les intersections par fermeture des intervalles définis par les cliques maximales. La complexité de cet algorithme est en $\mathcal{O}(n^2)$. En effet, chaque clique trouvée précédemment est une classe délimitant un intervalle sur l'ordre induit par la pyramide. Le niveau de chaque palier correspond au seuil h.

L'algorithme (listing 2.7 page précédente) permettant de rechercher les intersections des intervalles est donc le suivant : soient D la matrice triangulaire supérieure de taille n et $d_{i,j}$ un élément duquel i et j sont les indices de ligne et colonne. Il faut préciser que la présence d'une clique a été repérée lors de l'étape précédente dans la matrice. Les éléments de la matrice sont explorés par indices de ligne et colonne croissants et pour chaque élément $d_{i,j}$, nous vérifions cette condition : si sur la ligne i, une clique maximale est présente et si sur la colonne j, une clique maximale est présente , alors nous sommes en présence d'une intersection de deux intervalles. En conséquence, la matrice est parcourue en une seule passe.

Ces deux étapes sont illustrées par un exemple (*cf* figure 2.27 page suivante).

En résumé, nous proposons donc un algorithme complet en deux étapes :

1. **Recherche de cliques maximales** : En explorant successivement tous les graphes-seuil, nous obtenons directement les cliques maximales. Sur la figure 2.26 page 82, nous pouvons voir que la détection des cliques maximales se fait simplement à partir de la matrice de Robinson.
2. **Fermeture des intervalles par intersection** : par intersection de toutes les cliques maximales, nous obtenons toutes les classes de la pyramide. En fait, chaque clique trouvée lors de la première étape est une classe délimitant un intervalle sur l'ordre induit par la pyramide.

[Diday, 1984a] a énoncé plusieurs règles concernant la suppression de classes inutiles dans une pyramide. Il établit que ces arcs (ou arêtes) surviennent quand des égalités apparaissent de façon consécutive dans la matrice de Robinson.

Soient D la matrice triangulaire supérieure de taille n et $d_{i,j}$ la valeur de l'élément pour lequel i et j sont les indices de ligne et colonne. La figure 2.28 représente la règle générale de suppression, la flèche barrée indiquant l'arêtes à supprimer.

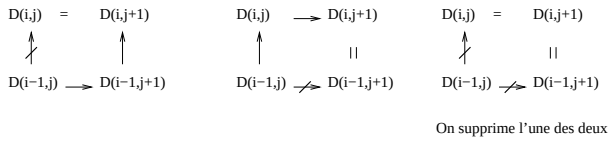


Fig. 2.28: Critères de suppression d'arêtes – Règle générale

Ce biais n'apparait pas dans notre façon de construire la pyramide. En effet, la recherche de cliques maximales permet d'éviter ce biais. En reprenant l'exemple de [Diday, 1984a] pour illustrer ce point (*cf* figure 2.29 page ci-contre), nous pouvons voir que les paliers supprimés à l'aide de ces critères ne sont pas construits par notre approche.

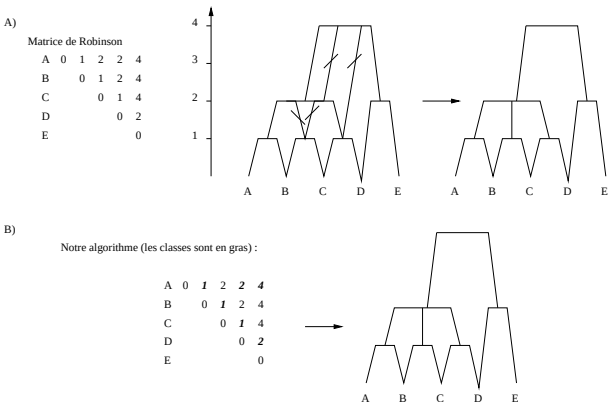


Fig. 2.29: Suppression des arêtes inutiles – A. Approche de Diday. B. Notre approche

Matrice de Robinson :

A	B	C	D	E	F	G
	1	1	2	3	3	5
		1	2	3	3	4
			1	3	3	4
				1	2	2
					1	2
						1

A	B	C	D	E	F	G
	1	**1**	**2**	3	**3**	**5**
		1	2	3	3	**4**
			1	3	3	4
				1	2	**2**
					1	2
						1

Cliques :
- de seuil 0 : tous les singletons
- de seuil 1 : {A,B,C}, {C,D},{D,E}, {E,F}, {F,G}
- de seuil 2 : {A,B,C,D},{D,E,F,G}
- de seuil 3 : {A,B,C,D,E,F}
- de seuil 4 : {B,C,D,E,F,G}
- de seuil 5 : {A,B,C,D,E,F,G}

A	B	C	D	E	F	G
	1	**1**	**2**	3	**3**	**5**
		$\hat{1}$	**$\hat{2}$**	3	**$\hat{3}$**	**4**
			1	3	3	4
				1	**$\hat{2}$**	**2**
					1	2
						1

Paliers :
- de seuil 0 : tous les singletons
- de seuil 1 : {A,B,C}, {C,D},{D,E}, {E,F}, {F,G}, {B,C}
- de seuil 2 : {A,B,C,D},{D,E,F,G},{B,C,D}, {D,E,F}
- de seuil 3 : {A,B,C,D,E,F},{B,C,D,E,F}
- de seuil 4 : {B,C,D,E,F,G}
- de seuil 5 : {A,B,C,D,E,F,G}

Pyramide :

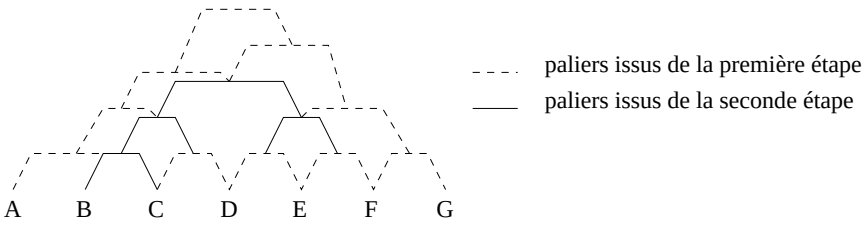


Fig. 2.27: D'une matrice de Robinson à une pyramide – Les cliques sont recherchées dans la matrice de Robinson (indiquées en gras). Puis la fermeture des intervalles formés par les cliques est calculée (indiqué en gras avec un accent circonflexe). Nous disposons alors de tous les paliers nécessaire à la construction de la pyramide.

Nous proposons donc un algorithme en deux étapes reconstruisant directement une pyramide à partir d'une matrice robinsonienne.

L'application de l'algorithme de classification ascendante pyramidale introduit un biais dans certains cas identifiés dans cette section : des inversions peuvent apparaître lors de la construction de la pyramide. Ceci est caractérisé (en regardant l'indice pyramidal) par le fait que la matrice de dissimilarité induite correspondant à la pyramide n'est pas de Robinson. Nous avons proposé une méthodologie en deux étapes afin de corriger ce biais. Tout d'abord, nous proposons deux types de filtrage : l'un est local et utilise une heuristique, l'autre est global et est optimal. Ils permettent de rendre robinsonienne une matrice tout en conservant la contrainte d'ordre sur les objets. La seconde étape consiste à reconstruire une pyramide à partir d'une matrice de Robinson. Nous disposons donc à présent d'une méthode complète pour obtenir une pyramide sans inversion. Ces différents algorithmes ont été implémentés et intégrés à une plateforme d'analyse de données biologiques définies par des distances, appelée DaTool. Ce projet est décrit dans l'annexe A page 171.

2.3.4 Comparaison des algorithmes de filtrage

Dans ce paragraphe nous allons comparer les deux méthodes de filtrage que nous venons de proposer pour résoudre le problème des inversions.

Comparaison des algorithmes

Comparaison théorique Le filtrage local correspond à un remplacement de chaque valeur "erronée" par une valeur dépendant de son voisinage. Dans le pire des cas, une modification d'un élément sur une diagonale explorée en début de filtrage peut avoir une forte incidence sur toutes les diagonales parcourues ultérieurement. L'algorithme pour ce type de filtrage est itératif. Cependant, en moyenne, les modifications de la matrice sur laquelle est appliquée le filtrage seront peu nombreuses ; en particulier dans le cas du minimum et du maximum où seule une valeur est modifiée à chaque passe.

Lorsque l'on applique le filtrage global par régression isotone, on cherche la matrice de Robinson la plus proche au sens des moindres de carrés de la matrice de départ. C'est à dire que l'on optimise le critère correspondant à D_2. Les modifications de la matrice ne sont plus uniquement locales mais concernent toute la matrice. Elles ne dépendent pas du voisinage propre de chaque valeur mais de la totalité de la matrice.

Complexité Une différence importante entre les deux solutions de filtrage concerne la complexité de ces algorithmes.

Dans le cas du filtrage local, l'implémentation est triviale et la complexité algorithmique est en $\mathcal{O}(n^2)$. Pour chacun des critères, la matrice de dissimilarité induite n'est parcourue qu'une seule fois. Par ailleurs, la complexité en terme d'occupation mémoire ne varie quasiment pas (i.e. aucune ressource mémoire supplémentaire n'est requise hormis les opérations de l'algorithme), le filtrage étant directement opéré sur la matrice de dissimilarité induite (en $\mathcal{O}(n^2)$).

En revanche, l'utilisation des régressisons isotones pour le filtrage global induit une complexité algorithmique et d'occupation d'espace mémoire bien supérieure. Ces niveaux de complexité sont directement liés à l'implémentation retenue. Dans notre cas nous avons utilisé l'algorithme SMOOTH obtenant la solution par un processus de convergence, et dont la complexité en espace mémoire en $\mathcal{O}(n^2)$. L'utilisation d'implémentations plus performantes (e.g. SIBC [Qian and Eddy, 1995]), après une adaptation au cas des matrices de distances, permettrait de réduire sensiblement ces niveaux de complexité.

Validation par simulation

Pour valider les différentes approches, nous avons procédé à plusieurs simulations. Notre objectif est de mettre en évidence la pertinence de ces méthodes en fonction des paramètres suivants : la taille du jeu de données, le type de filtrage, et pour le filtrage local, le critère d'agrégation utilisé. Avec la CAP, les paliers inversés dans une pyramide n'apparaissant que dans les cas autres que le lien complet, les simulations ont été réalisées pour le lien simple et pour la moyenne. Pour chaque simulation nous mesurons le temps CPU (en secondes) et les valeurs des critères D_1 et D_2 (voir la partie 1.3.3 page 41) pour estimer l'adéquation entre les différentes matrices. Les résultats numériques ayant permis de tracer les figures présentées dans cette section sont regoupés à l'annexe B page 185, de même que les courbes obtenues avec le critère du lien simple.

Temps de calcul et complexité En observant les figures 2.30 page suivante et B.1 page 185, nous pouvons voir immédiatement que le seuillage local par les différents critères est réalisé dans un temps raisonnable, par rapport au temps initial nécessaire pour calculer une pyramide. Au contraire, le filtrage par régression isotone nécessite un temps de calcul complémentaire, croissant de façon exponentielle avec la taille du jeu de données, du à sa complexité. Le filtrage par régression isotone est donc moins adapté à des volumes de données importants. Ces résultats correspondent aux complexités algorithmiques énoncées ci-dessus.

Mesures de la distorsion En observant la figure 2.31 page 89 qui mesure le degré de linéarité entre les deux matrices (D_1), chacun des filtrages apporte une amélioration. Les matrices issues de ces filtrages sont plus corrélées à la matrice initiale que la matrice induite par la CAP. Le

filtrage global par régression isotone et le filtrage local avec le critère du minimum sont très proches et présentent les meilleurs résultats.

En observant la figure 2.32 page 90 qui mesure la distance entre les deux matrices (D_2), tous les filtrages n'offrent pas les mêmes résultas. Seuls le filtrage global par régression isotone et le filtrage local avec le critère du minimum améliorent la CAP.

Les deux méthodes à retenir sont donc le le filtrage global par régression isotone et le filtrage local avec le critère du minimum. Lorsque l'on applique directement le filtrage par régression isotone sur la matrice initiale ordonnée d'après l'ordre de la pyramide (regiso_ini sur la figure 2.32 page 90), la matrice obtenue est la plus proche de la matrice de distance initiale. Une des perspectives de ce travail est donc de déterminer une approche ordonnant les données initiales et d'appliquer directement les régressions isotones sans l'étape préalable de la CAP.

Nous pouvons remarquer que les valeurs des critères D_1 et D_2 sont d'ordre de grandeur différents entre le critère du lien simple (minimum) et le critère de la moyenne, notamment pour les jeux de données les plus importants. Cela est du au nombre de paliers inversés, qui est

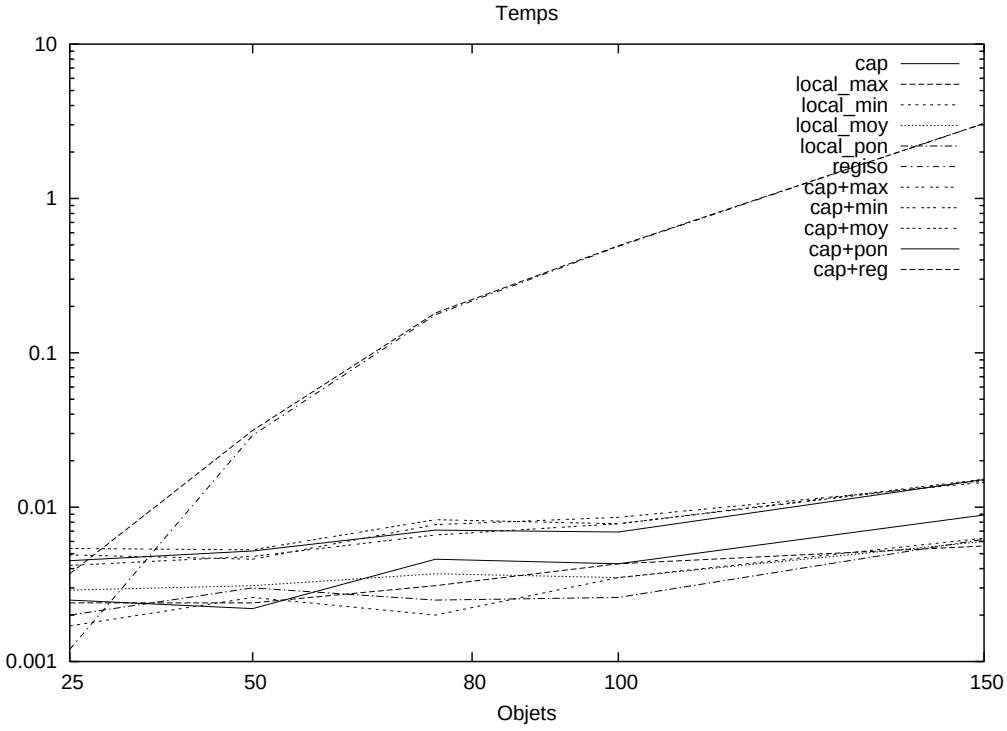


Fig. 2.30: Temps de calcul *cap* : classification ascendante pyramidale, *local_min* : filtrage local avec le critère du minimum, *local_max* = filtrage local avec le critère du maximum, *local_moy* : filtrage local avec le critère de la moyenne, *local_pon* : filtrage local avec le critère de la moyenne pondérée regiso : filtrage global par régression isotone, cap+*critère*

plus grand dans le cas du lien simple. On obtient donc des valeurs moins bonnes dans ce cas.

En conclusion, les deux approches permettent effectivement de supprimer les paliers inversés d'une pyramide. En revanche, on constate que le filtrage local (sauf le critère du minimum) donne de moins bons résultats, considérant un ordre identique sur les objets, que le filtrage par régression isotone. Cependant, ce gain qualitatif a un coût non négligeable au niveau de la complexité. Par conséquent, le choix du type de filtrage à utiliser dépendra principalement du contexte. Dans le cas d'une utilisation intensive de la classification pyramidale sur d'importants volumes de données on sera probablement amené à retenir le filtrage local. Inversement, on utilisera de préférence le filtrage par régression isotone pour les pyramides de taille plus réduite.

Application à une famille de protéines Nous avons choisis la famille 123, qui est une famille de protéines de la base de données Phytoprot pour illustrer ces filtrages sur un exemple réel.

La première figure 2.33 page 92 présente la pyramide avec inversions obtenue à partir de l'algorithme de la CAP. Les inversions sont indiquées en rouge. Avec ces inversions, l'interprétation

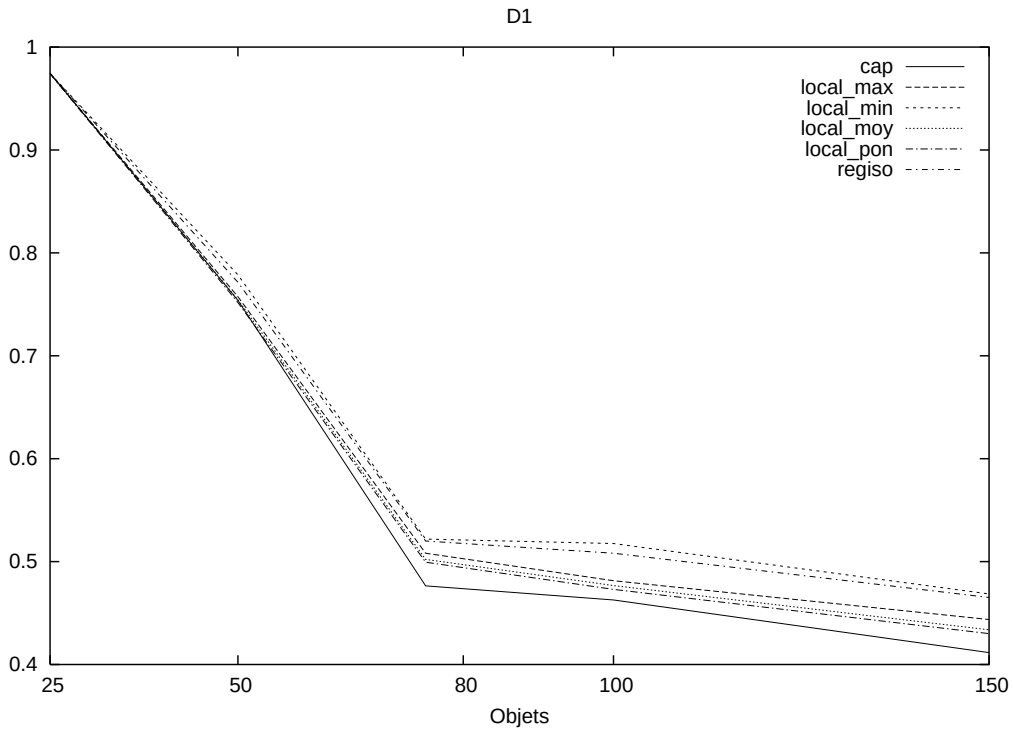


Fig. 2.31: D1 – cap : classification ascendante pyramidale, local_*critère* : filtrage local avec différents critères (min = minimum, max = maximum, moy = moyenne, pon = moyenne pondérée), regiso : filtrage global par régression isotone

est difficile.

La seconde figure 2.34 page 93 présente la pyramide sans inversion obtenue avec l'algorithme basé sur un filtrage local avec le critère du maximum et la reconstruction de la pyramide par notre méthode. L'ordre des données est conservé. La structure de la pyramide semble simplifiée par rapport à la première figure. Nous avons perdu l'information des proximités des groupes de séquences.

La troisième figure 2.35 page 94 présente la pyramide sans inversion issue d'un filtrage global par régression isotone et de la sreconstruction de la pyramide par notre méthode. L'ordre des données est conservé. La structure est très proche de celle de la première figure.

Par exemple, si nous considérons les protéines AT5G23590 et AT5G06110 en bas des trois figures : sur la première et la troisième figures, ces protéines sont proches des protéines voisines et sur la seconde figure, elles sont aussi éloignées de ces protéines que de toutes les autres. Les groupes de protéines sont structurés de la même façon sur la première et la troisième figures et sont différents sur la seconde (par exemple le groupe de protéines AT1G77020, AT1G76700, AT1G21080, AT4G39150, AT2G21510).

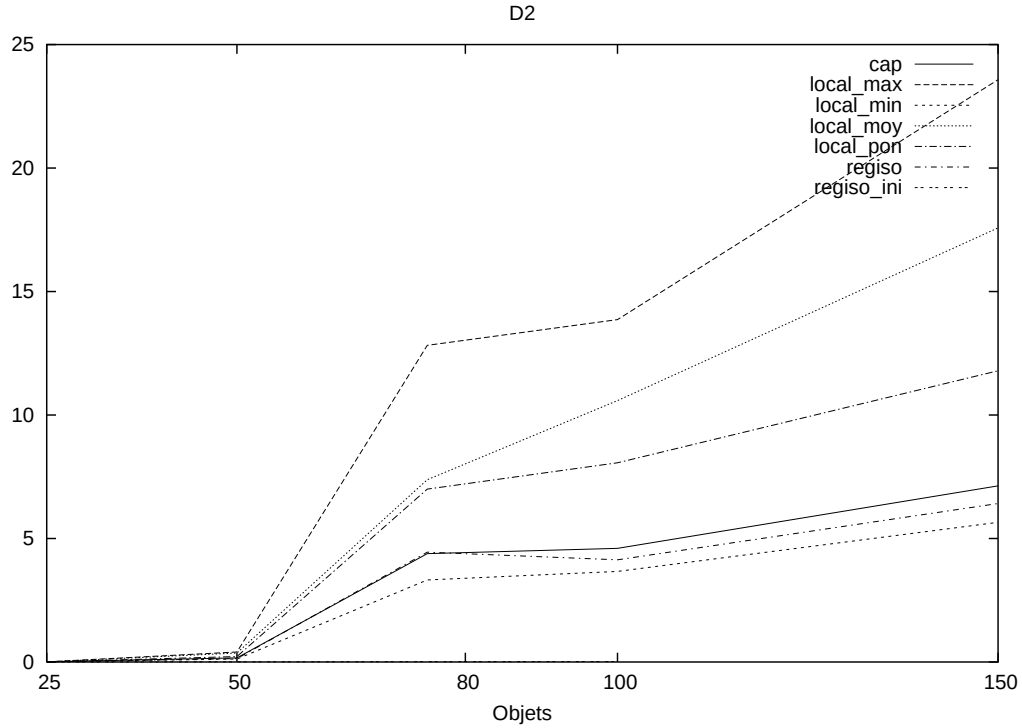


Fig. 2.32: D2 – cap : classification ascendante pyramidale, local_*critère* : filtrage local avec différents critères (min = minimum, max = maximum, moy = moyenne, pon = moyenne pondérée), regiso : filtrage global par régression isotone

Critère	CAP	Filtrage local (max)	Filtrage global
D1	0.563	0.609	0.638
D2	2.310	6.316	0.465

Tab. 2.3: Tableau regroupant les résultats obtenus pour les critères D1 et D2 par les différentes méthodes testées sur la famille 123 de Phytoprot.

Ces observations sont vérifiées numériquement à l'aide des critères D1 et D2 présentés dans le tableau 2.3. Le filtrage global par régression isotone obtient les meilleurs résultats pour ces deux mesures de la distorsion de la dissimilarité initiale.

Pour conclure, la pyramide obtenue avec le filtrage local est plus structurée mais moins précise que celle obtenue avec le filtrage par régression isotone. Ce filtrage conserve la structure originale des données et permet ainsi une meilleure interprétation des groupes de séquences et de leurs proximités.

2.4 Conclusion

Les méthodes de classification jouent un rôle proéminent dans les domaines d'application traitant d'importants volumes de données. Nous avons présenté ici la classification pyramidale qui est particulièrement adaptée à la complexité des données biologiques. En effet, ses propriétés supplémentaires, par rapport à la classification hiérarchique, permettent d'obtenir des représentations plus proches des données initiales. La Classification Ascendante Pyramidale est l'algorithme le plus fréquemment utilisé pour calculer une pyramide. Cependant, un biais de construction important était induit par cet algorihtme : la présence de paliers inversés, due à une matrice de dissimilarité induite non robinsonienne.

Dans ce chapitre, pour résoudre ce problème, nous avons proposé une solution optimale avec un filtrage global et une approche heuristique, avec le filtrage local. Le filtrage global est réalisé par régression isotone. Nous utilisons pour cela l'algorithme SMOOTH qui converge vers la solution optimale. Le filtrage local est décliné avec différents critères (minimum, maximum, moyenne, moyenne pondérée) et consiste au remplacement local des valeurs. Ces deux approches permettent d'obtenir une dissimilarité induite de Robinson, en accord avec l'extension du théorème de Johnson-Benzecri. Comme les simulations l'ont montré, l'utilisation de telle ou telle méthode dépend du volume des données de départ, du résultat attendu et des moyens informatiques à disposition. Nous avons également présenté un algorithme en deux étapes permettant de reconstruire une pyramide à partir de la matrice de Robinson obtenue par filtrage. Cette méthode repose sur la recherche de cliques maximales dans un graphe particulier, les cliques étant les paliers de la pyramide.

Nous disposons donc à présent d'une méthode de classification fiable et robuste, offrant de nombreux avantages : représentation de classes empiétantes et ordre partiel des individus. Ses domaines d'application en biologie sont donc à étendre, par exemple dans l'analyse du transcriptome. Méthodologiquement, nous avons décidé de l'appliquer à l'alignement multiple de séquences, et plus particulièrement à l'approche progressive qui utilise une structure de guidage pour déterminer l'ordre des séquences à aligner.

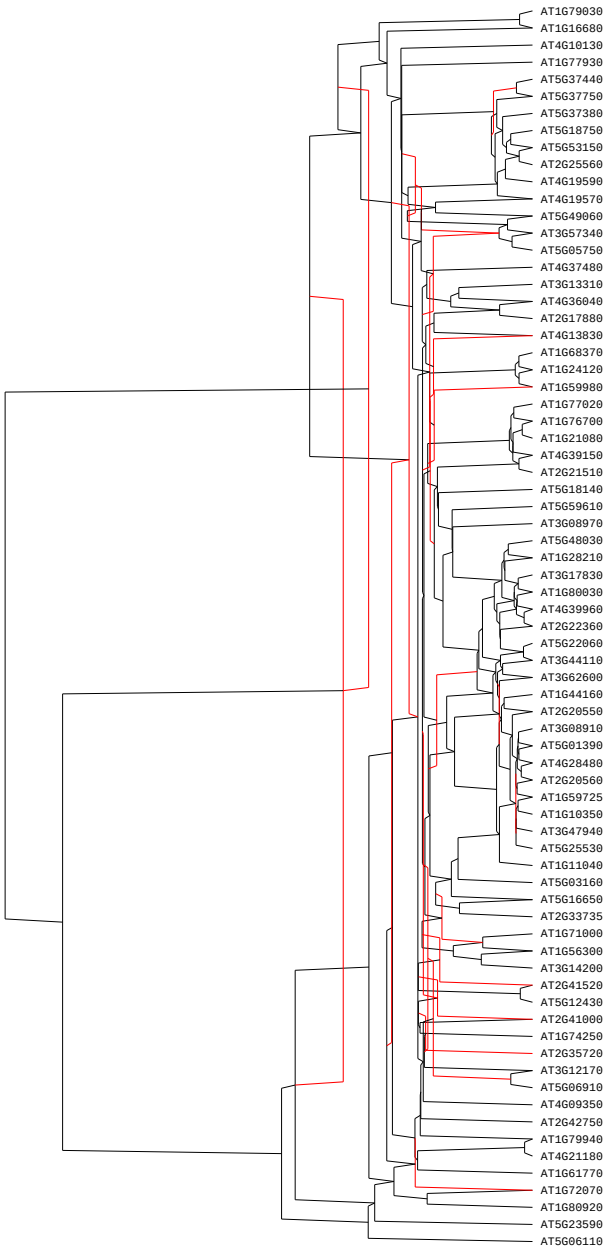


Fig. 2.33: Pyramide avec inversions (en rouge) obtenue à partir de l'algorithme de CAP sur la famille de protéines 123 (Phytoprot).

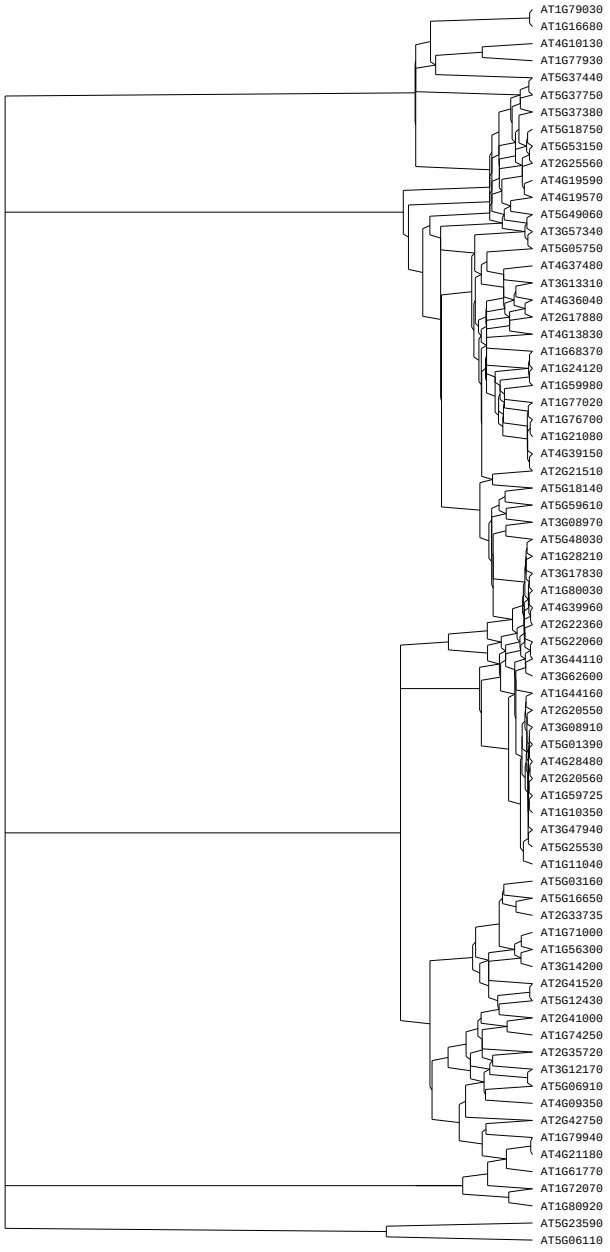


Fig. 2.34: Pyramide obtenue avec le filtrage local (critère du maximum) sur la famille de protéines 123 (Phytoprot).

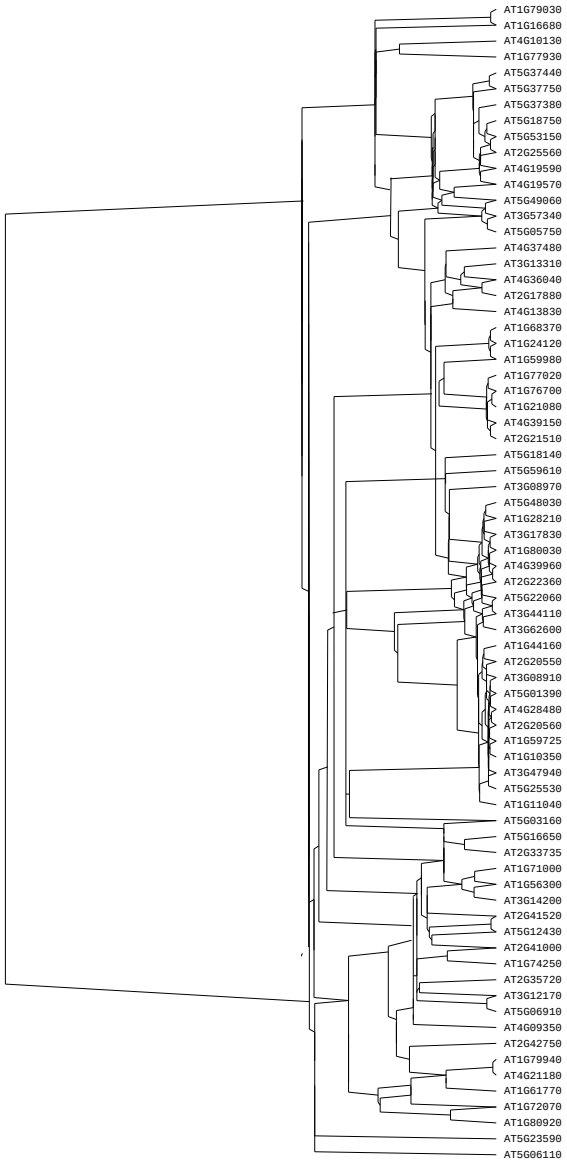


Fig. 2.35: Pyramide obtenue avec le filtrage global par régression isotone sur la famille de protéines 123 (Phytoprot)

Chapitre 3

Alignement de séquences

3.1 Introduction

La première partie de ce travail a été consacrée aux méthodes de construction d'arbres et plus particulièrement à la classification pyramidale dans le contexte de l'application à des données biologiques. Le biais induit par l'algorithme de la CAP ayant été corrigé, nous disposons à présent d'une méthode de classification fiable et robuste. Nous l'appliquons à l'alignement de séquences.

Une des premières questions que l'on se pose à propos d'un gène (ou d'une protéine) est si celui-ci (ou celle-ci) est lié(e) ou non avec un autre gène (ou une autre protéine). Une ressemblance au niveau de la séquence suggère tout d'abord que deux protéines sont homologues mais aussi qu'elles peuvent partager des fonctions identiques. Le nombre de séquences (protéiques ou nucléiques) étant très important, il est possible d'identifier des domaines ou motifs communs à un groupe de molécules. Ces analyses sont réalisées grâce à l'alignement de séquences.

L'alignement de séquences protéiques est souvent plus informatif qu'un alignement de séquences nucléiques. Les changements dans la séquence d'ADN en 3ème position d'un codon ne changent généralement pas l'acide aminé correspondant. La redondance du code génétique est importante : il existe 4^3 triplets de nucléotides possibles pour seulement 20 acides aminés et 3 non-sens (ou stop) (voir la figure 3.2). La première colonne correspond à la première lettre du codon, la première ligne à la seconde lettre, l'intérieur des cases correspond à la troisième lettre et à la traduction du codon en protéine. C'est pourquoi, nous nous restreignons au cadre des protéines.

L'alignement de séquences permet d'identifier les mutations qui ont eu lieu lors de l'évolution. Ces évènements sont à l'origine de la divergence des séquences. Les mutations sont au nombre de trois : les substitutions, les insertions et les délétions. Une substitution est le remplacement

d'un acide aminé par un autre. Dans un alignement, cela sera représenté par deux résidus non identiques alignés, par exemple une glutamine et une asparagine. Les insertions et délétions apparaissent lors de l'ajout ou le retrait d'un (ou plusieurs) résidu(s) dans la séquence. Elles sont symbolisées par le caractère blanc (- ou .). On les appelle gaps ou brèches dans un alignement.

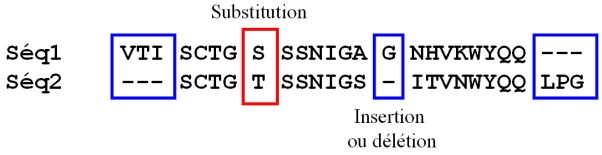


Fig. 3.1: Les différentes mutations à l'origine de la divergence des séquences

Dans ce chapitre, nous présenterons tout d'abord les méthodes permettant de mesurer, de donner un score à ces mutations. Ce score nous permettra alors de déterminer le score d'un alignement, défini comme la somme des scores des évènements mutationnels (illustrés sur la figure 3.1) étant survenus entre deux séquences. Ensuite, nous décrirons les alignements de deux séquences et enfin l'alignement multiple de séquences.

3.1.1 Les matrices de substitution

Les matrices de substitution, représentées usuellement sous forme de tableau de dimension $20 * 20$ (20 acides aminés), permettent d'associer un coût à chaque substitution possible. Elles fournissent une estimation du taux de changements entre deux résidus.

Les acides aminés trouvés dans les séquences protéiques sont au nombre de vingt. Ils diffèrent par les propriétés physico-chimiques de leur chaîne latérale telles que le volume, la charge, l'hydrophobicité... Chaque résidu d'une protéine aura donc un rôle dépendant de ces propriétés. Pour cette raison, bien que les substitutions soient des phénomènes aléatoires, leurs sélections sont régies par de nombreuses contraintes liées à la structure tridimensionnelle et la fonction des protéines.

Les deux prochains paragraphes sont consacrés aux deux types de matrices les plus fréquemment citées. De nombreuses autres matrices existent, souvent obtenues selon des méthodes voisines [Gonnet et al., 1992, Jones et al., 1992].

Les matrices PAM (Point Accepted Mutation) Les matrices PAM ont été créées par [Dayhoff et al., 1979] au NRBF (National Biomedical Research Foundation) dans les années 70. Un alignement de séquences a été réalisé manuellement à partir de 71 familles de protéines très similaires (plus de 85% d'identité). Ce travail est fiable et surtout facile à réaliser du fait de la grande ressemblance entre celles-ci. Ils ont pu alors mesurer la fréquence de chaque substitution

observée. Ces premiers calculs ont permis de créer la matrice PAM1. Les scores représentés dans cette matrice correspondent à une mutation ponctuelle pour une séquence de 100 acides aminés. Les matrices, obtenues en multipliant la matrice PAM1 par elle-même, vont de 1 à 500 PAM. Ce calcul est basé sur l'hypothèse de l'indépendance des substitutions entre elles, dont on sait qu'elle est erronée. Une seconde critique peut être faite aux matrices PAM. Ce sont des matrices symétriques : la probabilité du remplacement d'un acide aminé A par un acide aminé B est égale à la probabilité du remplacement de B par A (par exemple, le remplacement d'un acide aminé volumineux par un plus petit).

Il est nécessaire de transformer ces matrices de substitutions en matrices de score (aussi appelées *log-odd* matrices) pour les utiliser dans les algorithmes d'alignement par paire ou la recherche dans les banques.

A partir d'une matrice de substitution, le score S pour l'alignement des acides aminés i et j est :

$$S(i,j) = 10\ log_{10}\ (P_{ij}/(P_iP_j))$$

où P_{ij} est la probabilité que la paire de résidus (i,j) représente un alignement réel et la fréquence normalisée $P_{i(j)}$ représente la probabilité que le résidu $i(j)$ soit aligné par chance.

Si ce score est égal à 1, l'acide aminé est neutre. Si ce score est supérieur (resp. inférieur) à 1, la substitution est conservative (resp. non-conservative), c'est à dire que l'acide aminé est sélectionné (resp. contre sélectionné).

1 \ 2	T	C	A	G
T	TC PHE AG LEU	TCAG SER	TC TYR AG STOP	TC CYS A STOP G TRP
C	TCAG LEU	TCAG PRO	TC HIS AG GLN	TCAG ARG
A	TC ILE AG MET	TCAG THR	TC ASN AG LYS	TC SER AG ARG
G	TCAG VAL	TCAG ALA	TC ASP AG GLU	TCAG GLY

Fig. 3.2: Le code génétique

Les matrices de score BLOSUM (BLOcks SUbstitution Matrix) Les matrices BLOSUM [Henikoff and Henikoff, 1992] sont une alternative aux matrices PAM. Elles reposent également sur l'hypothèse de l'indépendance des sites. Les auteurs ont utilisé la base de données BLOCKS, contenant 500 groupes d'alignements locaux de séquences protéiques. Les matrices ont été déduites des segments conservés des protéines. Le principe de construction est de rassembler en une séquence toutes les séquences partageant plus d'un certain taux de similarité. Par exemple, la matrice BLOSUM 62 fusionne en une seule séquence toutes les séquences présentant plus de 62% d'identité. Autrement dit, elle est construite à partir de séquences présentant moins de 62% d'identité et, à ce titre, devrait être bien adaptée pour aligner de telles séquences.

Comparaison des matrices PAM et BLOSUM Plusieurs études comparatives ont été réalisées entre ces matrices performantes, largement utilisées pour réaliser des recherches dans les banques ou des alignements. La figure 3.3 établit la correspondance entre ces deux types de matrice. La plupart des études ont montré que les matrices BLOSUM donnaient de meilleurs résultats que les matrices PAM. Deux raisons expliquent ce résultat :

- ces matrices sont réalisées à partir d'alignements locaux (régions structurellement conservées) alors que les matrices PAM incluent des régions très divergentes
- les matrices BLOSUM sont plus récentes et donc ont bénéficié d'un plus grand nombre de données à disposition.

Les algorithmes de recherche dans les banques FASTA [Lipman and Pearson, 1985] et BLAST (Basic Local Alignment Search Tool) [Altschul et al., 1990], ainsi que le package gcg, utilisent par défaut la matrice BLOSUM 62.

3.1.2 Les pénalités

Les substitutions ne sont pas les seuls évènements pouvant survenir au cours de l'évolution. Des insertions et des délétions apparaissent également : deux séquences peuvent contenir des portions non liées et donc impossible à aligner. Il est difficile de déterminer si l'évènement originel ayant eu lieu est une insertion ou une délétion. C'est pourquoi le terme d'*indel* est

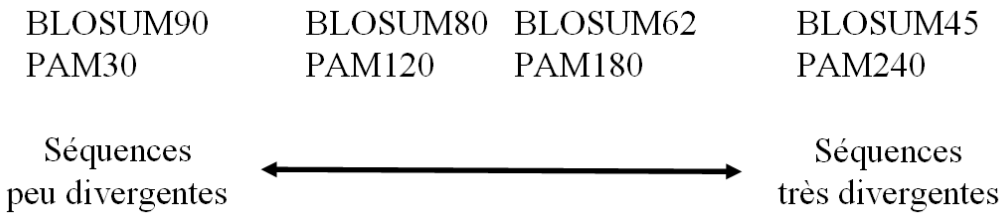


Fig. 3.3: Correspondance entre les matrices PAM et BLOSUM

fréquemment utilisé.

Contrairement aux substitutions qui ont pu être modélisées grâce au comptage manuel des fréquences des acides aminés, il n'existe pas de modèle biologique pour les indels. Afin de donner un score à ce phénomène, deux types de pénalité ont été définis : une pénalité pour l'ouverture d'une brèche et une pénalité pour l'extension d'une brèche existante. Ce sont les paramètres les plus difficiles à ajuster lorsque l'on aligne des séquences. En observant des alignements de structures, [Pascarella and Argos, 1992] ont déterminé des informations utiles sur les indels. Comme on peut s'en douter, les indels ne sont pas situés au hasard, mais dans des structures de faible contrainte stérique (par exemple les boucles).

Ces informations sont à prendre en compte pour déterminer le score à affecter à un gap. Le problème des pénalités est de répondre à ces deux questions : "Où apparaissent les brèches" et "La brèche est-elle assez longue ?". [Pascarella and Argos, 1992] ont déterminé une méthode permettant de déterminer la probabilité d'avoir un gap après un résidu donné. ClustalW [Thompson et al., 1994] (détaillé ultérieurement) implémente ce système de score : le coût d'ouverture d'un gap dépend du résidu après lequel il apparaît. De plus, l'ouverture d'un gap va être favorisée dans les régions hydrophiles, correspondant aux boucles, en y assignant une pénalité d'ouverture de gap faible. Ces méthodes ont le défaut d'être empirique et de ne pas dépendre des séquences à aligner.

Il existe cependant différents schémas de modélisation : par exemple, un modèle à coût constant (ainsi le coût de l'algorithme est linéaire) ou encore un modèle, plus réaliste, à coût exponentiel avec la longueur de l'insertion. Cependant, les algorithmes d'alignements par programmation dynamique ne peuvent utiliser ce dernier modèle.

[Vogt et al., 1995] ont clairement établi que la performance d'une matrice de substitution est étroitement liée aux choix des pénalités de gap. Il faut donc optimiser au mieux les pénalités pour améliorer significativement les résultats obtenus. Les pénalités de gap sont les paramètres critiques des différentes méthodes les utilisant.

Le score d'un alignement de deux séquences est défini comme la somme des scores des évènements mutationnels (identité, substitution, indel) étant survenus entre deux séquences. Dans la partie suivante, les algorithmes permettant de calculer ce score sont présentés.

3.2 Alignement de deux séquences

Les parties précédentes nous ont permis d'établir un score pour un alignement de deux séquences protéiques. Cela implique la génération d'un score pour deux résidus identiques, 2 résidus différents et la présence de gaps. A présent, il ne manque plus qu'un algorithme permettant d'aligner les séquences. L'algorithme de [Needleman and Wunsch, 1970], déterminant

l'alignement global de deux séquences (*i.e.* alignement sur la longueur totale des séquences), et l'algorithme de [Smith and Waterman, 1981], déterminant l'alignement local de deux séquences (*i.e.* détectant les régions de plus forte similarité entre les séquences) sont les méthodes les plus utilisées et sont détaillés ci-dessous.

3.2.1 [Needleman and Wunsch, 1970]

L'algorithme développé par [Needleman and Wunsch, 1970] est un des algorithmes les plus couramment utilisés pour aligner deux séquences. Il garantit de trouver l'alignement global de deux séquence de score maximum. Le résultat est donc optimal, cependant tous les alignements de même score possibles ne sont pas énumérés. Une comparaison exhaustive serait beaucoup trop coûteuse. Cet algorithme ne dit pas s'il existe un autre alignement ayant le même score.

Cet algorithme est composé de 3 étapes :

Initialisation de la matrice La première étape consiste à placer les deux séquences comme marges d'une matrice M : la première séquence A, de longueur m, est placée horizontalement sur l'axe x de façon à ce que ses acides aminés correspondent aux colonnes ; la seconde séquence B, de longueur n, est placée verticalement sur l'axe y, ses acides aminés correspondant aux lignes.

Transformation L'alignement des deux séquences est décrit comme un chemin à travers la matrice. Trois types de mouvements sont autorisés : diagonal, horizontal et vertical. La matrice est parcourue ligne par ligne, colonne par colonne à partir du coin supérieur gauche jusqu'au coin inférieur droit. Un appariement est représenté en diagonal. Ainsi, l'alignement de deux séquences identiques est simplement représenté par une diagonale du coin supérieur gauche au coin inférieur droit. Les mésappariements sont également représentés par une ligne diagonale. Cependant, le score affecté sera ajusté en tenant compte du système de score. Les brèches sont représentées dans la matrice en utilisant des chemins horizontaux ou verticaux. Une insertion sur la première séquence est représentée par une ligne horizontale alors qu'une insertion sur l'autre séquence sera représentée par une ligne verticale. Les brèches peuvent être internes ou terminales et de n'importe quelle longueur. Tout comme pour les mésappariements, le score sera ajusté à l'aide des pénalités choisies pour l'ouverture et l'extension d'une brèche.

Nous allons à présent formaliser cette étape sous la forme d'un problème de programmation dynamique. L'objectif est de déterminer le chemin conduisant à la plus grande valeur du score d'alignement. Ce score, correspondant au chemin à déterminer, est $M_{m,n}$. En effet, si le chemin optimal conduisant au score optimal se termine en $M_{m,n}$, il passe donc nécessairement par l'une des trois cases adjacentes de la matrice : $M_{m-1,n-1}$ par une mutation ou une identité,

$M_{m-1,n}$ par une délétion $M_{m,n-1}$ par une insertion. La fonction de pénalisation des gaps est ici une fonction constante du type : $g(k) = \psi$ avec $\psi \in \mathbb{R}$. Pour maximiser le score, le chemin optimal passe obligatoirement par la case correspondant au maximum de ces trois valeurs. La généralisation de cette observation permet d'écrire le problème de recherche d'alignement global sous la forme d'un problème de programmation dynamique dont l'équation de récurence est la suivante :

$$M_{i,j} = \max \begin{pmatrix} M_{i-1,j-1} + \delta(A_i, B_j) \\ M_{i-1,j} - \psi \\ M_{i,j-1} - \psi \end{pmatrix} \tag{3.1}$$

Identification de l'alignement Cette étape permet d'obtenir l'alignement final à partir de la matrice. Pour trouver l'alignement optimal, on établit dans la matrice le chemin correspondant au passage par les scores les plus élevés. Pour cela, on part du score maximum en partant de la case en bas à droite et on s'autorise les trois types de mouvements : diagonal, vertical et horizontal.

Les deux dernières étapes sont distinctes. Cette décomposition va être très utile dans le cadre de la recherche de séquences dans les banques (3.2.3). En effet, l'alignement sera calculé uniquement si le score attribué à la diagonale est supérieur à un seuil donné. Le temps de calcul sera donc réduit. Elle est utilisée dans le logiciel LASSAP [Glémet and Codani, 1996], dédié à l'analyse massive de données.

3.2.2 [Smith and Waterman, 1981]

L'algorithme de [Needleman and Wunsch, 1970] crée un alignement global. Cela signifie que la totalité de la première séquence est alignée avec la totalité de la deuxième séquence. Des sous-séquences courtes mais très similaires peuvent être manquées lors de l'alignement global car elles ont peu de poids par rapport au reste de la séquence. L'algorithme de [Smith and Waterman, 1981] détermine un alignement contenant la sous-séquence de score le plus élevé des deux séquences d'origine. L'algorithme de [Needleman and Wunsch, 1970] est contraint et ne possède aucun degré de liberté alors que celui de [Smith and Waterman, 1981] repose sur un certain nombre d'hypothèses, certaines arbitraires.

Les trois changements principaux, par rapport à l'algorithme décrit précédemment, sont :
- les mésappariements sont pénalisés par un score faible ou négatif ;
- 0 est le score minimum pouvant être enregistré dans la matrice ;
- l'origine et la fin du chemin optimal peuvent être trouvés n'importe où dans la matrice.

Le premier point est nécessaire pour revoir le score à la baisse au fur et à mesure que des

mésappariements sont ajoutés. Ainsi, le score augmentera dans les régions de forte similarité et diminuera en-dehors de celles-ci. Si deux segments de forte similarité sont détectés, soit ils seront assez proches pour n'être plus qu'un segment (reliés par une brèche), soit ils seront considérés comme deux segments indépendants de similarité locale. Le second point est nécessaire pour que chaque case de la matrice puisse être un nouveau départ potentiel pour un alignement. Ainsi, chaque court segment de similarité pourra commencer avec un score de zéro. Enfin, le dernier point indique que les régions de forte similarité locale sont recherchées dans la matrice entière.

Comme dans le cas de Needleman & Wunsch, cet algorithme se formalise sous la forme d'un problème de programmation dynamique, nous allons étudier les modifications apportées à l'équation (3.1).

La première modification majeure est de prendre en considération, non plus les alignements de séquences dans leur globalité, mais ceux de sous-fragments. Ceci nous conduit à modifier l'équation de récurrence en ajoutant la comparaison avec l'élément nul dans la partie droite de l'équation :

$$M_{i,j} = \max \begin{pmatrix} M_{i-1,j-1} + \delta(A_i, B_j) \\ M_{i-1,j} - \psi \\ M_{i,j-1} - \psi \\ 0 \end{pmatrix} \tag{3.2}$$

La seconde modification apportée par Smith & Waterman est de considérer une fonction de coût des insertions linéaires. Celle-ci a pour but de pénaliser moins fortement les gaps de grande longueur. L'introduction de cette fonction modifie l'équation de récurrence qui devient alors :

$$M_{i,j} = \max \begin{pmatrix} M_{i-1,j-1} + \delta(A_i, B_j) \\ \max\limits_{1 \leq k \leq i} (M_{i-k,j} - g(k)) \\ \max\limits_{1 \leq l \leq j} (M_{i,j-l} - g(l)) \\ 0 \end{pmatrix} \tag{3.3}$$

où $g(x)$ est la fonction de coût d'insertion d'un gap de longueur x défini par :

$$g(x) = gapo + gape \times (x - 1)$$

où *gapo* et *gape* correspondent respectivement au coût de création d'un gap (généralement dénommé gap-*open*) et extension d'un gap (généralement dénommé gap-*extend*).

3.2.3 Recherche de séquences dans les banques

L'algorithme de [Smith and Waterman, 1981], qui garantit de trouver le(s) alignement(s) optimal (optimaux) entre deux séquences, a cependant le désavantage d'être de forte complexité. Pour le problème de l'alignement de deux séquences, le temps d'exécution et l'espace mémoire ne sont pas des contraintes. Cependant, pour la recherche de similarité entre une séquence " requête " et une banque de séquences, un tel algorithme est inadapté. Des heuristiques rapides ont donc été développées pour répondre à ce besoin.

Les deux algorithmes les plus populaires ayant été développés pour fournir une alternative rapide sont FASTA [Lipman and Pearson, 1985] et BLAST (Basic Local Alignment Search Tool) [Altschul et al., 1990]. Chacun de ces algorithmes est effectivement plus rapide que celui de [Smith and Waterman, 1981]. En effet, ils restreignent l'espace de solutions en balayant la banque à la recherche de zones d'ancrage (zones de forte similarité entre deux séquences) avant de réaliser des alignements rigoureux. Ces heuristiques ne garantissent pas de trouver les alignements de scores optimaux (au contraire de [Smith and Waterman, 1981]).

FASTA

FASTA est le premier programme, largement utilisé, de recherche de similarité dans les banques. Il existe différenets variantes de FASTA pour comparer toutes les combinaisons possibles de requêtes à partir d'une séquence protéique ou nucléique contre une banque de données protéique ou nucléique. Ce programme recherche des alignements locaux en balayant les séquences avec de courtes séquences appelées *mots*.

L'algorithme de FASTA est composé de quatre étapes :

- Etape 1 : * Tout d'abord un dictionnaire de mots communs à la séquence requête et à la séquence de la banque est calculé. Tous les mots identiques entre les deux séquences vont être recherchés. La longueur des mots est fixe et est un des paramètres importants de l'algorithme, appelé *ktup*. Il est fixé à deux par défaut pour les séquences protéiques. La liste de mots est utilisée pour détecter les meilleures régions d'identité (aussi appelées diagonales), c'est à dire celles constituées d'un grand nombre de mots. La figure 3.4, étape 1, met en évidence l'ensemble des mots sous la forme de diagonales.
- Etape 2 : Pour les dix meilleures diagonales (*cf.* figure 3.4, les mots sélectionnés sont indiqués en gras), un score est recalculé en utilisant une matrice de substitution pour tenir compte des espaces entre les mots. Cette étape correspond à une recherche de similitudes sans indel.
- Etape 3 : * Puis un alignement rudimentaire est réalisé : les régions précédentes sont jointes, s'il en existe au moins deux et si chacune d'elles possède un score supérieur à

un score seuil. On réunira ces régions initiales à chaque fois que leur score diminué d'une pénalité de jonction est supérieur ou égal à un score donné (score de la meilleure diagonale avant jonction). Cette étape permet d'éliminer les segments peu probables parmi ceux définis à l'étape précédente (*cf.* figure 3.4, étape 3).

– Etape 4 : * Cette étape n'est réalisée que pour les alignements rudimentaires de haut score ; ainsi l'alignement avec l'algorithme de [Needleman and Wunsch, 1970] n'est calculé que pour la séquence requête et les " meilleures " séquences de la banque. L'alignement obtenu ainsi que la zone utilisée pour l'algorithme de programmation dynamique, sont indiqués

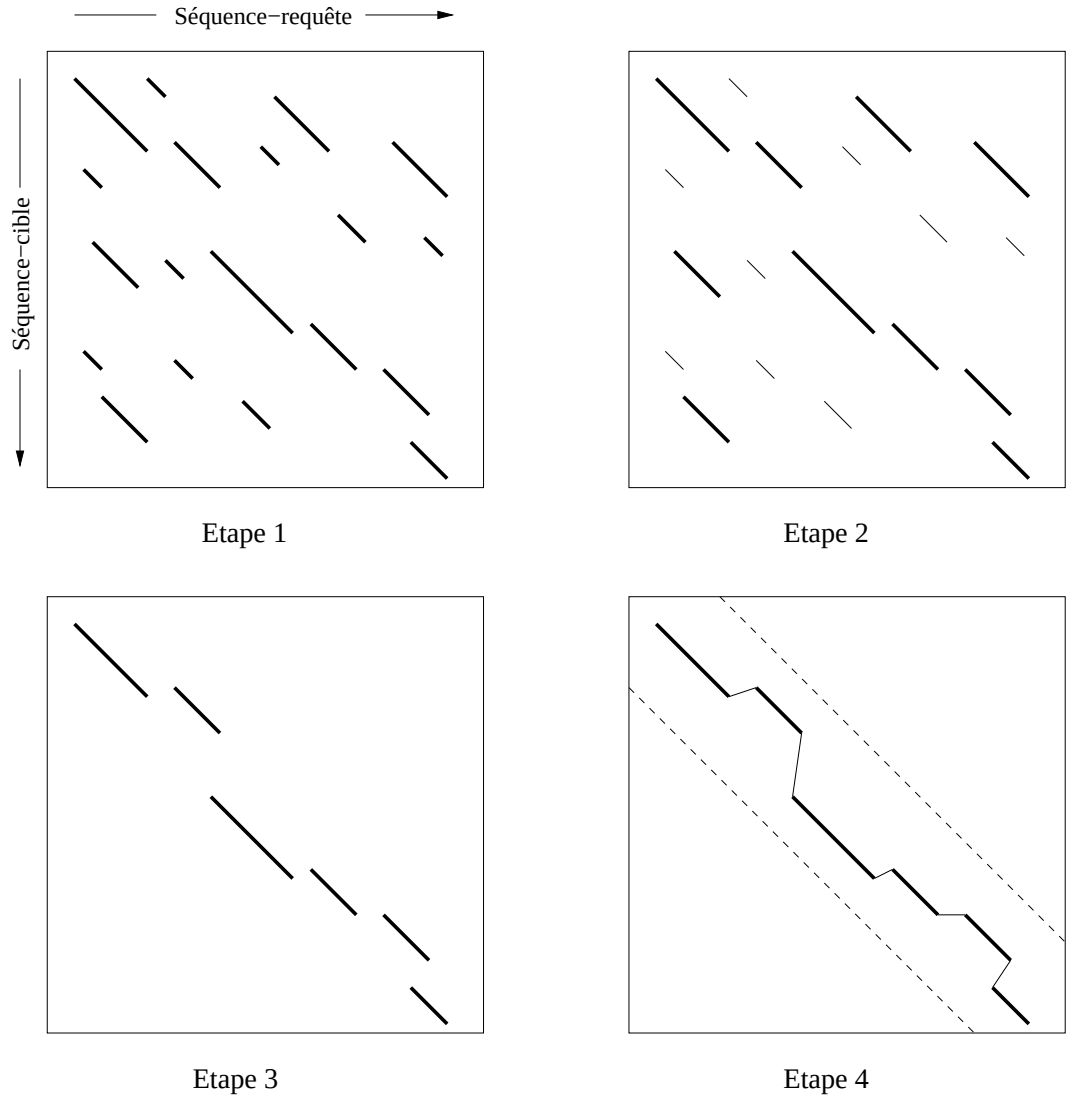


Fig. 3.4: Les quatre phases de l'algorithme *Fasta* – (Etape 1) Dictionnaire de mots communs représentés par les diagonales. (Etape 2) Les 10 meilleures diagonales sont conservées. (Etapes 3 et 4) Alignement des diagonales sélectionnées en fonction d'un score-seuil.

sur la figure 3.4, étape 4.
Les étapes commençant par une * sont les points critiques ; surtout la première étape. Par exemple dans le cas de deux séquences possédant une lettre identique sur deux, on ne détecte rien.

BLAST

BLAST est un programme de recherche de similarité développé au NCBI. Tout commme pour FASTA, il existe différents types de BLAST pour comparer toutes les combinaisons possibles de requêtes à partir d'une séquence protéique ou nucléique contre une banque de données protéique ou nucléique. Une recherche BLAST est l'une des façons les plus classiques d'obtenir des informations à propos d'un gène ou d'une protéine : la recherche révèle quelles séquences liées sont présentes dans le même organisme ou au sein d'autres organismes. L'intérêt de l'algorithme est que celui-ci est basé sur un modèle statistique. L'unité fondamentale de BLAST est le HSP (High-scoring Segment Pair), correspondant à une région de similitude la plus longue possible entre deux séquences ayant un score supérieur ou égal à un score seuil.

L'algorithme de BLAST est composé de trois étapes :

- Etape 1 : Un dictionnaire de mots synonymes (et non pas seulement identiques) composant la séquence “requête” est calculé. Un automate déterministe est créé. Cette étape est plus longue que celle de Fasta du fait de la construction d'une liste de mots similaires et non identiques. Cette ” perte ” de temps est relative car elle permet de gagner en sensibilité. Pour chaque paire de segments, une probabilité est calculable.
- Etape 2 : L'algorithme recherche alors toutes les occurences de mots de la séquence-requête pour chaque séquence-cible de la banque. Cette étape, réalisée à partir de l'automate précédemment construit, est très rapide.
- Etape 3 : Pour chaque occurence de ces mots de longueur fixe, on cherche à étendre l'alignement de part et d'autre. Les segments sont étendus à partir des mots (dans les deux directions, le long de la séquence) pour obtenir des HSP. Quand le score diminue, l'extension est arrêtée. Cette étape de l'algorithme est très longue.

Pour chaque alignement, Blast fournit une espérance mathématique (E-value), correspondant au nombre de séquences que l'on s'attendrait à trouver par hasard dans une banque de séquences de même taille. Plus la valeur de E est faible et plus l'alignement est significatif. Il faut faire attention à la zone limite de ce score : on peut préfèrer avoir des faux-positifs (semblables au niveau score mais pas au niveau biologique) pour éviter de rater des faux-négatifs (différents en terme de score mais semblable au niveau biologique).

3.3 Alignement multiple de séquences

L'alignement multiple de séquences est l'une des méthodes les plus utilisées en bioinformatique. Celle-ci est à la base de nombreuses analyses de séquences et couvre un vaste champ d'applications : annotation de séquences, prédiction fonctionnelle et structurale (structure secondaire et tertiaire), caractérisation de famille de protéines, étude phylogénétique... Dans le cas d'études phylogénétiques, [Phillips and Wheeler, 2000] ont publié une revue mesurant l'impact de la qualité de l'alignement et [Morrison and Ellis, 1997] ont mis en avant l'importance de l'alignement dans une étude de cas. L'importance d'un bon alignement a été également soulignée pour la détermination de la structure secondaire des protéines par [Jones, 1999]. C'est pourquoi l'amélioration, ne serait ce qu'infime, des méthodes automatiques permettant de générer un alignement multiple est primordiale. Comme on peut aisément le comprendre, plus l'alignement de séquences est de bonne qualité et meilleure sera l'analyse qui en découlera. Le calcul de l'alignement multiple est donc un thème de recherche particulièrement important. De plus, le nombre croissant d'articles récents à ce sujet montre que le problème est loin d'être résolu.

Bien que dans les cas classiques, de bons résultats soient généralement obtenus, certains cas sont encore problématiques pour les méthodes d'alignement : les grandes insertions/délétions, une séquence très divergente alignée avec une famille de protéines, les protéines multi-domaines, des séquences de longueurs différentes... De plus, la précision des méthodes automatiques chute de façon importante quand la similarité des séquences se situe dans la zone d'ombre (moins de 30% d'identité). Cependant, [Brenner et al., 1998] ont montré que des protéines présentant de faible taux de similarité sont souvent liées et partagent une structure commune. Mais cette similarité est difficile à détecter par une comparaison directe entre deux séquences. Un bon alignement multiple est donc nécessaire.

D'un point de vue purement informatique, l'alignement multiple est un problème complexe. [Batzoglou, 2005] date le début de l'alignement de séquences à 1966 avec la définition de distance entre deux chaînes de caractères. Bien qu'un algorithme de programmation dynamique existe [Lipman et al., 1989], garantissant un alignement mathématiquement optimal, la méthode est limitée en nombre et en taille de séquences. La généralisation de l'algorithme optimal de deux séquences (présenté ci-dessus) est impossible en pratique pour un grand nombre de séquences. Sa complexité en temps est proportionnelle à L^N, où N représente le nombre de séquences de longueur L. Comme dans la plupart des cas NP-complets, de nombreuses approches heuristiques ont été développées pour répondre au besoin d'aligner un grand nombre de séquences. En général, ces approches vont " réduire " la taille du problème pour revenir au cas que l'on sait résoudre : l'alignement de 2 séquences. Traditionnellement, l'approche la plus populaire est la méthode progressive [Feng and Doolittle, 1987]. Un alignement multiple est construit graduellement en alignant d'abord les séquences les plus proches et en ajoutant successivement les plus distantes. Comme pour les alignements de deux séquences, on peut également opposer

les stratégies globales, qui vont aligner les séquences sur la totalité de leurs longueurs, et locales dont le but est de retrouver les segments les plus conservés.

Dans cette section, nous allons décrire brièvement les différentes méthodes existantes, les critères et bases de référence permettant de les comparer. Enfin, nous expliquerons pourquoi nous avons choisi de travailler sur l'alignement multiple progressif de séquences dont nous détaillerons les différentes étapes.

3.3.1 Les différentes approches

Pour réaliser un alignement multiple d'un grand nombre de séquences, de nombreux algorithmes ont été développés. Par exemple, les revues de [Gotoh, 1999, Notredame, 2002, Batzoglou, 2005, Chakrabarti et al., 2006] permettent d'avoir un point de vue global sur la question. Il est possible d'opposer ces différents algorithmes par rapport à deux critères : la nature de l'alignement final (global ou local) et l'approche algorithmique (itérative ou progressive).

Global versus local

Comme pour l'alignement de paire de séquences, on peut opposer les approches globales qui tentent à aligner les séquences sur la totalité de leur longueur aux approches locales qui vont rechercher les régions de plus forte similarité.

[McClure et al., 1994, Briffeuil et al., 1998, Thompson et al., 1999b, Karplus and Hu, 2001, Lassmann and Sonnhammer, 2002] ont réalisé des études comparatives. Ces études ont un résultat commun : l'approche la plus efficace dépend de la nature des séquences. Les algorithmes globaux produisent de meilleurs alignements lorsque l'on aligne :

- des séquences équidistantes en terme de similarité ;
- des familles de séquences peu ou moyennement divergentes ;
- une séquence orpheline avec une famille de séquences.

Au contraire, les algorithmes locaux obtiennent les meilleurs résultats en présence d'importantes insertions N/C-terminales ou internes, dans le cas des protéines multi-domaines ou lorque la similarité entre les séquences est faible. En somme, l'approche locale est bien la plus adaptée lorsque ce sont des régions de similarité qui sont communes aux différentes séquences.

Fig. 3.5: *Alignement de 4 tRNA-synthétases réalisé avec ClustalW.* Les motifs caractéristiques sont encadrés. Remarque : l'alignement est au format clustal

Pour illustrer ce propos, l'exemple des tRNA-synthétases de *E.*coli, présente le cas de longues insertions dans une séquence. Les séquences d'isoleucyl- et leucyl- tRNA-synthétases sont proches, cependant la première présente une longue insertion. Pour réaliser un alignement multiple, deux séquences de tRNA-synthétases supplémentaires (également de type I) ont été ajoutées. ClustalW [Thompson et al., 1994] et DiAlign [Morgenstern, 1999], réciproquement

programmes de références de l'alignement global et local, ont été testés dans leur capacité à retrouver les motifs caractéristiques des tRNA-synthétases (HIGH, DWCISR et KMSKS). Les figures 3.5 et 3.6 représentent les alignements obtenus à partir de ces outils. Dans les deux cas, le premier motif, situé avant la longue insertion, sur les séquences est correctement aligné. Les deux motifs suivants sont situés après l'insertion. On observe que seul DiAlign parvient à les aligner correctement. Parmi les paramètres de ClustalW, le coût d'insertion d'un gap est souvent trop important et le programme a tendance à mésapparier au dépend des insertions. On ne retrouve donc pas l'alignement correct des motifs caractéristiques.

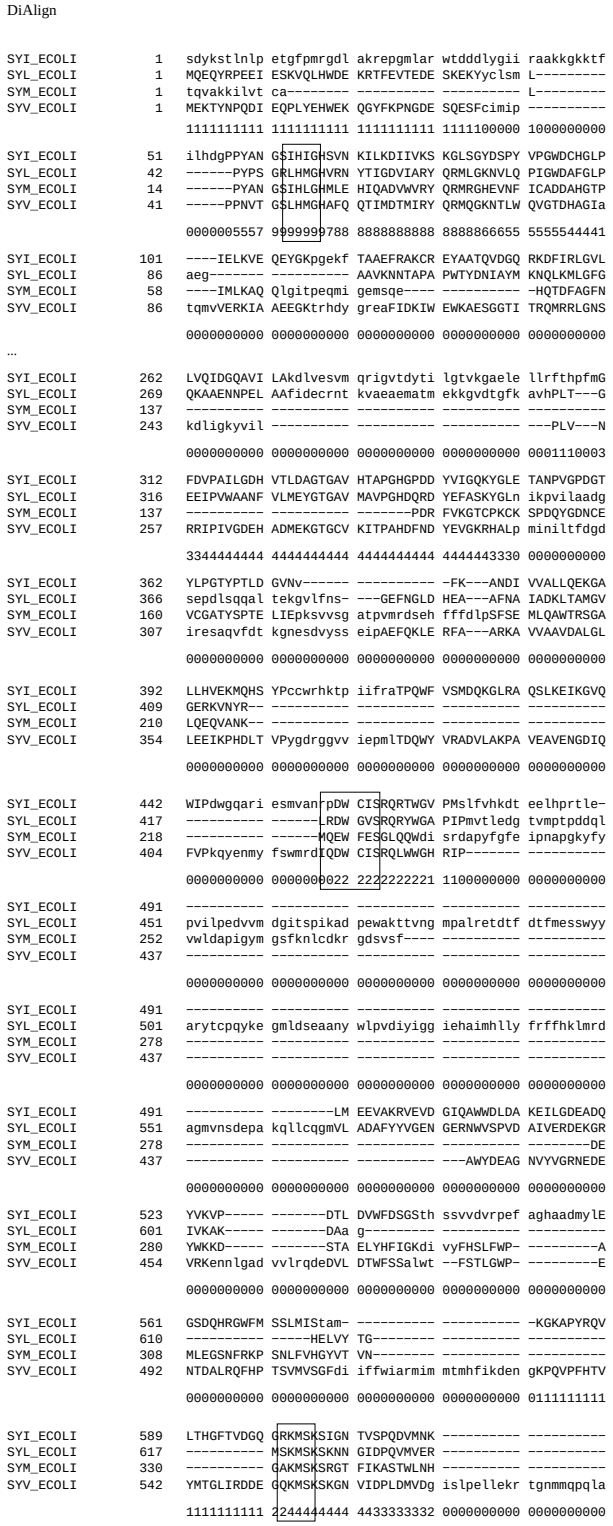


Fig. 3.6: *Alignement de 4 tRNA-synthétases réalisé avec DiAlign.* Les motifs caractéristiques sont encadrés. Remarque : l'alignement est au format clustal

Certaines méthodes [Notredame, 2002, Sammeth and Morgenstern, 2003] combinent ces deux approches et bénéficient alors des avantages des deux. Ce gain de précision a cependant un coût important en temps de calcul.

Progressif versus itératif

En plus de la nature de l'alignement final (global ou local), les différentes méthodes peuvent être opposées d'un point de vue algorithmique : les deux approches (itérative et progressive) sont décrites ci-dessous.

Approche itérative L'approche itérative permet de se ramener dans le cadre de l'alignement de deux séquences. L'alignement de paires de séquences est utilisé itérativement pour ajouter une nouvelle séquence à un alignement multiple donné, représenté par un profil.

L'approche itérative est une méthodologie en trois étapes. Une fois la paire de séquences les plus proches, déterminée et alignée, tant que toutes les séquences ne sont pas alignées, il faut :
- Calculer les distances entre les séquences et celles déja alignées ;
- Choisir parmi les séquences qui n'ont pas encore été alignées, la plus proche de celles déjà alignées ;
- Aligner celle-ci avec le profil de l'alignement multiple précédent. Les nouveaux gaps sont insérés dans toutes les séquences du groupe.

Approche progressive Tout comme l'approche itérative, l'approche progressive est une heuristique permettant de résoudre le problème de l'alignement multiple.

L'approche progressive est une méthodologie en trois étapes :
- Les distances entre les paires de séquences sont calculées et conservées dans une matrice ;
- A partir de cette matrice, un arbre (ou structure) de guidage est calculé ;
- Les séquences sont alignées progressivement en suivant l'ordre des branches.

Cette approche sera plus détaillée dans la section 3.4.

Panorama des programmes existants La figure 3.7 page suivante permet d'observer les relations entre les différents algorithmes et programmes.

DiAlign [Morgenstern et al., 1996, Morgenstern, 1999, Subramanian et al., 2005] est le programme de référence de l'alignement multiple local. Il est basé sur l'approche segments-segments. Au lieu de comparer des paires de résidus, l'approche basée sur les segments compare des sous-séquences entières des séquences à aligner. La construction des segments, pour les alignements

par paire ou multiple, est réalisée en alignant, localement et sans création de brèches possible, une paire de séquences.

Les méthodes Multalign [Barton and Sternberg, 1987], Multal [Taylor, 1988], Multalin [Corpet, 1988] et PileUp réalisent un alignement multiple progressif global et diffèrent de par leur façon de déterminer l'ordre des séquences à aligner.

PIMA [Smith and Smith, 1992] utilise un algorithme local qui aligne uniquement les motifs les plus conservés. Cette méthode propose deux techniques d'alignements par défaut (lien maximum ou sequential branching) afin de déterminer l'ordre des séquences à aligner.

L'approche par chaîne de Markov cachées a été utilisée à plusieurs reprises pour représenter les séquences à aligner, aussi bien pour l'approche itérative [Holmes and Bruno, 2001] que pour l'approche progressive [Loytynoja and Milinkovitch, 2003]. [Schwartz and Pachter, 2006]

Les méthodes progressives ClustalW (programme de référence de l'alignment multiple global), MAFFT, Muscle, Probcons et POA sont présentées dans la section 3.4.

Les dernières approches proposées (hors les méthodes progressives décrites par la suite) propose l'utilisation et l'amélioration d'un premier alignement incomplet ou imparfait. [Schwartz and Pachter, 2006 proposent une approche de "sequence annealing", consistant à déterminer les appariements

Fig. 3.7: Les principaux programmes d'alignement multiple – Les différentes approches sont opposées : Local *vs* global, progressif *vs* itératif.

simples consistents dans un alignement multiple partiel. Qoma [Zhang and Kahveci, 2007] propose, comme POA, une représentation de l'alignement de séquences sous forme d'un graphe et traite localement (à l'aide d'une fenêtre de n acides aminés) un alignement "brouillon" afin d'obtenir un alignement multiple quasi-optimal.

3.3.2 Fiabilité d'un alignement

Il est important de mesurer la qualité d'un alignement et ainsi, les améliorations apportées par les nouvelles méthodes. Pour cela, des bases d'alignement de référence existent. Elles ont été réalisées à partir d'alignements structuraux et tentent de représenter au mieux les cas critiques pour les méthodes d'alignement.

Les scores

Pour mesurer la précision d'un alignement test par rapport à un alignement de référence, différents scores ont été élaborés. Ces scores sont différents et ne mesurent pas les mêmes informations. Ils sont donc utilisés conjointement lors des études qualitatives.

Sensibilité et spécificité [Carillo and Lipman, 1988] Dans le cas d'un alignement multiple, où toutes les séquences sont comparées entre elles, il est naturel de considérer que le score de cet alignement est la somme de tous les alignements deux à deux qu'il induit.

La sensibilité (également appelée sum-of-pairs ou $F_{Developer}$ [Sauder et al., 2000]) calcule la somme des résidus correctement alignés dans chaque paire de séquences par rapport à la longueur de l'alignement de référence

$$\text{Sensibilité} = \frac{\sum \text{Nombre de résidus correctement alignés}}{\text{Longueur de l'alignement de référence}}$$

Plus la sensibilité est grande et plus l'alignement test couvre l'alignement de référence.

La spécificité (également appelée reverse sum-of-pairs ou $F_{Modeler}$) se calcule de la même façon que la sensibilité mais cette fois-ci par rapport à la longueur de l'alignement test.

$$\text{Spécificité} = \frac{\sum \text{Nombre de résidus correctement alignés}}{\text{Longueur de l'alignement test}}$$

Plus la spécificité est grande et moins l'alignement test contient de résidus mal alignés.

La sensibilité ne pénalise pas le sur-alignement (i.e. alignement de paires de résidus non alignables au point de vue structure) ; la spécificité ne pénalise pas le sous-alignement.

Ces mesures, bien que les plus utilisées, présentent de nombreux désavantages. Un bon score total ne signifie pas un bon critère de qualité pour un alignement multiple car ce score est la somme des scores d'alignements par paire. De plus, ces méthodes ne donnent aucun crédit aux régions de l'alignement test qui sont uniquement décalées d'une ou quelques positions et ne donnent pas d'importance aux différents évènements évolutifs. Des scores plus complexes ont été mis au point ces dernières années. Ils permettent notamment de repérer les zones mal alignées et ainsi de corriger un alignement multiple.

Cline score [Cline et al., 2002] Contrairement aux deux scores présentés ci-dessus, ce score, appelé CS, prend en compte les décalages pouvant intervenir entre deux alignements. Les auteurs ont souhaité développer une méthode identifiant les positions suspectes d'un alignement et permettant alors de les retirer.

Circular Sum [Gonnet et al., 2000] Les auteurs ont défini une nouvelle fonction de score basée sur un modèle markovien d'évolution. Ils souhaitent prendre en compte la structure associée à un arbre phylogénétique issu d'un alignement et ainsi donner un poids propre aux différents évènements évolutifs. Pour cela, ils proposent de réaliser le cycle le plus court en parcourant uniquement les séquences (feuilles de l'arbre). Cependant, il n'est pas nécessaire de calculer l'arbre : les séquences, munies de leurs scores d'alignements par paire, vont être parcourues en utilisant un algorithme heuristique implémentant le problème du voyageur de commerce.

Score statistique basé sur l'analyse de profils [Ahola et al., 2006] Les auteurs proposent une nouvelle fonction de score reposant sur un score statistique. Ce score est basé sur le comptage de positions significativement conservées dans l'alignement en conjonction avec l'analyse statistique du profil. En plus de l'étude d'un cas donné et de la mesure de qualité des algorithmes les plus récents, ils ont montré que ce score apporte d'aussi bons résultats que la sensibilité décrite ci-dessus.

Les bases d'alignements de références

Balibase [Thompson et al., 1999a] Balibase est une base d'alignements de référence créée pour l'évaluation des programmes d'alignements multiples. La première version a été publiée en 1999 et actuellement, la version 3 est disponible en ligne. Les auteurs ont distingué cinq catégories critiques représentées par les cinq ensembles de références suivants :

– des séquences équi-distantes avec différents niveaux de similarité
– des familles de séquences alignées avec une séquence orpheline très divergente
– des sous-groupes de séquences avec moins de 25% d'identité entre les groupes
– des séquences avec des extensions N et C-terminales
– des séquences comprenant des insertions internes.

La seconde version comprend trois références supplémentaires :
– des séquences possédant des domaines trans-membranaires
– des séquences comprenant des permutations de domaines
– des séquences présentant des répétitions de domaines.

La troisième version reprend les références de la première version, tout en y ajoutant les alignements de séquences tronquées. Seules les régions homologues ont été conservées.

Cependant les deux dernières versions n'ont pas encore été utilisées pour des études comparatives.

Sequence Alignment Benchmark : SABmark Cette base d'alignements de référence a pour but d'étudier le comportement des méthodes d'alignement face à de faibles taux de similarité entre les séquences. Elle a été réalisée à partir de données de structure. Elle est composée de deux sous-ensembles :
– la zone d'ombre, comprenant des séquences présentant moins de 25% de similarité
– des séquences ayant moins de 50% de similarité regroupées sous le nom de super-familles.

Chaque alignement est composé de 25 séquences au plus pour limiter l'impact des structures 3D les plus abondantes. Les auteurs ont créé deux autres jeux des données : ils ont repris les deux précédents auxquels ils ont ajouté autant de faux-positifs (séquences appartenant à un repliement différent) que de séquences de l'alignement de départ.

Il existe un grand nombre de bases d'alignements de référence : Préfab [**?**], OXBench [Raghava et al., 2003], PALI [Balaji et al., 2001]... Toutes ces bases de benchmark représentent une importante source d'informations. Cependant il faut garder à l'esprit que nombreuses d'entre elles sont liées à une méthode de génération d'alignements multiples. Par exemple, certains auteurs [Subramanian et al., 2005] ont montré que Balibase était biaisée en faveur de ClustalW.

3.4 Alignement multiple progressif

Cette partie est consacrée à l'approche progressive décrite par [Feng and Doolittle, 1987] et [Taylor, 1988]. La généralisation des algorithmes de programmation dynamique permettant d'aligner une paire de séquences est difficilement applicable dans la pratique pour l'alignement multiple de séquences. Cependant, il est possible de représenter un alignement par un profil

(*i.e.*un alignement (par paire ou multiple) peut être considéré comme une séquence généralisée), et d'aligner deux profils avec un algorithme d'alignement par paire.

La mise en place de cette méthodologie a été possible car chacune des différentes étapes la constituant avait été développée indépendamment pour répondre à une problématique précise [Doolittle, 2000] : l'alignement d'une paire de séquences, la construction d'arbres phylogénétiques, l'alignement d'une séquence à un profil et de deux profils. Cette méthode possède l'avantage d'être sensible au point de vue biologique mais aussi d'être facile à calculer du point de vue informatique.

Cette stratégie repose sur un algorithme en trois étapes :
- Un alignement par paire de toutes les séquences est calculé. Les scores attribués à ces alignements permettent de représenter les distances entre séquences. L'alignement peut être grossier ou optimal.
- Les distances entre séquences, stockées dans une matrice, permettent alors de construire un arbre de guidage. Il existe de nombreuses méthodes utilisant une matrice de distances pour aboutir à un arbre.
- Enfin, les séquences sont alignées progressivement en suivant l'ordre donné par les branches de l'arbre.

Cette approche présente de nombreux avantages, notamment une meilleure précision et une rapidité de calcul. Cependant, un problème spécifique des méthodes progressives est de tomber dans un minimum local : une erreur ayant lieu au départ de l'alignement progressif sera répercutée dans les étapes suivantes. Ce dysfonctionnement a été énoncé ainsi : " Once a gap, ever a gap " par [Gotoh, 1999]. De plus, il a été montré que l'ordre d'alignement des séquences (déterminé par la structure de guidage) était un paramètre important pour la qualité de l'alignement final. Toutes les étapes de l'approche progressive sont donc importantes.

3.4.1 ClustalW [Thompson et al., 1994]

Clustalw est, *de facto*, le programme de référence pour calculer un alignement multiple progressif global. Bien qu'étant le standard, Clustalw présente un système de score compliqué. Les auteurs se sont inspirés des observations de [Pascarella and Argos, 1992] pour fixer les pénalités d'insertion et de délétion. Celles-ci dépendent de nombreux paramètres tels que la matrice de substitution, la similarité des séquences, la longueur des séquences, la différence de longueurs des séquences, les positions des brèches, la présence de brèches, la proximité d'une brèche, la présence de régions hydrophiles... Le choix de ces paramètres empiriques est le point critique de ce programme (voir 3.1 page 118 qui regroupe différents paramètres). Le coût d'insertion d'un gap est souvent trop important et le programme a tendance à mésapparier au dépens des insertions.

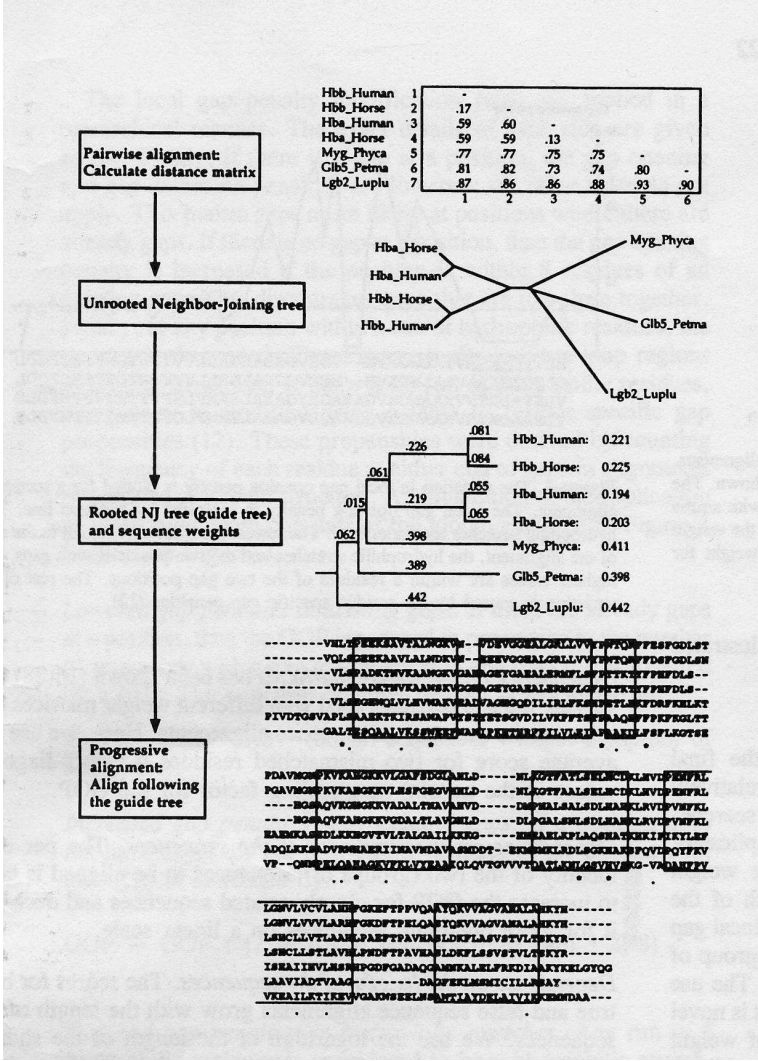


Fig. 3.8: Les différentes étapes de ClustalW. Figure tirée de [Thompson et al., 1994] – Alignement par paire de toutes les séquences. Construction d'un arbre de guidage. Alignement progressif des séquences en parcourant les branches de l'arbre.

Depuis [Feng and Doolittle, 1987] et [Thompson et al., 1994], aucune amélioration n'avait été proposé au niveau du temps de calcul ; hormis [Li, 2003] qui a proposé une parallélisation de ClustalW. Depuis les années 2000, de nouvelles méthodes ont vu le jour et ont apporté de nombreuses solutions : rapidité, système de score plus simple... Dans la section suivante, nous décrirons brièvement ces méthodes en mettant en avant leurs particularités.

Etape	option	paramètre
Alignement par paire	-PWMATRIX -PWDNAMATRIX	Choix de la matrice de substitution (protéines ou ADN)
	-PWGAPOPEN -PWGAPEXT	Choix des pénalités d'ouverture et d'extension de gaps
Alignement multiple	-PWMATRIX -PWDNAMATRIX	Choix de la matrice de substitution (protéines ou ADN)
	-PWGAPOPEN -PWGAPEXT	Choix des pénalités d'ouverture et d'extension de gaps
	-MAXDIV	pourcentage d'identité entre les séquences

Tab. 3.1: Tableau regroupant différents paramètres externes du programme de référence ClustalW

3.4.2 Nouvelles méthodes

Toutes ces nouvelles méthodes ont en commun de reposer sur un algorithme progressif mais elles diffèrent de par leur idée de base : utilisation des propriétés physico-chimiques des acides aminés, reconnaissance de motifs, modèle probabiliste ou encore théorie des graphes...

MAFFT

MAFFT [Katoh et al., 2002, Katoh et al., 2005] présente un système de score simplifié qui permet d'augmenter la précision des alignements et de diminuer le temps CPU ; ceci même pour des séquences contenant de grandes insertions ou extensions aussi bien que pour des séquences distantes. Les régions homologues sont rapidement détectées par la transformée de Fourier rapide (FFT) pour laquelle une séquence d'acides aminés est convertie en une séquence composée de vecteurs (polarité/volume) représentant les acides aminés. La corrélation entre 2 séquences est la somme des corrélations des volumes et des polarités.

Dans MAFFT, un alignement initial est construit grâce à une simple (ou double) approche progressive, puis affiné au cours d'une étape de consolidation. Premièrement, une distance grossière est calculée entre chaque paire de séquences en fonction du nombre de 6-uplets partagés par les deux séquences. Un arbre de guidage est alors construit avec la méthode de l'UPGMA avec un critère d'agrégation modifié. Les séquences sont alors progressivement alignées en suivant les branches de l'arbre, grâce à la FFT. Cette méthodologie a été nommée FFT-NS-1 par

les auteurs.

Cette procédure peut être suivie par un deuxième alignement progressif (FFT-NS-2). La matrice de distance initiale calculée à partir des 6-uplets est moins fiable que celle basée sur des alignements par paire. Une matrice de distance plus fiable est obtenue à partir du premier alignement progressif, en calculant la distance de Kimura (propres aux séquences protéiques) pour chaque paire de séquences. Pour les protéines, cette distance, notée d, se calcule ainsi $d = -ln(1 - p - 0,2p^2)$ où p est la fraction de différences observées. Les deux étapes suivantes de l'algorithme sont inchangées.

L'étape de consolidation (décrite dans la section 3.4.3 page 124) aboutit à une méthodologie appelée FFT-NS-i. MAFFT offre aussi 3 autres possibilités : NW-NS-[1,2,i] en utilisant l'algorithme de programmation dynamique de [Needleman and Wunsch, 1970] à la place de la FFT.

MUSCLE [Edgar and Sjölander, 2003, Edgar, 2004b, Edgar, 2004a]

Les caractéristiques principales de MUSCLE sont très similaires à l'approche NW-NS-[1,2,i], développée dans MAFFT et expliquée ci-dessus. Cette méthode offre un système de score simplifié permettant d'augmenter la qualité des alignements multiples tout en réduisant le temps CPU.

Cet algorithme est composé de plusieurs étapes représentées sur la figure 3.9.

- La première étape permet d'obtenir un alignement multiple "grossier", en mettant l'accent sur la vitesse plutôt que la précision. Une distance grossière est calculée entre chaque paire de séquences en fonction du nombre de 6-uplets partagés par les deux séquences. Un arbre de guidage est alors construit avec la méthode de l'UPGMA avec un critère d'agrégation modifié. Les séquences sont alors progressivement alignées en suivant les branches de l'arbre.
- La principale source d'imprécision liée à la première étape est la matrice de distance initiale calculée à partir des 6-uplets. En effet, elle est moins fiable que celle basée sur des alignements par paire. Une matrice de distance plus fiable est obtenue à partir du premier alignement progressif, en calculant la distance de Kimura pour chaque paire de séquences. Les deux phases suivantes (calcul de l'arbre et alignement des séquences) de l'algorithme sont inchangées.
- L'étape de consolidation est différente de celle décrite dans la section 3.4.3. L'arbre de guidage est dissocié en deux sous-arbres ; les deux alignements correspondant sont calculés. Ces deux alignements sont alignés et l'on conserve le nouvel alignement en cas d'amélioration du score obtenu. Le nombre d'itération peut être défini ou non.

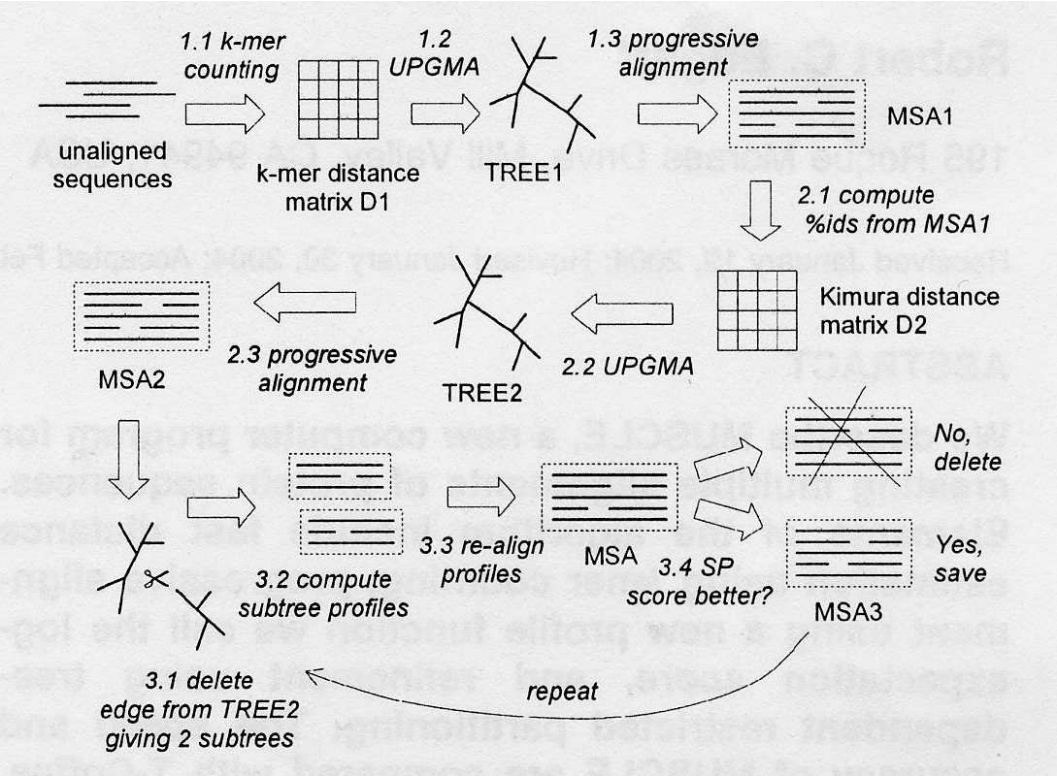


Fig. 3.9: Les différentes étapes de Muscle. Figure issue de [Edgar, 2004b] – On observe les étapes classiques (répétées deux fois) d'un alignement progressif : Alignement par paire de toutes les séquences. Construction d'un arbre de guidage. Alignement progressif des séquences en parcourant les branches de l'arbre. La dernière ligne représente l'étape de consolidation.

PROBCONS [Do et al., 2005]

PROBCONS présente une première étape très importante et très riche. Pour chaque paire de séquences : une matrice contenant les probabilités d'alignement entre chaque lettre x_i d'une séquence et y_i de l'autre est calculée. Cette étape est basée sur un modèle de Markov caché pour spécifier la distribution des probabilités de tous les alignements de 2 séquences. Les probabilités d'émission, qui correspondent au traditionnel score de substitution, sont basées sur la matrice BLOSUM 62. Les probabilités de transition, correspondant aux pénalités de gap sont calculées à l'aide d'un algorithme EM (Expectation-Maximisation) non-supervisé.

Un algorithme EM, proposé par [Dempster et al., 1977], est un algorithme itératif permettant l'estimation des paramètres d'une loi de probabilité, en s'appuyant d'une part sur des données disponibles (les données observées), et d'autre part sur des données cachées (ou latentes). Son objectif est de déterminer l'estimateur du maximum de vraisemblance du paramètre

de la loi considérée. Cette approche est composée de deux étapes : *(i)* étant donné un modèle et des données, de nouveaux paramètres sont estimés pour le modèle ; *(ii)* étant donnés les nouveaux paramètres et l'ancien modèle, un nouveau modèle est estimé. La procédure est répétée jusqu'à la stabilisation du modèle et des paramètres.

Partial-Ordered Alignment [Lee et al., 2002, Lee, 2003, Grasso and Lee, 2004]

POA est un algorithme utilisant des graphes partiellement ordonnés (appelés PO-MSA) au lieu de profils généralisés pour représenter les séquences alignées (cf. figure **??**). Les problèmes causés par la perte de l'information inhérente aux profils disparaissent avec cette approche. L'utilisation des graphes reflète plus précisément le contenu biologique des séquences. Le graphe formé par les séquences alignées n'a aucun rapport avec le graphe formé par la structure de guidage utilisée pour déterminer l'ordre d'alignement des séquences dans l'approche progressive.

Dans [Lee et al., 2002], les auteurs proposent un algorithme de progammation dynamique permettant d'aligner une séquence avec un PO-MSA. L'application directe de cette méthode est un algortihme itératif pour un alignement multiple de séquences. Les séquences sont alignées une à une avec le PO-MSA précédemment calculé. Cette première version, apelée PO-MSA itérative, présente quelques faiblesses, notamment propres à l'approche itérative. L'ordre des séquences à aligner est important : l'alignement final diffère en fonction de celui-ci. Cet inconvénient est absent lors d'un alignement progressif, car l'ordre d'alignement des séquences est déterminé par la structure de guidage. Pour cela, dans [Grasso and Lee, 2004], les auteurs ont étendu l'utilisation des PO-MSA en présentant un algorithme de programation dynamique permettant d'aligner deux PO-MSA.

Cet algorithme, qui détermine l'alignement optimal parmi un ensemble d'alignements, repose sur le principe de l'algorithme standard d'alignement par programmation dynamique. Cela implique le calcul d'un score et l'alignement de chaque paire possibles (résidus, insertion/délétion), par détermination du meilleur chemin. S'il s'agit de l'alignement d'une simple séquence, le chemin ne parvient que d'un seul prédecesseur. S'il s'agit de l'alignement d'un ou plusieurs PO-MSA, les prédecesseurs peuvent être multiples. Il faut donc considérer toutes les combinaisons possibles de prédecesseurs afin de trouver le chemin de meilleur score.

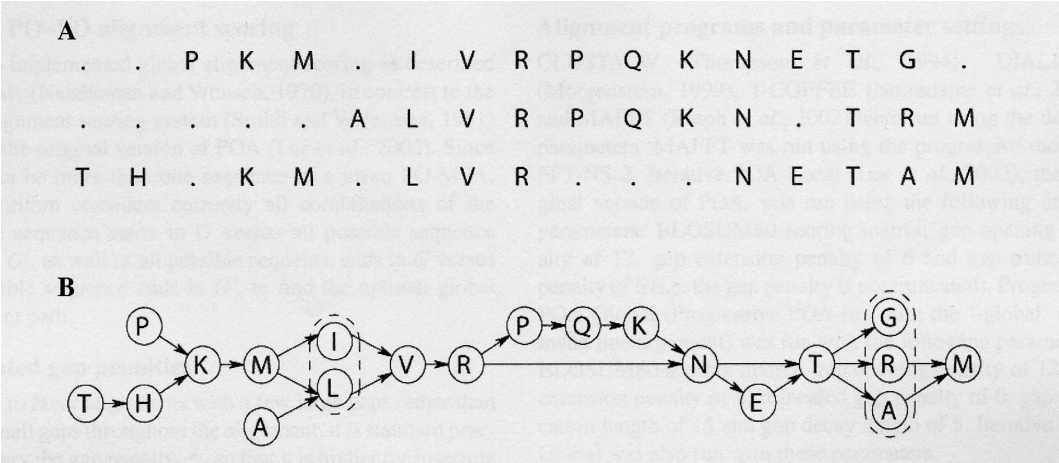


Fig. 3.10: Représentation d'un alignement multiple sous forme linéaire (A) et sous forme de graphe (B). Figure issue de [Lee et al., 2002]

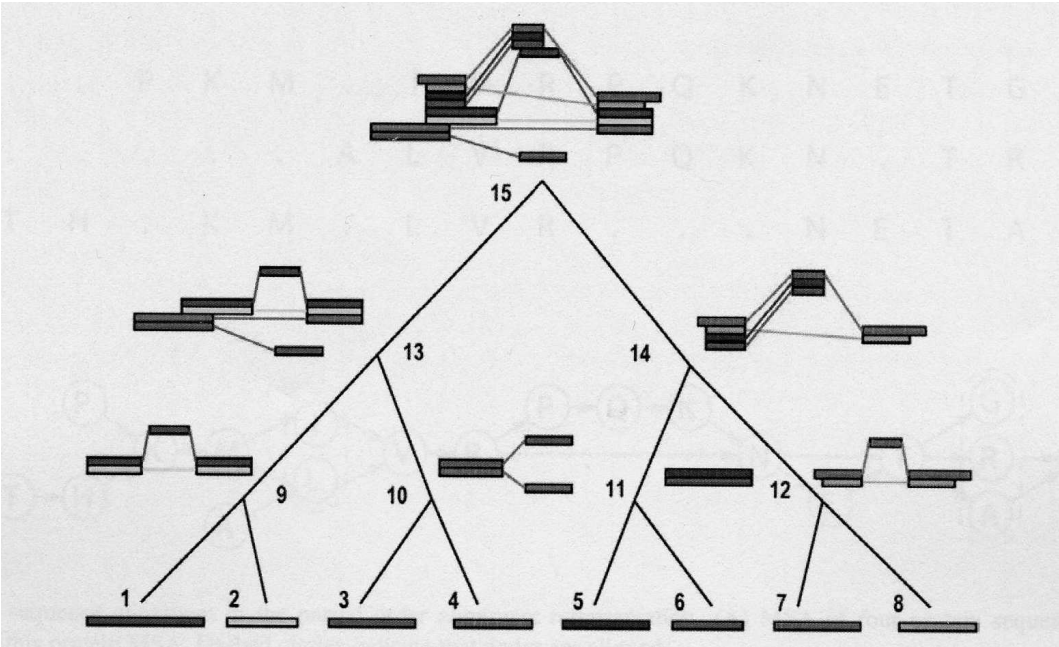


Fig. 3.11: Alignement multiple progressif à partir de graphes partiellement ordonnés. Figure issue de [Lee et al., 2002] – Les séquences à la base de l'arbre sont représentées sous la forme de graphes très simples, graphes qui vont se complexifier afin d'obtenir le graphe final représentant l'alignement

[Grasso and Lee, 2004] montrent l'intérêt et la rapidité de leur méthode. Ils proposent d'incorporer la notion d'alignements partiellement ordonnés dans les méthodes existantes, utilisant

traditionnellement les profils linéaires.

Ces nouvelles méthodes offrent de très bons résultats, et l'on ne retrouve plus la distinction entre global et local. MAFFT, MUSCLE, PROBCONS ont été appliquées à l'exemple des tRNA-synthétases et alignent correctement les trois domaines caractéristiques de ces protéines, sans distinction entre méthodes globales ou locales. Les gains en précision apportés par les nouveaux systèmes de score sont donc efficaces dans le cas des longues insertions. Les améliorations apportées par ces programmes sont indéniables.

T-Coffee [Notredame, 2002]

T-Coffee est un algorithme à part puisqu'il permet de tenir compte de données hétérogènes pour réaliser l'alignement multiple de séquences. Ces données peuvent être des alignements globaux, des alignements locaux, des données de structure... L'utilisation d'un grand nombre de données permet une nette amélioration de la qualité des alignements, tout en sacrifiant la rapidité de calcul de ceux-ci.

Dans [Notredame, 2002], les auteurs ont choisi de montrer l'intérêt de combiner les approches locale et globale. Sur la figure 3.12 page suivante, les différentes étapes de l'algorithme sont représentées. Les données utilisées en entrée du programme sont des alignements par paire issus de deux programmes d'alignements : ClustalW (pour le global) et Lalign (pour le local). Une bibliothèque primaire est alors constituée en pondérant les alignements, par un simple processus d'addition. Si une paire est dupliquée dans les deux approches, elle n'est représentée que par une seule entrée, affectée d'un poids égal à la somme des deux poids. Cette première bibliothèque pourrait être utilisée pour calculer un alignement multiple. Cependant, la valeur de l'information contenue dans la bibliothèque est énormément augmentée en examinant la consistance de chaque paire de résidus avec les paires de résidus de tous les autres alignements. Un poids reflétant cette consistance est alors assigné. Cette étape est appelée "extension" de la bibliothèque. C'est à partir de cette bibliothèque étendue qu'est calculé, de façon classique, l'alignement progressif des séquences.

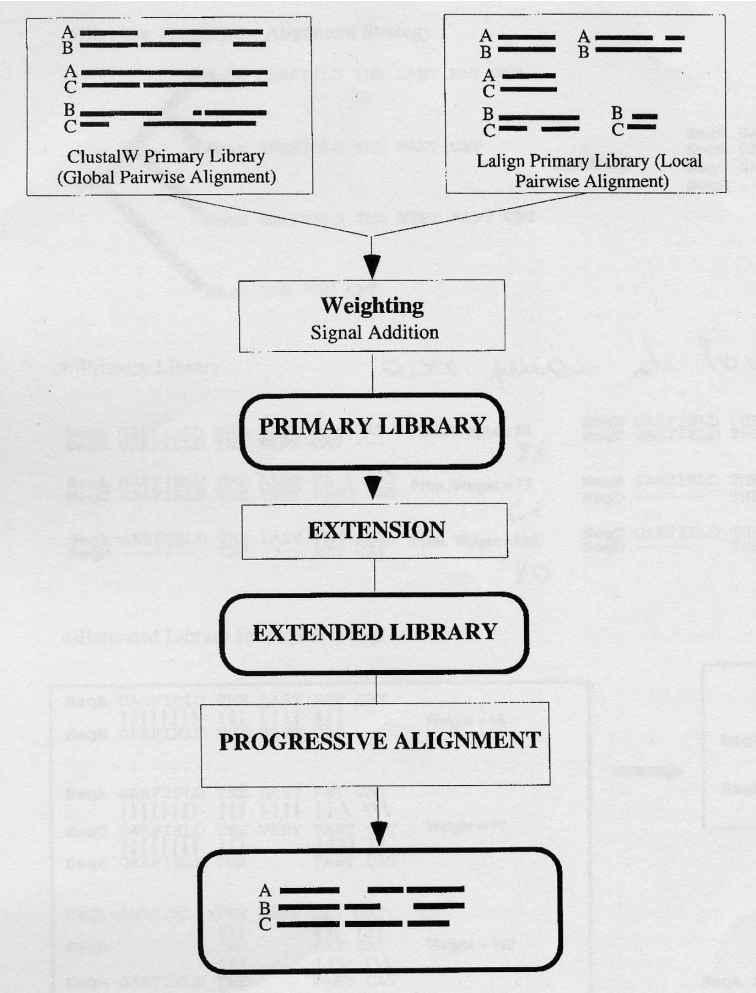


Fig. 3.12: Les différentes étapes de T-Coffee. Figure issue de [Notredame, 2002] – Les alignements provenant de sources hétérogènes vont être pondérés de façon à quantifier chaque source d'information. Une bibliothèque primaire de données va être étendue jusqu'à obtenir une bibliothèque permettant de réaliser un alignement progressif.

Remarque : [Loytynoja and Milinkovitch, 2003] ont proposé une approche progressive basée sur des triplets de séquences plutôt que sur des paires.

3.4.3 Etape de consolidation

Cette étape intervient lorsque l'alignement multiple progressif a été construit. Cet alignement est alors considéré comme solution initiale et sert de base à différentes stratégies pour l'améliorer.

[Corpet, 1988] est la première à avoir proposé une étape succédant à l'alignement multiple

progressif. Pour calculer le score affecté à un alignement, les méthodes *sum-of-pairs* calculent tous les scores des séquences 2 à 2. A partir de ces scores, une nouvelle classification hiérarchique est construite. Si celle-ci est semblable à celle de l'étape 2, l'algorithme prend fin. Sinon, on calcule un nouvel alignement à partir de ce nouvel arbre. Même si mathématiquement, la convergence de ce processus itératif est difficile à prouver, en pratique, l'auteur a noté que seules une ou deux itérations suffisaient.

[Wallace et al., 2005] ont réalisé une étude de cette étape, simple à la fois en terme d'implémentation, en temps de calcul et en utilisation de la mémoire. Cette approche est très efficace et apporte une amélioration significative pour la précision des alignements. De plus, les auteurs ont montré également que l'étape d'itération peut être utilisée directement dans l'alignement progressif : chaque sous-alignement à chaque étape de l'alignement progressif est amélioré. Ainsi, l'impact du problème du minimum local, propre à l'approche progressive, peut être réduit.

3.5 Conclusion

Le nombre de données génomiques générées automatiquement croît de façon importante. Afin de traiter ces données, des outils informatiques existent. L'alignement de séquences est une méthode permettant de prédire des informations de structure (secondaire ou tertiaire), de déterminer si une nouvelle séquence appartient à une famille de protéines ou non. L'alignement multiple de séquences est également une étape préliminaire à un grand nombre d'analyses de séquences, d'analyses phylogénétiques. Les approches permettant de calculer les alignements, c"étapeest à dire d'identifier les mutations qui ont eu lieu lors de l'évolution, sont donc importantes.

Dans ce chapître, nous avons tout d'abord présenté les façons de modéliser ces mutations : les matrices de score qui permettent de quantifier les substitutions d'acides aminés ; les pénalités de gaps (insertion et extension) qui sont des paramètres empiriques. L'introduction de ces modèles a permis de définir les méthodes d'alignement de deux séquences [Needleman and Wunsch, 1970, Smith and Waterman, 1981] ainsi que les méthodes de recherche dans les banques de séquences [Lipman and Pearson, 1985, Altschul et al., 1990]

Enfin, nous avons décrit les différentes approches aboutissant à un alignement multiple de séquences. Un grand nombre de programmes existent. Leurs résultats respectifs ont été comparés à l'aide de bases d'alignements de référence. Ces comparaisons ont permis de déterminer quels cas étaient correctement résolus par quel type d'approche. Cependant, la précision des méthodes automatiques chute de façon importante quand la similarité des séquences se situent dans la zone d'ombre (moins de 30%).

L'approche progressive, offrant un bon compromis entre temps de calcul et précision, utilise

une structure de guidage pour déterminer l'ordre des séquences à aligner. La première étape (*i.e.* l'alignement de toutes les paires de séquences) et la troisième (*i.e.* l'alignement progressif des séquences en suivant les branches de l'arbre) ont été très étudiées. Ces études ont non seulement permis d'améliorer la qualité d'un alignement mais aussi, de réduire le temps de calcul nécessaire. Ces études se sont appliquées à résoudre le problème du minimum local souvent rencontré lors de l'utilisation de l'approche progressive.

L'ordre d'alignement des séquences est également très important. Cet ordre est déterminé par la structure de guidage, et dépend donc de la technique choisie pour calculer celle-ci. Souhaitant utiliser l'ordre induit par la classification pyramidale pour fixer l'ordre d'alignement des séquences, nous avons donc réalisé une étude de l'influence de cette structure sur la précision des alignements multiples. Nous souhaitons répondre à différentes questions : l'étape de construction de la structure de guidage a t'elle une influence significative sur la qualité de l'alignement ? , une bonne structure de guidage permet-elle de réduire le nombre d'itérations de l'étape d'amélioration ?

Chapitre 4

La structure de guidage, rôle dans un alignement progressif

Les méthodes d'alignements multiples décrites dans le chapitre précédent sont primordiales pour de nombreuses analyses de bio-informatique (*e.g* analyses phylogénétiques, prédictions de structures). Ces dernières années de nombreuses avancées ont permis une amélioration sensible des résultats obtenus par les approches classiques dites progressives. Un algorithme d'alignement multiple progressif, à l'exemple de CLUSTALW [Thompson et al., 1994], construit l'alignement final en alignant itérativement toutes les séquences. Cet algorithme fonctionne en trois étapes distinctes :

1. Alignement de toutes les séquences par paires. Les scores attribués à ces alignements permettent de construire une matrice de distance entre séquences
2. Construction d'une structure de guidage à partir des distances entre séquences
3. Calcul de l'alignement final en parcourant la structure de guidage.

Apparue dans les années 80, cette méthode offre le meilleur compromis entre temps de calcul et précision des alignements. Des améliorations se sont depuis toutes attachées à modifier la façon d'aligner les paires de séquences (étape 1) et/ou l'alignement progressif des séquences (étape 3). En revanche, l'influence de la structure de guidage dans l'algorithme n'a fait l'objet d'aucune étude à ce jour. Elle est traditionellement calculée en utilisant l'algorithme de reconstruction phylogénétique appelé "Neighbor Joining" [Saitou and Nei, 1987]. Nous avons donc entrepris d'étudier plus particulièrement cette étape.

Par ailleurs, nous avons souhaité appliquer la classification pyramidale comme structure de guidage afin d'utiliser ses propriétés particulières d'inférence d'ordre partiel sur les données et de recouvrement. En effet, nous pensons que l'ordre induit sur les séquences par la pyramide peut être une information importante pour déterminer l'ordre des séquences à aligner. De plus,

la propriété de recouvrement devrait permettre de représenter au mieux la modularité des objets biologiques dont il est important de tenir compte pour établir un alignement multiple.

Dans un premier temps, afin de montrer l'importance d'une bonne structure de guidage pour la qualité d'un alignement multiple, nous avons réalisé une étude comparative en utilisant différentes bases d'alignements de référence. Nous avons sélectionné un ensemble de méthodes de construction d'arbre (décrites dans le chapitre 1 page 15) et nous les avons implémentées dans ClustalW, l'algorithme de référence. Nous avons choisi ce logiciel également pour sa disponibilité et pour sa simplicité, ce programme permet d'intervenir au niveau des différentes étapes. En plus du Neighbor-Joining, la méthode de construction d'arbre par défaut de ClustalW, nous étudierons : BioNJ [Gascuel, 1997] et Weighbor [Bruno et al., 2000], deux méthodes dérivées du NJ ; d'autres méthodes également basées sur le principe d'évolution minimum, FastMe [Desper and Gascuel, 2002] ; ainsi que la classification ascendante hiérarchique [Gordon, 1996b] et la classification pyramidale ; ces deux algorithmes de classification ayant été implémentés par nos soins. Pour cette dernière méthode, nous avons développé un algorithme permettant de transformer une pyramide en hiérarchie ordonnée. Nous utiliserons ains la propriété d'ordre sur les objets induit par la pyramide pour déterminer un ordre parmi les séquences à aligner. La première partie de ce chapitre sera consacrée à la description de cet algorithme.

Dans un second temps, nous présenterons une approche mixte, basée sur les stratégies d'alignement local et global ; la méthodologie précédemment proposée pour utiliser l'ordre induit par la pyramide étant une approximation. Ce nouvel algorithme utilise la propriété de recouvrement des pyramides : un objet peut appartenir à deux classes. L'utilisation des séquences appartenant à deux classes, permet de faire le lien entre deux classes préalablement existantes. On pourra aussi traiter ces classes de façons distinctes en s'appuyant sur la réalité biologique. Les propriétés des pyramides seront ainsi exploitées au mieux. Après avoir présenté l'algorithme général, nous décrirons un exemple détaillé montrant la pertinence de cette approche, ainsi que plusieurs perspectives.

La première partie de ce chapitre est consacrée à la description de l'algorithme permettant de hiérarchiser une pyramide. Nous présenterons ensuite l'étude de l'influence de la structure de guidage, au cours de laquelle sera utilisée cette implémentation. Pour terminer, nous détaillerons l'approche mixte que nous avons développée afin d'optimiser l'application de la classification pyramidale aux alignements multiples de séquences.

4.1 Transformation d'une pyramide en hiérarchie

La classification pyramidale induit un ensemble d'ordres compatibles sur les objets étudiés. Afin d'étudier uniquement l'impact d'un de ces ordres, nous avons défini un algorithme per-

mettant de calculer une hiérarchie ordonnée à partir de la matrice de Robinson équivalente à la pyramide. Auparavant, [Brossier, 1980] a remarqué le fait qu'un arbre de classification hiérarchique admet plusieurs représentations équivalentes possibles liées à l'ordre des objets sur l'axe horizontal. Il se pose alors le problème de déterminer laquelle, au sens de la relation d'ordre qu'elle induit sur l'ensemble des objets, est la plus proche des données initiales.

4.1.1 Algorithme de [Diday, 1984b], basé sur une matrice ultramétrique

[Diday, 1984b] a proposé une méthode de hiérarchisation d'une pyramide en transformant la matrice de Robinson en matrice ultramétrique. Une ultramétrique u, est un indice de dissimilarité défini sur Ω, qui vérifie l'inégalité suivante :

$$\forall x, y, z \in \Omega, u(x, z) \leq max(u(x, y), u(y, z))$$

[Benzécri, 1973] a démontré l'existence d'une bijection entre l'ensemble des hiérarchies indicées et celui des ultramétriques. Ainsi, en transformant la matrice de Robinson équivalente à la pyramide en matrice ultramétrique, on obtient une hiérarchie ordonnée. Le principe de cet algorithme est de repérer chaque triplet de valeurs de la matrice ne vérifiant pas l'inégalité précédente et de modifier les valeurs de ce triplet afin d'obtenir un triangle isocèle .

Soit Ω l'ensemble des individus, P la matrice de Robinson contenant les indices d'agrégation, notés μ. On note a, b et c les indices d'agrégation d'un triplet, avec $c < a < b$. On repère ainsi ces éléments dans la matrice $a = \mu(i, j-1)$, $b = \mu(i, j)$ et $c = (i+1, j)$ avec i l'indice des lignes et j l'indice des colonnes, croissants depuis la diagonale. L'algorithme est le suivant :

1. **Recherche** du couple de valeurs consécutives (a, b) et (b, c) ayant l'écart minimum $(a, b) = min(\mu(i, j-1), \mu(i, j))$ ou $(b, c) = min(\mu(i, j), \mu(i+1, j))$
2. **Remplacement** de la valeur de a par :
   – la valeur de c, dans le cas du critère du minimum
   – la valeur de b, dans le cas du critère du maximum
   Le triplet (a, b, c) forme alors un triangle isocèle.
3. **Test d'arrêt :** Tant que la matrice n'est pas ultramétrique, on recommence à la première étape.

Par exemple, sur la figure 4.1 page suivante, le couple de valeurs ayant l'écart minimum est $\mu(A, E), \mu(B, E)$, soit 10-9 = 1, et le triplet concerné est $\mu(A, D), \mu(A, E), \mu(B, E)$, soit (8, 9, 10). Dans le cas du minimum, 9 sera modifié en 8, tandis que dans le cas du maximum, il sera modifié en 10. L'algorithme se poursuit tant que tous les triplets ne forment pas des triangles isocèles. Un couple de valeurs consécutives peut donc être repéré et ces deux valeurs peuvent être modifiées plusieurs fois. Cet algorithme a donc une forte complexité. De plus, nous pouvons voir que les hiérarchies conservent la même topologie, quel que soit le critère choisi.

Seul l'indiçage change. Cet algorithme n'est donc pas efficace aussi bien en terme de temps de calcul qu'en terme de résultat.

Matrice de Robinson :

A	0	1	5	8	10
B		0	2	6	9
C			0	3	7
D				0	4
E					0

Minimum

A	0	1	2	3	4
B		0	2	3	4
C			0	3	4
D				0	4
E					0

Maximum

A	0	1	5	8	10
B		0	5	8	10
C			0	8	10
D				0	10
E					0

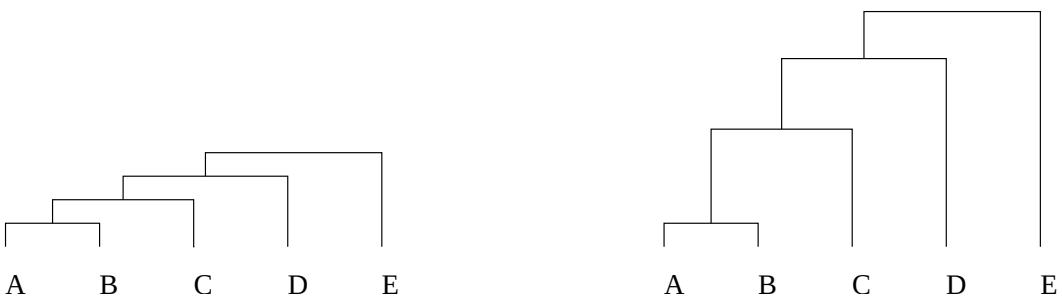


Fig. 4.1: Algorithme de hiérarchisation d'une pyramide proposé par [Diday, 1984b] – Exemples, avec deux crières de modification.

4.1.2 Algorithme de hiérarchisation de pyramide : pyr2hier

Principe

Nous proposons un algorithme agglomératif simple et efficace, qui, en plus de la contrainte de relation d'inclusion des paliers, respecte la contrainte suivante : conserver l'ordre des individus. Cet algorithme, appelé pyr2hier, utilise comme point de départ la matrice de Robinson équivalente à la pyramide. A chaque étape, la valeur minimale de la matrice est sélectionnée. Les deux éléménts concernés sont regroupés en un palier. La matrice est alors réduite, en rem-

plaçant les deux éléments par le palier et en recalculant les distances de ce palier avec les autres éléments à l'aide du critère choisi (minimum, maximum, moyenne...). Ce processus est répété jusqu'à ce que tous les éléments soient regroupés. La matrice équivalente à la pyramide étant robinsonienne, la valeur minimale de celle-ci est située sur la première diagonale. Sa recherche est donc plus rapide que dans le cas de la CAH (décrite section 2.1.1 page 46), où il faut parcourir toutes les valeurs de la matrice. De plus, ceci a l'avantage de conserver l'ordre des objets : sur la première diagonale se trouvent en effet les distances entre deux objets consécutifs. Le fonctionnement de cet algorithme est illustré par la figure 4.2 page suivante.

Formalisation

Soit Ω l'ensemble des individus, P la matrice de Robinson contenant les indices d'agrégation, notés μ. L'algorithme, appelé pyr2hier, est le suivant :

- **Initialisation :** Les seuls paliers existants sont les singletons de Ω. Les distances entre les objets sont stockées dans la matrice P.
- **Agrégation :** Deux paliers p^* et q^* sont agrégés si l'indice d'agrégation $\mu(p^*, q^*)$ est minimum parmi l'ensemble des indices d'agrégation $\mu(p, q)$, situés sur la première diagonale de P, où $(p, q) \in P \times P$. L'ordre induit par la pyramide est respecté par p^* et q^*, ceux-ci étant deux éléments consécutifs.
- **Test d'arrêt :** L'algorithme prend fin quand le palier que l'on vient de créer, $(p^* \cup q^*)$, est égal à Ω. Dans le cas inverse on reprend l'exécution de l'algorithme à partir de la phase d'agrégation.

Preuve

La preuve de l'algorithme pyr2hier complet est la même que celle de la CAH et ne sera pas donnée ici. En effet, la recherche de la valeur minimum sur la première diagonale revient à chercher la valeur minimum de la matrice car celle-ci est robinsonienne. La contrainte d'ordre est respectée par l'algorithme pyr2hier car les paliers agrégés sont composés d'éléments appartenant à la première diagonale de la matrice ; c'est à dire que ces éléments sont consécutifs dans la pyramide.

A présent, nous allons prouver que, lors du recalcul des distances, la nouvelle matrice est toujours de Robinson. Il est à noter que [Bertrand and Diday, 1990b] ont montré que l'ensemble des ultramétriques est inclus dans l'ensemble des indices pyramidaux.

Soient D la matrice triangulaire supérieure de taille n, i et j les indices de ligne et colonne de D. On notera $d_{i,j}$ la valeur de l'élément repéré par i et j. Sur la figure 4.3 page 133, cet élément est représenté avec son voisinage dans la matrice.

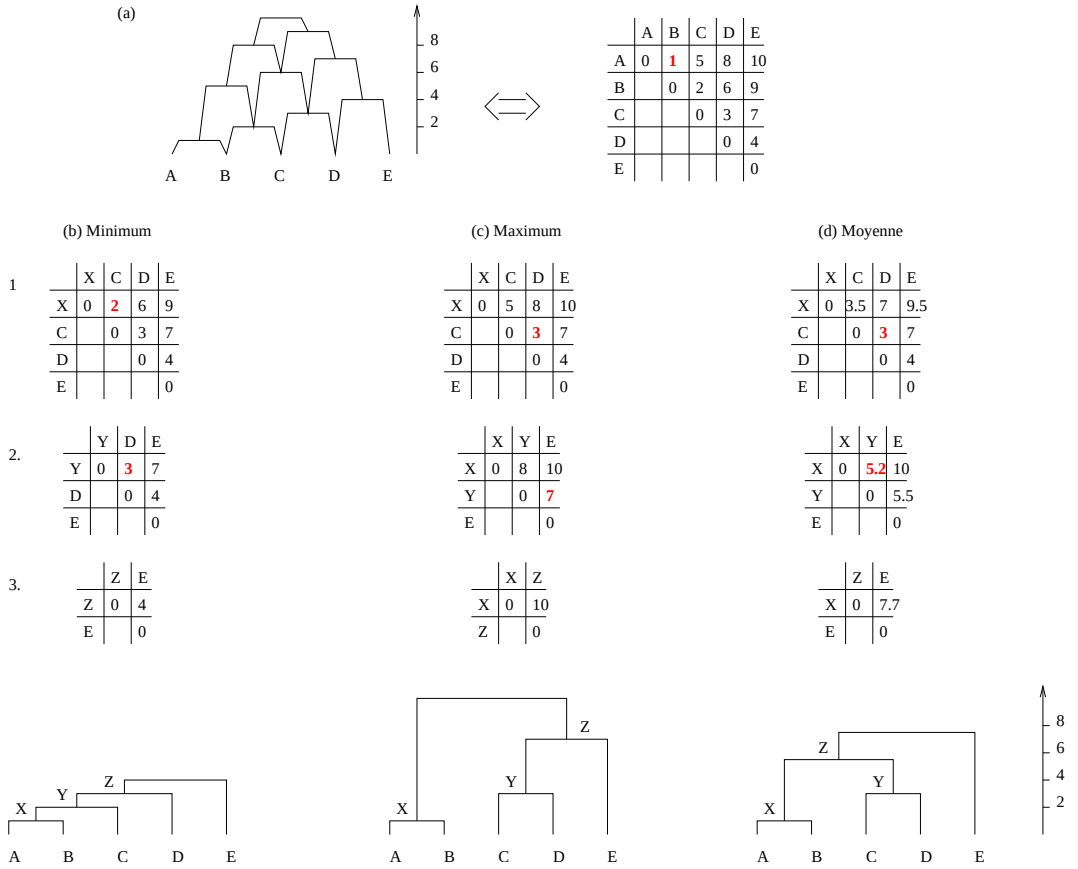


Fig. 4.2: Algorithme pyr2hier – Exemple (la matrice est la même que dans l'exemple précédent). Les valeurs minimales sont indiquées en rouge. (a) La pyramide et la matrice de Robinson équivalente sont les points de départ de l'algorithme pyr2hier. Les différentes étapes aboutissant à une hiérarchie sont décrites pour 3 critères différents : (b) minimum, (c) maximum et (d) moyenne.

D'après la définition d'une matrice de Robinson, quelques assertions utiles peuvent être formulées en utilisant les notations de la figure 4.3 page ci-contre, pour A :

$$d(i-1,j-1) \leq d(i-1,j) \leq d(i-2,j) \tag{4.1}$$

$$d(i-1,j-1) \leq d(i-2,j-1) \leq d(i-2,j) \tag{4.2}$$

$$d(i-1,j) \leq d(i-1,j+1) \tag{4.3}$$

De même, pour B :

$$d(i+1,j+1) \leq d(i,j+1) \leq d(i,j+2) \tag{4.4}$$

$$d(i+1,j+1) \leq d(i+1,j+2) \leq d(i,j+2) \tag{4.5}$$

$$d(i,j+1) \leq d(i-1,j+1) \tag{4.6}$$

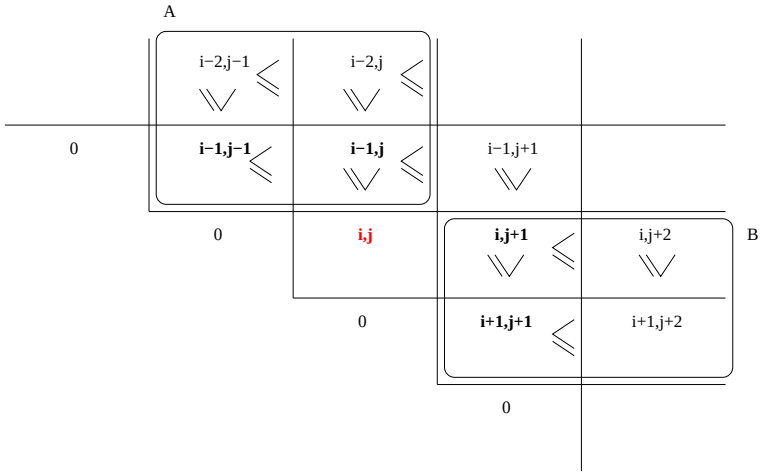


Fig. 4.3: Valeur minimale de la matrice et son voisinage – A et B correspondent aux deux ensembles à modifier lors du recalcul des distances. La valeur minimale est indiquée en rouge et en gras. Les valeurs en gras sont celles à réduire (avec le critère d'agrégation choisi), en fonction de leur environnement.

Nous allons à présent prouver que, lors du recalcul des distances dans la matrice, celle-ci reste de Robinson. Pour cela, nous allons utiliser le principe de conservation du sens des inégalités.

Critère du minimum Pour l'ensemble A, nous avons :
$min(d(i-1,j-1),d(i-1,j)) \leq min(d(i-2,j-1),d(i-2,j))$
$d(i-1,j-1) \leq d(i-2,j-1)$, d'après 4.1 et 4.2,
et
$min(d(i-1,j-1),d(i-1,j)) \leq d(i-1,j+1)$
$d(i-1,j-1) \leq d(i-1,j) \leq d(i-1,j+1)$, d'après 4.1, 4.2 et 4.3.

De même, pour l'ensemble B :
$min(d(i+1,j+1),d(i,j+1)) \leq min(d(i+1,j+2),d(i,j+2))$
$d(i+1,j+1) \leq d(i+1,j+2)$, d'après 4.4 et 4.5,
et
$min(d(i+1,j+1),d(i,j+1)) \leq d(i-1,j+1)$
$d(i+1,j+1) \leq d(i,j+1) \leq d(i-1,j+1)$, d'après 4.4, 4.5 et 4.6.

Critère du maximum Pour l'ensemble A :
$max(d(i-1,j-1),d(i-1,j)) \leq max(d(i-2,j-1),d(i-2,j))$
$d(i-1,j) \leq d(i-2,j)$, d'après d'après 4.1 et 4.2,
et
$max(d(i-1,j-1),d(i-1,j)) \leq d(i-1,j+1)$

$d(i-1,j) \leq d(i-1,j+1)$, d'après d'après 4.1, 4.2 et 4.3.

De même, pour l'ensemble B :
$max(d(i+1,j+1), d(i,j+1)) \leq max(d(i+1,j+2), d(i,j+2))$
$d(i,j+1) \leq d(i,j+2)$, d'après 4.4 et 4.5,
et
$max(d(i+1,j+1), d(i,j+1)) \leq d(i-1,j+1)$
$d(i,j+1) \leq d(i-1,j+1)$, d'après 4.4, 4.5 et 4.6.

Critère de la moyenne Pour l'ensemble A :
$d(i-1,j-1) \leq d(i-2,j-1)$ et $d(i-1,j) \leq d(i-2,j)$
d'où $\frac{d(i-1,j-1)+d(i-1,j)}{2} \leq \frac{d(i-2,j-1)+d(i-2,j)}{2}$, d'après d'après 4.1 et 4.2,
et
$d(i-1,j-1) \leq d(i-1,j+1)$ et $d(i-1,j) \leq d(i-1,j+1)$
d'où $\frac{d(i-1,j-1)+d(i-1,j)}{2} \leq d(i-1,j+1)$, d'après 4.1, 4.2 et 4.3.

De même, pour l'ensemble B :
$d(i+1,j+1) \leq d(i+1,j+2)$ et $d(i,j+1) \leq d(i,j+2)$
d'où $\frac{d(i+1,j+1)+d(i,j+1)}{2} \leq \frac{d(i+1,j+2)+d(i,j+2)}{2}$, d'après 4.4 et 4.5,
et
$d(i+1,j+1) \leq d(i-1,j+1)$ et $d(i,j+1) \leq d(i-1,j+1)$
d'où $\frac{d(i+1,j+1)+d(i,j+1)}{2} \leq d(i-1,j+1)$, d'après 4.4, 4.5 et 4.6.

Cet algorithme de hiérarchisation de pyramide est simple et efficace. Sa complexité en temps est en $\mathcal{O}(n^2)$ alors que celui de [Diday, 1984b] est en $\mathcal{O}(n^2 logn)$. Nous l'avons donc utilisé dans l'étude comparative suivante ayant pour but de mesurer l'influence de la structure de guidage. Une des perspectives possibles de ce travail est tout d'abord la comparaison de notre algorithme avec un algorithme permettant directement l'ordonnancement d'une hiérarchie par permutation des paliers puis la proposition d'un nouvel algorithme de ce type.

4.2 Etude comparative des méthodes de construction d'arbres

Même si ces méthodes sont habituellement classées sous différentes catégories (reconstruction phylogénétique, classification...), nous les regroupons sous une seule et même appellation : méthodes de construction d'arbres ou méthodes de distance (décrites dans le chapitre 1 page 15). Dans notre cas, nous souhaitons, à partir de la matrice de distance issue des alignements deux à deux, contruire l'arbre qui guidera l'alignement progressif des séquences. Il a été montré que

l'ordre d'alignement des séquences (déterminé par la structure de guidage) est un paramètre important pour la qualité de l'alignement final (voir la section 3.4 page 115).

4.2.1 Les méthodes de construction d'arbre

D'une part, nous avons choisi de comparer des méthodes "classiques" de construction d'arbre et d'autre part la CAP.

Les méthodes de construction d'arbre "classiques" sont très similaires de par leur principe et sur les données traitées. Elles sont basées sur des algorithmes agglomératifs et possèdent toutes le même type de données en entrée (une matrice de distance) et en sortie (un arbre). Ceci a pour effet de faciliter l'implémentation pour pouvoir les comparer. Ces techniques, à partir d'une matrice de distance, vont calculer un arbre binaire représentant au mieux les données initiales. Seule, la façon de calculer cette structure est différente et ces différences sont mises en avant dans le chapitre 1 page 15. Ces méthodes sont : la classification ascendante hiérarchique (programme développé par nos soins), BioNJ (GAS97), Weighbor (BRU00) et FastME (DES02). Le Neighbor-Joining est la méthode par défaut de ClustalW. Le tableau 4.1 page suivante présente les lignes de commande utilisées pour exécuter les différents programmes que nous présentons ci-dessous. Par la suite, nous noterons la CAH avec les critères du lien complet CAHc, de la moyenne CAHm, du lien simple CAHs et de Ward CAHw.

Pour la classification pyramidale, nous avons utilisé l'algorithme de la CAP consolidé par notre approche (voir le chapitre 2 page 43) : puis nous avons utilisé l'algorithme pyr2hier, décrit dans la première partie de ce chapitre, pour obtenir une pyramide hiérarchisée (i.e. un arbre utilisable par ClustalW).

4.2.2 Implémentation dans ClustalW

ClustalW est le programme de référence pour calculer un alignement multiple de séquences. Il construit l'alignement final en trois étapes distinctes :

- Etape 1 : Alignement de toutes les séquences par paires
- Etape 2 : Construction d'une structure de guidage
- Etape 3 : Calcul de l'alignement final en incorporant progressivement toutes les séquences selon la structure de guidage.

Nous avons choisi ClustalW également pour sa disponibilité et pour sa simplicité : ce programme permet d'intervenir au niveau des différentes étapes. Nous avons modifié la version initiale du code source de ce logiciel (version 1.83) afin d'obtenir la matrice de distance obtenue après l'étape d'alignement par paire. De plus, ce logiciel offre la possibilité nécessaire nous permettant d'implémenter d'autres méthodes de construction d'arbre pour calculer une struc-

Méthodes de construction d'arbre	Ligne de commande
BioNJ	bionj matrix.dst bionj.tree
CAH Complet	cah_exec -i matrix.dst -c complet -o cah_compl.tree
Moyenne	cah_exec -i matrix.dst -c mean -o cah_mean.tree
Simple	cah_exec -i matrix.dst -c single -o cah_singl.tree
Ward	cah_exec -i matrix.dst -c ward -o cah_ward.tree
FASTME BMEn	fastme -i matrix.dst -o bme.tree -b BME -s n
FASTME BMEb	fastme -i matrix.dst -o bmebnni.tree -b BME -s b
FASTME BMEo	fastme -i matrix.dst -o bmefnni.tree -b BME -s o
FASTME GMEn	fastme -i matrix.dst -o gme.tree -b GME -s n
FASTME GMEb	fastme -i matrix.dst -o gmebnni.tree -b GME -s b
FASTME GMEo	fastme -i matrix.dst -o gmefnni.tree -b GME -s o
NJ	Programme par défaut de ClustalW
WEIGHBOR	weighbor -b 20 -i matrix.dst -o weighbor.tree

Tab. 4.1: Tableau récapitulatif des différentes méthodes utilisées et des lignes de commande nécessaires pour exécuter les programmes. Le fichier en entrée, appelé matrix.dst, est la matrice de distance issue de la phase d'alignement par paire. Le fichier de sortie, appelé *.tree, est la structure de guidage qui sera utilisée lors de l'alignement multiple progressif final.

ture de guidage : utiliser un arbre de guidage fourni par l'utilisateur pour calculer l'alignement multiple.

Ceci est illustré sur la figure 4.4 page ci-contre. Le déroulement de la comparaison est le suivant :

- Etape 1 : On exécute ClustalW avec la ligne de commande notée (1). On obtient la matrice de distance *matrix.dst*. Le fichier de sortie appelé align_clustal est le fichier obtenu avec le Neighbor-Joining comme méthode de construction d'arbre.
- Etape 2 : On exécute la ligne de commande correspondant à la méthode choisie pour calculer la structure de guidage (4.1). On obtient la structure de guidage.
- Etape 3 : On exécute ClustalW avec la ligne de commande notée (3). La structure de guidage obtenue précédemment permet d'obtenir l'alignement final.

4.2.3 Résultats

Etude préliminaire

Dans un premier temps, nous avons réalisé une étude réduite à un nombre limité de méthodes de construction d'arbre sur la base d'alignements de référence Balibase. Pour cela, nous avons

choisi la classification hiérarchique (calculée avec l'algorithme de la CAH et différents critères d'agrégation), le Neighbor-Joining, BioNJ et la classification pyramidale. Balibase, qui est la base la plus fréquemment utilisée pour comparer les méthodes d'alignement, a été développée par les auteurs de ClustalW. C'est pourquoi nous avons choisi cette base pour réaliser cette première étude.

Un programme, bali_score, est fourni avec la base de données Balibase, permettant de comparer les alignements tests calculés aux alignements de référence. La façon dont les alignements de Balibase ont été construits est décrite dans la section 3.3.2 page 114. Les scores SP et TC fournis par le programme de Balibase sont décrits dans la section 3.3.2 page 113. Plus ces scores sont élevés et plus l'alignement test est proche de l'alignement de référence : ces alignements sont des comptages de résidus correctement alignés.

Afin de démontrer les différences significatives entre les méthodes de construction d'arbre, nous avons utilisé un test de rang de Friedman [Friedman, 1937]. Ce test d'analyse de variance teste l'hypothèse H_0 que k variables appariées proviennent de la même population ; c'est à dire

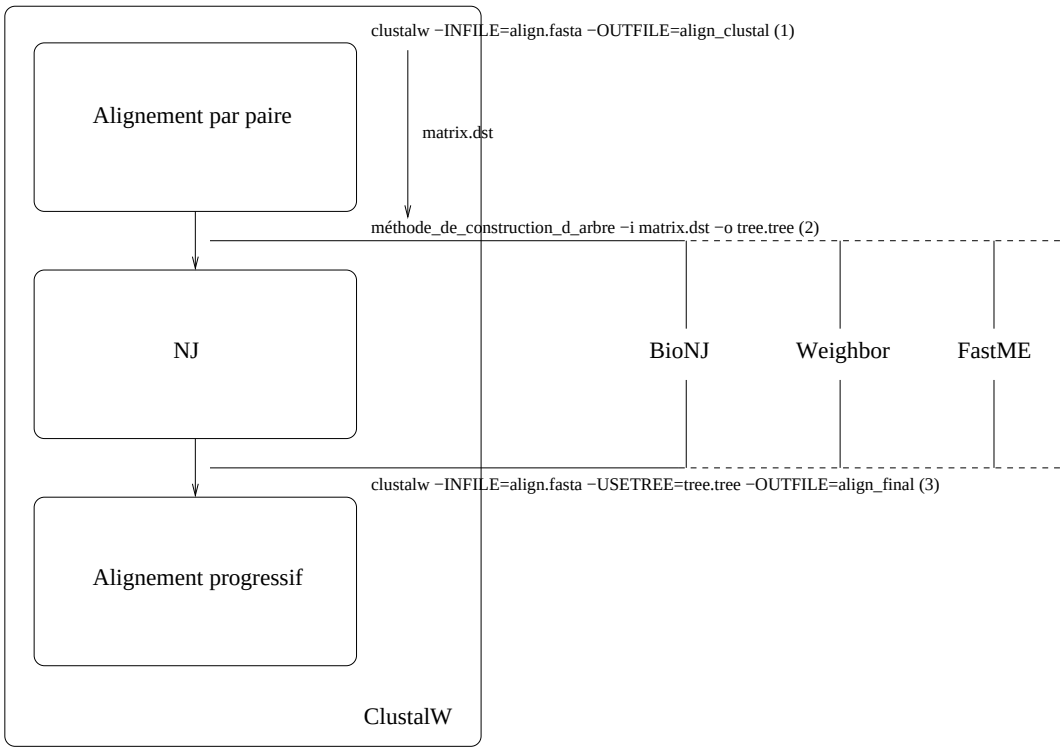


Fig. 4.4: Implémentation des différentes méthodes dans ClustalW La partie gauche de la figure représente les trois étapes classiques du logiciel ClustalW. La partie droite indique les méthodes de construction d'arbre que nous avons implémentées dans ClustalW à l'aide des deux lignes de commande indiquées.

qu'il n'y a pas de différence significative entre chaque variable k. Chaque individu est classé sur chacune des variables de rang k (variables de 1 à k). Le test s'appuie sur ces rangs. Lorsque une p-value obtenue avec ce test est inférieure à 0.05, cela signifie que l'on rejette l'hypothèse H_0 avec un seuil de 5%. Dans notre cas, ce test évalue statistiquement la significativité de la différence entre les scores obtenus par deux méthodes (variables). Le tableau 4.2 page 140 regroupe les résultats obtenus sur la totalité des données de la base, sans tenir compte des différentes catégories d'alignements. Peu de résultats sont statistiquement significatifs à 95% avec ce test. Cependant, la première observation est que le Neighbor-Joining, méthode utilisée par défaut par ClustalW, n'est pas la meilleure méthode. Malgré l'utilisation récurrente du test de Friedman à cet effet dans la littérature, nous avons formulé une critique sur ce travail. En effet, le test de Friedman est un test multiple et donc est mal adapté à la comparaison de méthodes deux à deux. Un test de Wilcoxon, dédié à la comparaison de deux méthodes, est plus judicieux. Ce test appliqué de la même façon sur les données donne des résultats comparables : aucune méthode (notamment la méthode utilisée par défaut) n'est significativement la meilleure.

Afin d'obtenir une représentation synthétique de la comparaison des méthodes, nous avons choisi d'utiliser les boxplots (figure 4.5 page suivante). Les données utilisées pour cette analyse sont l'ensemble des scores d'alignement. La boite à moustache ou boxplot est un résumé graphique d'une distribution. Le corps de la boîte est formé par le premier et le troisième quartile et est coupé par le deuxième quartile (médiane). Les valeurs suspectes potentiellement aberrantes sont situées au-delà des moustaches. Ainsi, la taille de la boîte symbolise l'étendue inter-quartile, la position de la médiane est un bon indicateur de la symétrie de la distribution, la taille des moustaches traduit la dispersion (les extrémités correspondant aux valeurs maximale et minimale). Ce type de diagramme permet de comparer facilement plusieurs séries de données, en terme de médianes, quartiles et valeurs extrêmes. Sur cette figure, BioNJ apparaît comme la meilleure méthode en moyenne et de part la faible dispersion de sa distribution : la taille de sa boîte est la plus petite. Cela signifie que BioNJ est la méthode offrant le moins de risque d'obtenir un alignement de mauvaise qualité. Nous rappelons que plus le score SP est élevé et plus l'alignement est de bonne qualité.

En étudiant en détail les différentes références de Balibase, nous pouvons voir que l'analyse des résultats est complexe. Le tableau 4.3 page 140 indique pour chacune des références la meilleure méthode et la moins bonne, en fonction des scores SP obtenus. Nous observons qu'aucune des méthodes n'est la meilleure dans tous les cas. La CAH avec le critère de Ward donne de mauvais résultats et est la moins bonne des méthodes pour quatre références sur 6. La méthode dérivée de la classification pyramidale ne se distingue pas par ses résultats. L'ordre apporté par cette structure ne semble pas être un grand apport pour la précision des alignements, tout du moins dans sa version hiérarchisée.

Ces résultats sont également mis en avant par les deux boxplots représentés sur la figure 4.6

page 141 ; la référence 2 comprend des familles de séquences alignées avec une séquence orpheline très divergentes et la référence 3 est composée d'alignements de sous-groupes de séquences avec moins de 25% d'identité entre ces groupes. Sur le premier boxplot concernant la référence 2, nous pouvons voir que la meilleure méthode est la CAH avec le critère de la moyenne alors que dans le cas de la référence 3, à droite, il s'agit de BioNJ. Aucune méthode ne s'impose dans tous les cas de données traitées. La nature, les propriétés des séquences à aligner sont des paramètres importants.

Les résultats obtenus à partir de cette étude préliminaire nous ont apporté beaucoup d'informations. Aucune méthode n'est la meilleure dans tous les cas rencontrés et la qualité des alignements obtenus à partir d'une méthode de construction d'arbre donnée dépend des propriétés des séquences à aligner. De plus, l'ordre sur les données apporté par la hiérarchisation des pyramides n'améliore pas la qualité des alignements. C'est pourquoi nous avons décidé de compléter cette étude en augmentant le nombre de méthodes utilisées et le nombre de bases d'alignements de référence. La CAH avec le critère de Ward sera retirée pour l'étude suivante afin de ne pas biaiser l'analyse des autres techniques. En effet, son comportement atypique par rapport aux autres méthodes, pourrait dissimuler des résultats sur celles-ci. De plus, nous avons choisi d'utiliser plus de critères sur les séquences telles que la longueur des séquences, le nombre de séquences à aligner, leur hydrophobicité. Notre but est de déterminer des modèles de décision en fonction des caractéristiques de séquences à aligner.

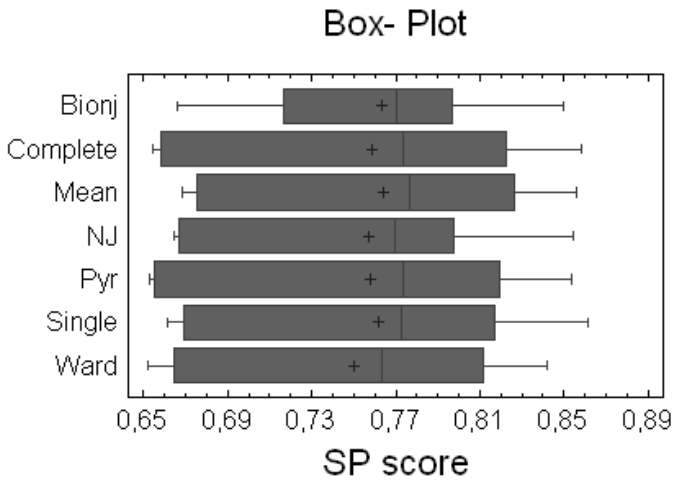


Fig. 4.5: Boxplot représentant les scores SP obtenus sur Balibase – Etude préliminaire. Les méthodes de construction d'arbre ont été testés sur la totalité de Balibase. Plus le score SP est élevé et plus l'alignement est de bonne qualité.

	BioNJ	CAHc	CAHm	NJ	CAHs	CAHw	Pyr2hier
BioNJ		0.298	0.628	0.093	0.832	0.102	0.467
CAHc	-0.285		-0.088	0.868	-0.467	0.071	0.861
CAHm	0.928	0.191		0.144	0.238	0.002	0.412
NJ	-0.018	-0.182	-0.024		-0.880	0.276	0.733
CAHs	0.858	0.140	-0.646	0.034		0.002	0.263
CAHw	-0.000	-0.000	-0.000	-0.119	-0.001		-0.086
Pyr2hier	0.736	-0.542	0.423	-0.358	0.147	0.098	

Tab. 4.2: Tableau récapitulatif des résultats obtenus à partir de l'étude préliminaire. Chaque entrée contient la p-value obtenue avec un test de Friedman. La partie inférieure gauche de la matrice contient les scores SP, la partie supérieure droite le score TC. Si la méthode à gauche est moins bien classée que la méthode comparée, la p-value est précédée d'un -. Les cases grisées indiquent une p-value < 0.05. * indique le critère d'agrégation de la CAH.

références	BioNJ	CAHc	CAHm	NJ	CAHs	CAHw	Pyr2hier
Réf 1_1	-		+				
Réf 1_2			+			-	
Réf 2					+	-	
Réf 3	+					-	
Réf 4					+		-
Réf 5						-	+

Tab. 4.3: Tableau récapitulant les performances des différents programmes de construction d'arbre en fonction de la qualité de l'alignement final. Le + indique le programme obtenant les meilleurs résultats en moyenne pour la référence donnée. Le - indique le moins bon. * indique le critère d'agrégation de la CAH.

Etude complémentaire

L'étude précédente sur Balibase nous a permis de déterminer que la nature de la structure de guidage, et donc également la méthode utilisée pour la calculer, ont une influence sur la précision de l'alignement multiple final. Cependant, aucun résultat global n'a pu être mis en avant, si ce n'est que le Neighbor-Joining n'est pas la meilleure méthode dans chacune des catégories étudiées. Dans cette section, nous présentons une étude approfondie en augmentant le nombre de critères sur les séquences à aligner (nombre, longueur, propriétés physico-chimiques) ; le nombre de bases d'alignements de référence (en plus de Balibase, nous utiliserons Sabmark) et le nombre de méthodes de construction d'arbres (notamment avec le logiciel FastME dont les algorithmes et leurs acronymes sont présentés dans la section 1 page 15). Nous avons choisi de retirer de l'étude la CAH avec le critère de Ward, en raison des mauvais résultats observés lors de l'étude préliminaire, afin que celle-ci ne fausse pas l'étude des autres méthodes ; son

comportement pourrait dissimuler des résultats sur les autres méthodes. ?ous avons également supprimer pyr2hier, devant son faible apport et pour ne pas complexifier l'interprétation des résultats déjà compliquée. Afin de pouvoir comparer les résultats fournis sur les différentes bases d'alignement, nous avons choisi d'exploiter le score SP de Balibase et le score FD de Sabmark, qui mesurent la sensibilité d'une méthode d'alignement (voir le paragraphe 3.3.2 page 113).

Sabmark Cette base d'alignements de référence est dédiée à l'étude des comportements des méthodes d'alignement multiple dans les cas de faible taux de similarité entre les séquences. Cette base est divisée en 2 catégories :

- les superfamilles de protéines, composée de séquences ayant moins de 50% de similarité
- la zone d'ombre (twilight zone), comprenant des séquences présentant moins de 25% d'identité.

D'après la définition de ces catégories, celles-ci sont proches des références pré-définies 1_1, 1_2 et 3 de Balibase, que nous avons utilisées précédemment. Nous vérifierons cela lors de la synthèse des conclusions relatives à chacune de ces bases. Nous avons également défini nos

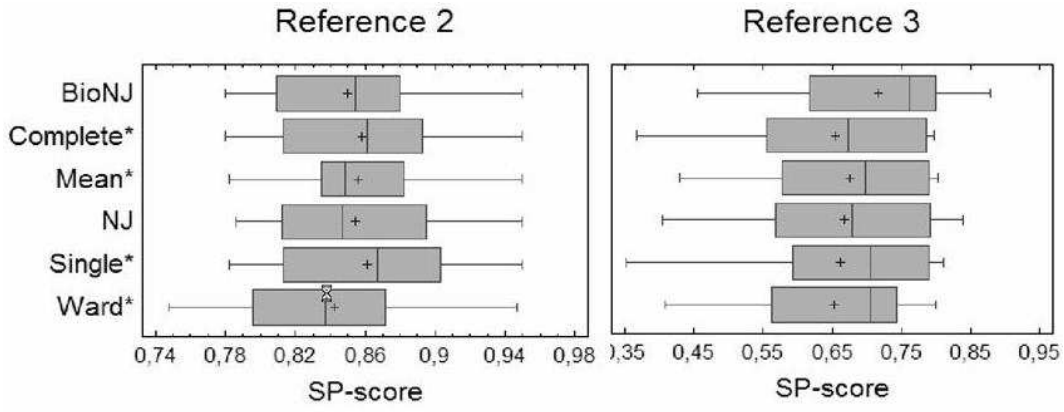


Fig. 4.6: Boxplot représentant les scores SP des références 2 et 3 de Balibase Etude préliminaire. Les méthodes de construction d'arbre ont été testés sur les références 2 et 3 de Balibase. Plus le score SP est élevé et plus l'alignement est de bonne qualité.

propres catégories en fonction des critères que nous souhaitons étudier (voir la tableau 4.4).

Propriété des séquences	Catégories
Longueur des séquences	34 à 199 acides aminés 200 à 678 acides aminés
Nombre de séquences	3 à 8 séquences 9 à 25 séquences
Hydrophobicité	26 à 38% 39 à 54%

Tab. 4.4: Tableau récapitulant les catégories que nous avons définies pour la base d'alignements de référence Sabmark

Nous avons choisi d'utiliser l'analyse en composantes principales (ACP), introduite en 1901 par K. Pearson et développée par H. Hotelling en 1933, qui est une méthode très puissante pour explorer une population statistique de taille n munie d'un nombre élevé p de données quantitatives observées. Son but est de trouver une représentation dans un espace réduit de k dimensions ($n << p$) qui conserve "le meilleur résumé " (au sens du maximum de la variance projetée). L'analyse des données observées doit tenir compte de leur caractère multidimensionnel et révéler les liaisons existantes entre leurs composantes. Le principe de l'ACP est d'obtenir une représentation du nuage des données dans un sous-espace de faible dimension par projection sur des axes orthogonaux (indépendants) qui ont la propriété d'extraire le maximum de la variance projetée (ou inertie projetée) des individus ou objets. Nous avons donc choisi de représenter nos $n = 12$ méthodes à l'aide d'une représentation en biplot : les axes étant les deux composantes principales détectées par cette méthode. Les données quantitatives observées sont les scores obtenus pour chaque alignement de référence testé. Il est à noter que pour chacun des biplots obtenus, l'information contenue par les deux composantes est supérieur à 95% de l'information totale. Ces représentations sont donc de bonne qualité.

Les représentations graphiques des ACPs réalisées pour chacune des catégories de Sabmark et nos propres catégories sont très proches les unes des autres. Les figures 4.7 page suivante, 4.8 page 144 et 4.9 page 144 sont quasi-similaires. Nous pouvons y observer que quatre méthodes (Neighbor-Joining, CAHc, CAHm et CAHs) se détachent des autres. Ces méthodes ont donc un comportement similaire. Afin d'observer plus en détail ce qui se passe dans le nuage composé des méthodes restantes, nous avons réalisé la même analyse sans ces 4 méthodes. Chacune des représentations est alors identique à la figure 4.10 page 145. Nous pouvons y voir que BioNJ et Weighbor se comportent de la même façon, ainsi que les méthodes issues du logiciel FastMe. Selon ces premiers résultats, nous pouvons classer l'ensemble des méthodes en trois groupes :

- Groupe 1 : Neighbor-Joining, CAHc, CAHm et CAHs
- Groupe 2.1 : BioNJ et Weighbor
- Groupe 2.2 : BMEb,n,o et GMEb,n,o

Pour la suite de l'interprétation des résultats, nous utiliserons les groupes de méthodes au lieu des noms de celles-ci.

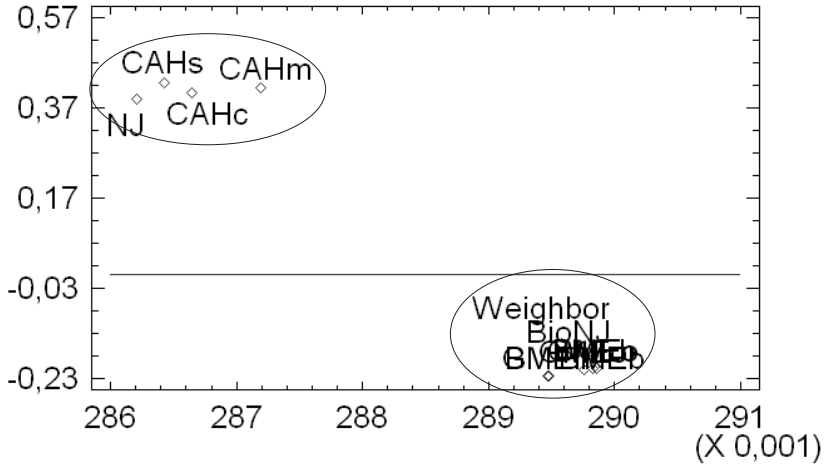


Fig. 4.7: Représentation graphique d'une ACP des scores obtenus par différentes méthodes de construction d'arbre – Analyse réalisée sur la totalité de SabMark

Ces premières observations nous indiquent uniquement que les comportements de ces méthodes sont proches mais pas d'informations sur la qualité des alignements obtenus. Sur la totalité de la base, ainsi que sur les deux catégories pré-définies, les résultats sont peu significatifs. Aucune information ne peut être mise en avant. En revanche, à l'aide des catégories que nous avons définies et de l'étude des scores (fournis dans l'annexe C page 193), nous obtenons des résultats exploitables. Afin d'avoir un aperçu simple de ces données dans ce chapitre, nous les avons résumés sur la figure 4.11 page 146.

Les méthodes du groupe 1 offrent les meilleurs résultats dans les catégories suivantes :
- Longueur des séquences de 34 à 199 acides aminés
- Nombre de séquences de 9 à 25 séquences
- Hydrophobicité de 39 à 54%

et au contraire les plus mauvais résultats dans les autres cas.

Sabmark est une base destinée à mettre en avant les difficultés des méthodes à aligner des séquences peu similaires. Les méthodes du groupe 1 offrent les meilleurs résultats quand un maximum d'informations est intrinsèque aux séquences : plus le nombre de séquences est important, plus il y a de résidus hydrophobes (ancres permettant d'aligner au mieux les séquences)

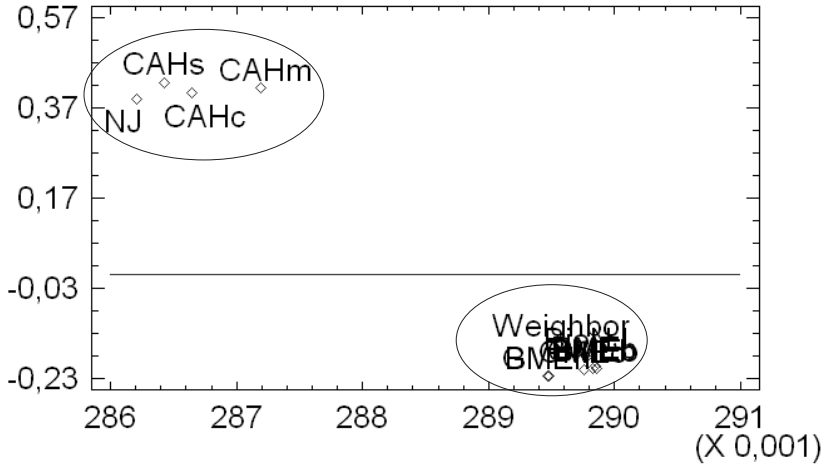


Fig. 4.8: Représentation graphique d'une ACP des scores obtenus par différentes méthodes de construction d'arbre – Analyse réalisée sur la catégorie Superfamille de SabMark

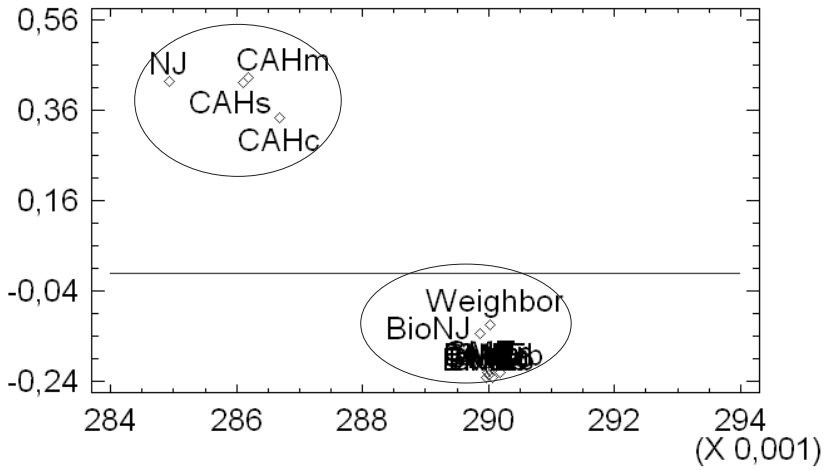


Fig. 4.9: Représentation graphique d'une ACP des scores obtenus par différentes méthodes de construction d'arbre – Analyse réalisée sur la catégorie Twilight zone de SabMark

ou encore plus les séquences sont courtes (moins de possibilité de mésaligner). C'est pourquoi, quand les alignements sont difficiles à réaliser du fait du peu de similarité et du peu d'information contenue dans les séquences, ces méthodes offrent des résultats de moins bonne qualité. Au contraire les méthodes du groupe 2 restent constantes et sont capables de générer de bons alignements même dans les cas difficiles (peu de similarité et peu d'information dans les données à analyser).

Balibase Cette étude est très proche de l'étude préliminaire réalisée. Sans connaissance a priori (ici la définition d'autres catégories et les résultats obtenus sur la base de données Sabmark), les résultats n'auraient pas pu être plus exploités. Balibase étant différente de Sabmark, les catégories que nous avons définies portent sur les mêmes critères mais sont découpées différemment (voir le tableau 4.5 page 147).

Les résultats obtenus ici sont les mêmes que ceux de l'étude précédente en y ajoutant ceux des méthodes Weighbor, BMEo et GMEo et leurs dérivées (BMEb,n et GMEb,n). Nous pouvons observer qu'aucune des méthodes n'est la meilleure pour toutes les références définies par Balibase. D'après les caractéristiques des bases décrites par leurs auteurs, on peut mettre en correspondance certaines références de celles-ci :

1. la référence 1_1 de Balibase, comprenant des séquences équi-distantes de moins de 20% de

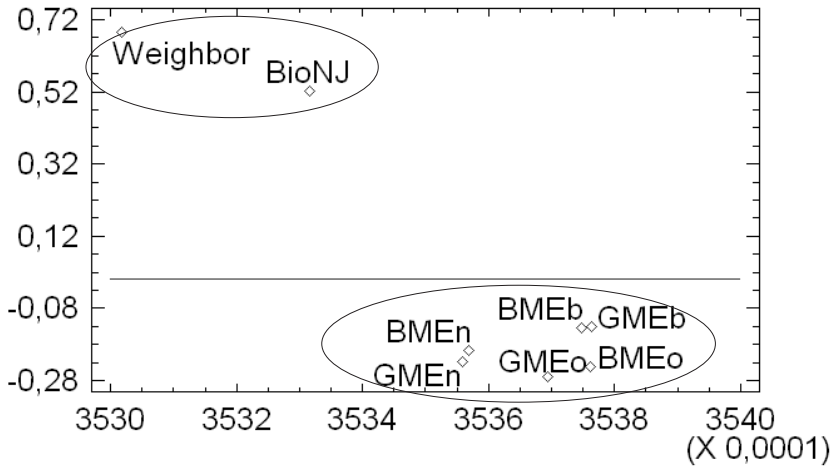


Fig. 4.10: Représentation graphique d'une ACP des scores obtenus par différentes méthodes de construction d'arbre (hormis le groupe 1) – Analyse réalisée sur la totalité de SabMark

similarité et la référence "Twilight zone" de Sabmark regroupant des séquences présentant moins de 25% de similarité

2. la référence 1_2 de Balibase, comprenant des séquences équi-distantes de moins de 40% de similarité et la référence "Superfamilles" de Sabmark regroupant des séquences présentant moins de 50% de similarité
3. et de façon plus lointaine, la référence 3 de balibase composée d'alignements de sous-groupes de séquences avec moins de 25% d'identité entre ces groupes et la référence "Twilight zone" de Sabmark

Les résultats obtenus par les méthodes testées sont les mêmes pour les deux bases d'alignements uniquement dans le cas de la première correspondance. Dans les deux autres cas, les résultats diffèrent complètement et aucune correspondance ne peut être faite.

L'analyse des ACPs sur les différentes catégories, sans connaissance a priori, est plus difficile de celle de Sabmark. Toutes les représentations graphiques (disponibles dans l'annexe C page 193) ne sont pas identiques. Nous avons donc choisi de repérer celles où l'on retrouve les groupes de méthodes identifiés précédemment :

– Toute la base

SUP	TWI	3 à 8 séqs	9 à 25 séqs	34 à 199 aa	200 à 678 aa	%hy : 26 à 38	%hy : 39 à 54	tout
NJ	NJ	BMEb	NJ	NJ	BioNJ	GMEo	NJ	NJ
BioNJ	CAHm	GMEb	CAHs	CAHs	Weigh	BMEo	CAHs	BMEo
BMEo	CAHs	BioNJ	CAHm	CAHm	GMEo	BMEb	CAHm	GMEo
Weigh	GMEb	NJ	BMEo	BMEo	BMEo	NJ	BioNJ	CAHm
GMEo	CAHc	BMEo	GMEo	GMEo	GMEb	GMEn	CAHc	BMEb
BMEb	BMEb	GMEo	CAHc	BMEb	BMEb	GMEb	GMEb	BioNJ
GMEb	BMEo	Weigh	BioNJ	GMEb	GMEn	BMEn	BMEb	GMEb
CAHm	GMEo	GMEn	BMEb	CAHc	BMEn	BioNJ	BMEo	CAHs
BMEn	GMEn	BMEn	GMEb	BioNJ	CAHm	Weigh	Weigh	Weigh
CAHs	BMEn	CAHm	BMEn	BMEn	CAHc	CAHm	GMEo	BMEn
GMEn	BioNJ	CAHs	Weigh	Weigh	NJ	CAHs	BMEn	GMEn
CAHc	Weigh	CAHc	GMEn	GMEn	CAHs	CAHc	GMEn	CAHc

Fig. 4.11: Classement des méthodes de construction d'arbre pour SabMark – Les méthodes sont classées en fonction de la moyenne de leurs scores par catégorie, la plus haute étant celle ayant le meilleur résultat. Les méthodes du groupe 1 sont colorées.

Propriété des séquences	Catégories
Longueur des séquences	0 à 300 acides aminés 300 à 600 acides aminés 600 à 1630 acides aminés
Nombre de séquences	4 à 19 séquences 20 à 59 séquences 60 à 142 séquences
Hydrophobicité	24 à 37% 38 à 45%

Tab. 4.5: Tableau récapitulant les catégories que nous avons définies pour la base d'alignements de référence Balibase

- Références pré-définies : 1_1, 1_2, 5 (et moins nettement 3)
- Longueur des séquences : 4 à 19 acides aminés (et moins nettement 20 à 59)
- Nombre de séquences : 0 à 300 séquences et 300 à 600 séquences
- Hydrophobicité : les deux catégories

A partir de ces observations, nous avons choisi d'analyser les résultats de cette étude en conservant les groupes de méthodes définis précédemment ainsi que l'hypothèse formulée. Nous souhaitons vérifier si le premier groupe, formé par la CAH avec trois différents critères et le Neighbor-Joining, fournit de bons résultats lorsque les conditions (nombre de séquences, longueur des séquences, présence de résidus hydrophobes) sont adéquates et des alignements de moins bonne qualité lorsque l'information contenue dans les séquences est plus difficile à mettre en évidence.

Les scores obtenus par les différentes méthodes (présentés dans l'annexe C page 193) nous permettent de conclure sur leurs performances en fonction des catégories que nous avons définies. Afin d'avoir un aperçu simple de ces données dans ce chapitre, nous les avons résumés sur les figures 4.12 page suivante et 4.13 page 149.

Concernant la longueur des séquences, les deux premières catégories définies pour Balibase ([0,300] et [300,600]) sont proches des catégories de Sabmark ([24,199] et [200,678]). On y observe le même phénomène. Lorsque la longueur des séquences est faible, le groupe 1 fournit les meilleurs résultats, tandis que ceux-ci sont moins bons lorsque les séquences comprennent un plus grand nombre d'acides aminés. On ne retrouve pas les deux groupes de méthodes dans la dernière catégorie de Balibase ([600,1630]). Celle-ci correspond à un cas critique pour les méthodes d'alignement de séquences. Plus les séquences sont longues et plus il est difficile d'obtenir un alignement de bonne qualité.

L'analyse du nombre de séquences par alignement est difficile à mettre en relation avec celle de Sabmark du fait de la différence de contenu des bases. On observe pour Balibase que le groupe 1 obtient les meilleurs résultats lorsque le nombre de séquences composant l'alignement

est inférieur à 60 (catégories [4,19] et [20,59]). Dès que ce nombre est plus important la qualité des alignements de ce groupe chute par rapport à la qualité de ceux de l'autre groupe.

Au niveau du pourcentage d'hydrophobicité, les différences de résultats du groupe 1 entre les deux catégories sont moins marquées que pour Sabmark. On peut cependant noter que les alignements obtenus avec la CAHc et le Neighbor-Joining sont plus précis lorsque le nombre de résidus hydrophobes est plus important.

Les conclusions que nous avons obtenues à partir de Sabmark sont consolidées par les résultats de cette analyse.

Conclusions Cette étude, réalisée sur deux bases d'alignements de référence et prenant en compte plus de critères sur les séquences à aligner, a pour but d'approfondir les résultats obtenus précédemment et de déterminer des modèles de décision en fonction des caractéristiques de

réf 11	réf 12	réf 2	réf 3	réf 4	réf 5	tout
NJ	CAHs	BioNJ	BMEn	CAHm	BMEn	CAHs
CAHm	GMEn	CAHs	GMEn	NJ	CAHm	CAHm
CAHs	BMEn	BMEn	GMEb	CAHs	BioNJ	BMEn
CAHc	GMEb	CAHm	Weighbor	BMEb	GMEo	BioNJ
Weighbor	BMEo	BMEo	BioNJ	BMEn	BMEo	BMEb
BMEn	BMEb	GMEo	GMEo	BioNJ	GMEb	NJ
GMEn	CAHc	Weighbor	BMEb	CAHc	BMEb	GMEn
BioNJ	GMEo	GMEn	BMEo	GMEo	Weighbor	Weighbor
BMEb	CAHm	GMEb	CAHs	BMEo	GMEn	GMEo
GMEb	BioNJ	BMEb	CAHm	Weighbor	CAHc	BMEo
GMEo	Weighbor	NJ	NJ	GMEb	CAHs	CAHc
BMEo	NJ	CAHc	CAHc	GMEn	NJ	GMEb

Fig. 4.12: Classement des méthodes de construction d'arbre pour Balibase (références pré-définies) – Les méthodes sont classées en fonction de la moyenne de leurs scores par catégorie, la plus haute étant celle ayant le meilleur résultat. Les méthodes du groupe 1 sont colorées.

séquences à aligner.

Les conclusions de cette étude sont nombreuses. Tout d'abord, nous avons pu classer en deux groupes principaux les méthodes étudiées. Le premier groupe est constitué de la CAH avec les différents critères (complet, moyenne, simple) et du Neighbor-Joining. Le second groupe est composé de BioNJ, Weighbor et des différentes approches proposées par le logiciel FastME (BMEb,n,o et GMEb,n,o).

Les méthodes du groupe 1 obtiennent de meilleurs résultats que les autres dans les cas énumérés ci-dessous :

- Nombre de séquences moyen (compris entre 9 et 60)
- Séquences de longueur moyenne (compris entre 0 et 300 acides aminés)
- Présence de résidus hydrophobes (plus de 39%)

Nous avons défini les intervalles entre parenthèses à l'aide des résultats des comparaisons précédentes. Ils pourraient être précisés par une étude approfondie de ces critères.

Par contre, dans les cas plus compliqués, la qualité des alignements obtenus par ce groupe de méthodes est moins bonne et les méthodes du second groupe sont meilleures. Celles-ci restent

%hy : 24 à 37	%hy : 38 à 45	0 à 300 aa	300 à 600 aa	600 à 1630 aa	4 à 19 séqs	20 à 59 séqs	60 à142 séqs
CAHs	BMEn	CAHs	BMEn	CAHm	CAHs	CAHs	BioNJ
BioNJ	CAHm	CAHm	BMEb	BioNJ	CAHc	CAHm	GMEb
CAHm	CAHs	NJ	BioNJ	CAHc	CAHm	BMEn	BMEn
Weighbor	BioNJ	BMEn	GMEn	BMEb	GMEn	BMEo	BMEb
BMEb	NJ	CAHc	CAHs	GMEb	BMEn	GMEo	Weighbor
GMEo	BMEb	BioNJ	GMEo	BMEn	GMEb	BioNJ	GMEo
BMEn	GMEn	GMEn	BMEo	NJ	GMEo	BMEb	BMEo
GMEn	GMEb	Weighbor	Weighbor	Weighbor	BMEo	GMEn	GMEn
BMEo	CAHc	GMEo	CAHm	GMEn	NJ	NJ	NJ
NJ	BMEo	BMEo	GMEb	CAHs	BMEb	Weighbor	CAHm
CAHc	GMEo	BMEb	NJ	GMEo	Weighbor	GMEb	CAHc
GMEb	Weighbor	GMEb	CAHc	BMEo	BioNJ	CAHc	CAHs

Fig. 4.13: Classement des méthodes de construction d'arbre pour Balibase (nos catégories) – Les méthodes sont classées en fonction de la moyenne de leurs scores par catégorie, la plus haute étant celle ayant le meilleur résultat. Les méthodes du groupe 1 sont colorées.

constantes et sont capables de générer de bons alignements même dans les cas difficiles (peu de similarité et peu d'information dans les données à analyser).

Le groupe 1 est constitué de la CAH avec 3 critères différents et du Neighbor-Joining. Bien que celui-ci soit une méthode de reconstruction phylogénétique, il obtient de moins bons résultats lorsque l'horloge moléculaire symbolisant l'évolution des séquences intervient que BioNJ et les autres méthodes basées sur le principe d'évolution minimum que nous avons utilisées. De ce point de vue et de part son algorithmique, cette méthode parait donc plus proche de la CAH, une méthode de classification. Cette hypothèse nous permet de justifier et de comprendre la formation des deux groupes de méthodes.

Ces études nous permettent donc de fournir des modèles de décision en fonction des propriétés des séquences. Lorsque les séquences à aligner possèdent des caractéristiques appartenant aux intervalles définis ci-dessus, les méthodes du premier groupe sont les plus appropriées ; le Neighbor-Joining et la CAHm sont d'ailleurs les méthodes utilisées classiquement. Dans tous les autres cas, nous conseillons d'utiliser une méthode du second groupe. Si aucune information n'est disponible sur la nature des séquences, une des méthodes de ce groupe apparait comme la plus appropriée, notamment BioNJ dont la dispersion des résultats est faible. Nous avons observé ce résultat sur le premier boxplot présenté 4.5 page 139, résultat qui a été confirmé par la suite. Cela signifie que BioNJ est la méthode offrant le moins de risque d'obtenir un alignement de mauvaise qualité. Une implémentation systématique de ces modèles de décision dans ClustalW est simple puisque ce logiciel utilise déjà les données de longueur de séquences, nombre de séquences et pourcentage d'hydrophobicité dans son système de score.

Cependant, les différences de performance des différentes méthodes observées grâce aux test de rang de Wilcoxon ne sont pas statistiquement significatives. Cette étape, importante pour l'algorithme d'alignement multiple de séquences progressif, a pour point de départ des données de faible qualité. En effet, on possède uniquement des scores d'alignement (souvent grossiers pour gagner en rapidité) pour chacune des paires de séquences. Par exemple, les méthodes de reconstruction phylogénétique construisent un arbre à partir d'un alignement multiple de séquences, ce qui n'est pas le cas ici. La qualité des données stockées dans la matrice de distance pour construire l'arbre de guidage sera toujours limitante pour cette étape dont nous avons montré l'importance. L'étape dite de "refinment" se justifie donc entièrement puisque la qualité des données de départ pour calculer l'arbre sera améliorée à chacune des itérations.

A l'aide des études réalisées, nous avons mis en avant l'influence de l'étape de calcul de la structure de guidage sur la précision des alignements de séquences et proposer des modèles de décision en fonction des séquences à aligner. Une perspective de ce travail est de tester ces mêmes méthodes sur d'autres algorithmes d'alignement progressif tels que Muscle, Mafft, Probcons... Il serait intéressant de vérifier si ces résultats sont généralisables ou dépendent de l'algorithme d'alignement complet. Nous allons à présent revenir sur notre méthode pyr2hier

et déterminer pourquoi une hiérarchie ordonnée n'apporte pas d'information complémentaire comme structure de guidage dans un algorithme d'alignement multiple progressif.

4.2.4 Discussion des résultats de pyr2hier

Afin de mesurer la pertinence de notre algorithme pyr2hier et de comprendre pourquoi une structure ordonnée n'apporte pas plus d'information dans l'étude précédente, nous avons réalisé un ensemble de simulations sur la distorsion entre la matrice de données initiale et la matrice correspondant à la hiérarchie ordonnée.

4.2.5 Simulation

Pour tester et valider notre approche, nous avons procédé à plusieurs simulations. Notre objectif est de tester le comportement de la méthode en fonction des paramètres suivants : la taille du jeu de données et le critère d'agrégation utilisé.

Afin de créer les jeux de données, nous avons utilisé un programme qui génère des classes de points aléatoires, de dimension quelconque. Pour cette simulation, différents ensembles de jeux de $n = 25, 50, 75, 100, 150$ données ont été considérés. Pour chaque ensemble, nous avons calculé 100 matrices de dissimilarité indépendantes, $d_i(n)$. Ces matrices sont composées de valeurs aléatoires obtenues avec l'algorithme suivant :

- Nous considérons une classe de taille N (taille de la matrice de dissimilarité)
- Nous choisissons 5 centres C aléatoires dans un plan
- Pour chacun des centres C, nous choisissons aléatoirement $N/5$ points dans un cercle dont C est le centre. Les points sont déterminés en considérant deux variables aléatoires : un angle α suivant une loi aléatoire $u_\alpha \sim U(0, 2\pi)$; un rayon δ suivant une loi aléatoire $u_\delta \sim U(0, \Delta)$ où Δ est un paramètre du programme.
- Ces points sont alors utilisés pour calculer la dissimilarité finale $d_i(n)$

L'algorithme pyr2hier, utilisé avec différents critères, a été appliqué sur la matrice de Robinson issue de la CAP (calculée avec le critère du maximum). Nous utiliserons alors la notation suivante : pyr2hier_*critère*. Pour chaque simulation nous mesurons les valeurs des critères D_1 et D_2 (définis à la section 1.3.3 page 41) pour estimer l'adéquation entre les différentes matrices. Les résultats numériques ayant permis de tracer les graphiques présentés dans cette section sont regroupés à l'annexe B page 185 et leurs représentations sont les figures 4.14 page suivante et 4.15 page suivante.

Les deux mesures nous apportent les mêmes informations. Nous pouvons voir que les deux courbes présentant les meilleurs résultats sont celles de la CAH avec les critères de la moyenne et de Ward. Les courbes des différents pyr2hier (excepté pyr2hier_min) sont très proches de la

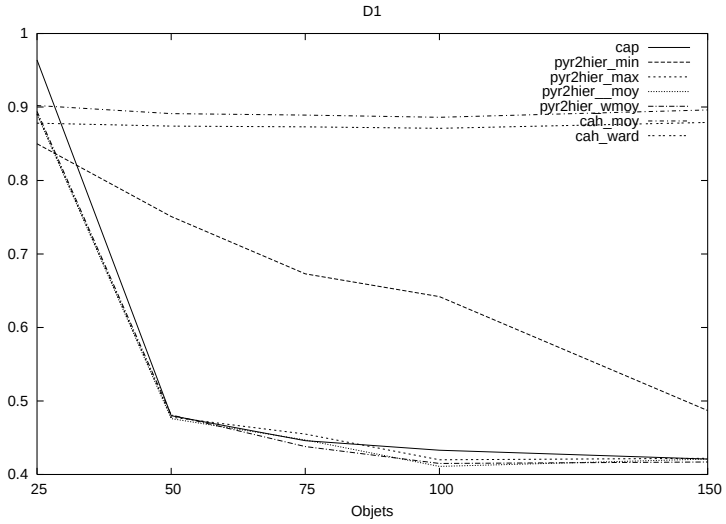


Fig. 4.14: Critère D1 pour chacune des méthodes testées par rapport aux données initiales – cap : classification ascendante pyramidale, pyr2hier_*critère* : pyr2hier avec différents critères (min = minimum, max = maximum, moy = moyenne, wmoy = moyenne pondérée), cah_*critère* : classification ascendante hiérarchique avec différents critères (moy = moyenne, ward = Ward)

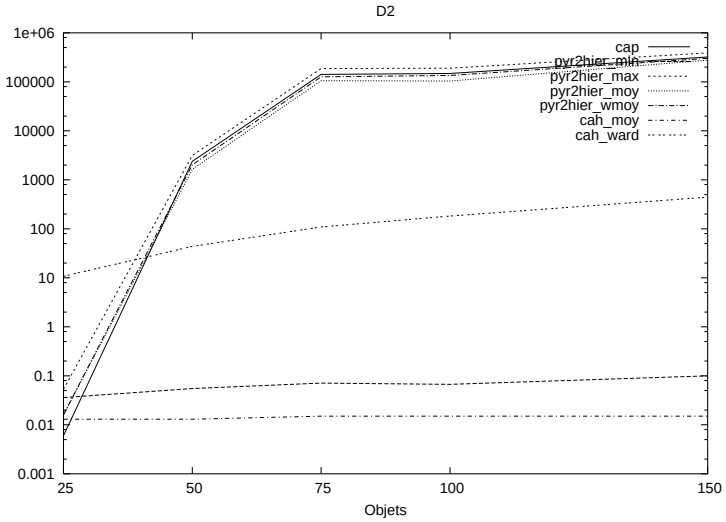


Fig. 4.15: Critère D2 pour chacune des méthodes testées par rapport aux données initiales – cap : classification ascendante pyramidale, pyr2hier_*critère* : pyr2hier avec différents critères (min = minimum, max = maximum, moy = moyenne, wmoy = moyenne pondérée), cah_*critère* : classification ascendante hiérarchique avec différents critères (moy = moyenne, ward = Ward)

CAP. Notre algorithme pyr2hier ne modifie que très peu l'information apportée par la CAP. Cependant les courbes montrent clairement que pyr2hier est moins proche des données initiales que la CAH. La représentation des données par Pyr2hier est donc plus mauvaise que celle de la CAH. Nous pouvons voir que la courbe de pyr2hier_min est meilleure que celle de pyr2hier calculé avec d'autres critères. Ce résultat est attendu car en appliquant pyr2hier avec le critère du minimum sur la matrice issue de la CAP avec le critère du maximum, les deux "effets" des critères sont moyennés.

Ces simulations nous permettent d'expliquer pourquoi notre méthode, bien qu'apportant la notion d'ordre, n'est pas meilleure que les autres méthodes de construction d'arbres que nous avons présentées. Les résultats obtenus montrent clairement que la représentation des données obtenue en appliquant pyr2hier n'est pas la plus proche des données initiales. Cette méthode étant une approximation, nous avons développé une approche utilisant directement la représentation pyramidale comme structure de guidage.

4.3 Alignement multiple progressif, alliant stratégies globale et locale

Précédemment, nous avons mis en évidence que l'étape de calcul de la structure de guidage avait une importance certaine pour l'approche progressive. Nous avons présenté un algorithme permettant d'obtenir une hiérarchie ordonnée. Cela nous permet d'exploiter la propriété d'ordre de la classification pyramidale. Cependant, cette méthode n'est qu'approximative et l'information contenue dans la pyramide n'est pas entièrement utilisée. C'est pourquoi nous proposons ici une stratégie mixte, alliant alignements global et local, avec la classification pyramidale.

Dans le chapitre précédent, nous avons opposé deux approches d'alignements de séquences : une globale, visant à aligner les séquences sur la totalité de leur longueur et une locale, cherchant à déterminer des fragments de séquences similaires. Ces deux stratégies, comme le montrent les études comparatives réalisées par : [McClure et al., 1994, Briffeuil et al., 1998, Thompson et al., 1999b, Karplus and Hu, 2001, Lassmann and Sonnhammer, 2002], sont adaptées à différents types de "cas biologiques". Nous avons choisi de les illustrer à l'aide de comparaisons sur la base d'alignements de référence Balibase (voir 3.3.2 page 114).

Sur les figures 4.16 page suivante et 4.17 page suivante, les deux algorithmes de référence ClustalW (global) et DiAlign (local) sont comparés aux approches mixtes T-Coffee [Notredame, 2002] et DC-Mixed [Sammeth and Morgenstern, 2003]. T-Coffee est un algorithme permettant d'utiliser des données hétérogènes pour calculer un alignement de séquences ; dans cette implémentation les données choisies sont des alignements par paire issus de deux programmes d'alignement : un global (ClustalW) et un local (Lalign). Les alignements sont de meilleure qualité grâce à

l'utilisation d'un grand nombre de données mais ceci au détriment de la rapidité de calcul. DC-Mixed combine une approche "divide-and-conquer" globale et une approche "segment à segment" locale comme DiAlign. Les segments locaux sont calculés puis sont mis bout à bout. L'intégration de ces deux techniques est coûteuse en temps.

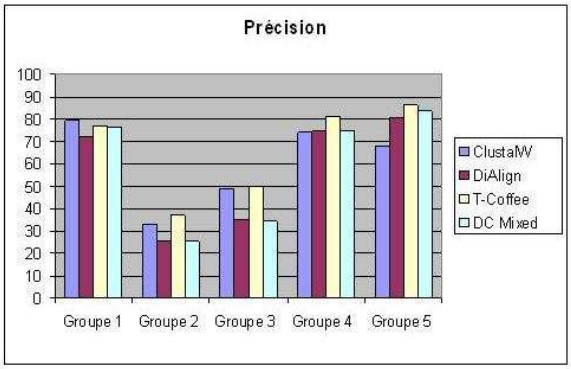


Fig. 4.16: Comparaison de 4 méthodes sur la base Balibase – Précision, mesurée avec le score CS.

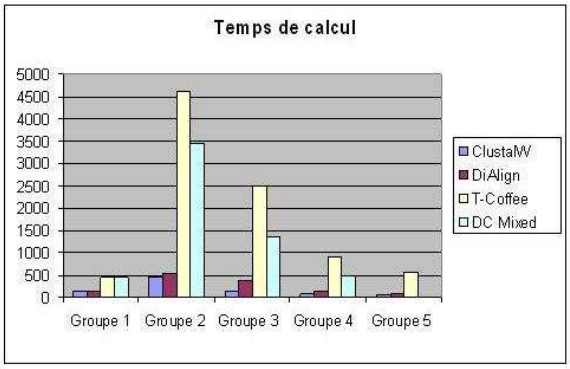


Fig. 4.17: Comparaison de 4 méthodes sur la base Balibase.2 – Temps de calcul, en secondes

En terme de précision, les méthodes mixtes apportent de bons résultats dans tous les cas alors que les stratégies simples ne sont performantes que dans certains (ClustalW pour les groupes de 1 à 3 et DiAlign pour les groupes 4 et 5). En revanche, les approches mixtes présentées sur cet exemple sont fort coûteuses en temps. De plus, le nombre de séquences alignées dans ces jeux de test est faible, par rapport au grand nombre de séquences disponibles et utilisé dans les analyses réelles de séquences. L'intérêt d'insérer une stratégie mixte dans une approche progressive est donc d'autant plus flagrant. Il est en effet important de pouvoir cumuler une bonne précision

et un temps de calcul relativement faible.

La nouvelle méthode que nous proposons introduit des modifications au niveau des étapes 2 et 3 de l'approche progressive décrite dans le chapitre précédent. Dans l'étape 2, la modification est simple. La structure de guidage, généralement calculée à l'aide de l'algorithme du Neighbor-Joining ou de l'UPGMA, est remplacée par une pyramide calculée à l'aide de l'algorithme de la CAP. L'objectif de cette approche est d'utiliser la propriété de recouvrement des pyramides afin de sélectionner la meilleure méthode d'alignement (local ou global) à l'étape 3.

Les pyramides font appaître des recouvrements emboîtés (*i.e.* deux classes ne sont pas nécessairement disjointes). Une séquence peut donc appartenir à une seule classe (séquence intra-classe, par exemple, sur la figure 4.18, A et B,C) ou à deux classes (séquence inter-classe, par exemple : D possède un motif propre à la famille A,B,C,D et un motif propre à la famille D,E,F). Les séquences intra-classes sont plus proches et donc plus adaptées à un alignement global. Les séquences inter-classes font le lien entre deux groupes préalablement alignés. Ces deux ensembles de séquences sont plus éloignés et donc à aligner localement. Ainsi, en tenant compte des caractéristiques biologiques des séquences mises en évidence par la pyramide, nous obtenons un ordre et un type d'alignement des séquences.

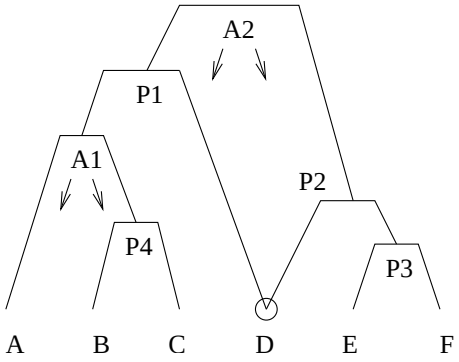


Fig. 4.18: Exemple de pyramide pouvant être utilisée comme structure de guidage – A1 et A2 sont deux paliers choisis afin d'illustrer le type d'alignement possible

4.3.1 Algorithme d'alignement multiple mixte : pyrAlign

Notations et définitions

Soit (P, f) une pyramide indicée où $P = \{p_i | i \in [1, |P|]\}$. La pyramide est représentée par l'ensemble des paliers p_i la composant.

Nous notons :
– $L(p_i)$ (*resp.* $R(p_i)$) un opérateur renvoyant le successeur gauche (*resp.* droit) d'un palier

p_i de P ;
- $C(P) = \{p_i | p_i \in P, f(p_i) \leq f(p_{i+1}), \forall i \in [1, |P| - 1]\}$ l'ensemble des paliers ordonnés par indice croissant ;
- $\overline{p_i}$ les séquences ou l'alignement associés au palier p_i.
- $\overline{L(p_i)|I(p_i)}$ les séquences ou l'alignement associés au successeur gauche du palier p_i, moins les séquences ou l'alignement associés à $I(p_i)$
- $\tilde{p_i}$ la séquence consensus de $\overline{p_i}$

Nous définissons deux opérateurs *global*() et *local*() permettant de représenter le type d'alignements effectués. L'opérateur *global*() est utilisé pour l'alignement de séquences proches et sera donc appliqué dans le cas de séquences intra-classes. Au contraire, les séquences inter-classes, permettant de faire le lien entre 2 groupes préalablement alignés, seront alignées à l'aide de l'opérateur *local*(). Ces règles d'alignement sont formalisées ci-dessous lors de leur utilisation dans l'algorithme. Ces deux opérateurs peuvent aligner différentes entités : une séquence *vs* une séquence ; une séquence *vs* un groupe de séquences alignées ou encore un groupe de séquences alignées *vs* un groupe de séquences alignées. Nous verrons que l'opérateur *local*() n'est pas forcément un opérateur binaire.

Algorithme

L'algorithme que nous proposons est un algorithme progressif, alignant les séquences ou groupes de séquences depuis les feuilles jusqu'à la racine de la pyramide (P, f). Les paliers sont parcourus par ordre d'indice croissant, du palier le plus homogène au palier le moins homogène grâce à $C(P)$. Ainsi, tous les paliers sont parcourus et à chaque étape, les paliers précédents, plus homogènes, ont été traités. A la fin de cet algorithme, on obtient un alignement multiple de toutes les séquences.

L'algorithme est le suivant :

1. **Initialisation :** La structure de guidage contenant l'ordre et le type d'alignement des séquences (représentées par les singletons ou feuilles de la structure) est la pyramide notée (P, f). L'ensemble $C(P)$ associé à la pyramide est calculé. On fixee $i = 0$ et $p_i = 0 \in C(P)$
2. **Alignement :**
   (a) Pour chaque palier p_i, on considère ses 2 successeurs $L(p_i)$ et $R(p_i)$. Ces 2 successeurs sont : soit une séquence dans le cas où $L(p_i)$ et/ou $R(p_i)$ correspondent à une feuille ; soit un alignement déjà formé dans le cas où $L(p_i)$ et/ou $R(p_i)$ correspondent à un palier.
   (b) L'alignement des 2 successeurs est fait grâce aux opérateurs *global*() ou *local*(). Les règles d'alignement sont les suivantes, $\forall p_i \in P | f(p_i) \leq f(p_{i+1}), i \in [i, || - 1]$:
   – Si $L(p_i) \cap R(p_i) = \emptyset$ alors $\overline{p_i} = global(\overline{L(p_i)}, \overline{R(p_i)})$

– Sinon $I(p_i) = L(p_i) \cap R(p_i)$ et $\overline{p_i} = local(\overline{L(p_i)|I(p_i)}, \overline{I(p_i)}, \overline{R(p_i)|I(p_i)})$
On obtient un nouvel alignement pour le palier p_i.

3. **Test d'arrêt :** L'algorithme prend fin quand le palier p_i que l'on vient d'aligner est égal à Ω. Dans le cas inverse on reprend l'exécution de l'algorithme à partir de la phase d'alignement, avec le palier $_{i+1}$

En reprenant ces notations utilisées pour les règles d'alignement, sur la figure 4.18 page 155, on obtient :

– $L(A1) = A$ et $R(A1) = P4 = \{B, C\}$ L'intersection de ces deux ensembles est nulle et la règle d'alignement sera *global*(A, P4).
– $L(A2) = P1 = A, B, C, D$ et $R(A2) = P2 = D, E, F$ L'intersection des ces deux ensembles est D et la règle d'alignement sera *local*(P1|D,D,P2|D) = *local*(A1,D,P3).

Pour illustrer l'intérêt de développer une telle méthode, nous avons choisi l'exemple simple utilisé dans [Thompson et al., 1994]. Pour cela, nous avons pris la matrice issue des comparaisons de paires de séquences comme matrice de dissimilarité initiale pour calculer la pyramide. Sur la figure 4.19, on observe la structure de guidage pour déterminer notre alignement multiple (figure 4.20 page suivante). L'ordre et le type de méthode d'alignement sont répertoriés dans le tableau 4.6 page suivante. Les paliers sont parcourus du plus homogène au moins homogène. L'étape 4 a été mise en avant pour représenter un cas d'alignement local avec la séquence HBB_HUMAN comme intersection de deux groupes de séquences à aligner.

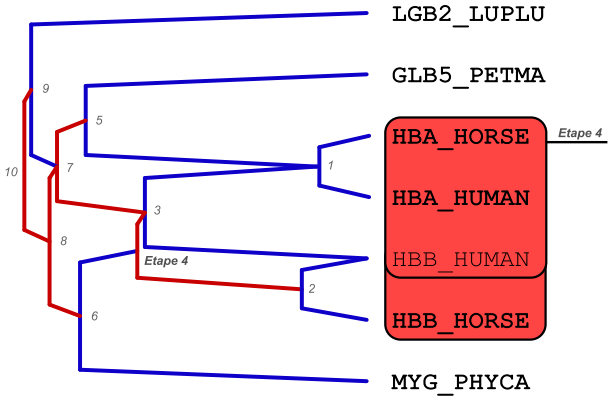


Fig. 4.19: Pyramide obtenue à partir de la matrice des comparaisons par paires de séquences. – En bleu, sont représentés les paliers menant à un alignement global, en rouge, les alignements menant à un alignement local. L'étape 4 a été mise en avant pour représenter un cas d'alignement local avec la séquence HBB_HUMAN comme intersection de deux groupes de séquences à aligner.

L'alignement obtenu manuellement, en utilisant DIALIGN pour les alignement locaux et ClustalW pour les alignements globaux, est comparable en tout point à celui de ClustalW comme le montre la présence des 7 hélices α. DIALIGN réalise des alignements locaux de deux entités. Pour aligner les deux groupes de séquences (moins les séquences à l'intersection de

Etapes	L(C)	R(C)	Type d'alignement
1	HBA_HUMAN	HBA_HORSE	global
2	HBB_HORSE	HBB_HUMAN	global
3	HBB_HUMAN	1	global
4	2	3	local
5	1	GLB5_PETMA	global
6	MYG_PHYCA	4	global
7	3	5	local
8	6	7	local
9	7	LGB2_LUPLU	global
10	8	9	local

Tab. 4.6: Tableau récapitulant l'ordre d'alignement (étapes, dont les numéros sont utilisés pour nommer les groupes de séquences), les groupes à aligner et le type d'alignement à effectuer

ces groupes) et le groupe formé par les séquences à l'intersection, nous avons effectué deux étapes d'alignement. D'abord, nous avons choisi d'aligner arbitrairement un des deux groupes de séquences avec l'intersection des deux groupes afin d'obtenir un alignement intermédiaire. Puis, nous avons réutilisé DIALIGN pour aligner le deuxième groupe de séquences avec cet alignement intermédiaire.

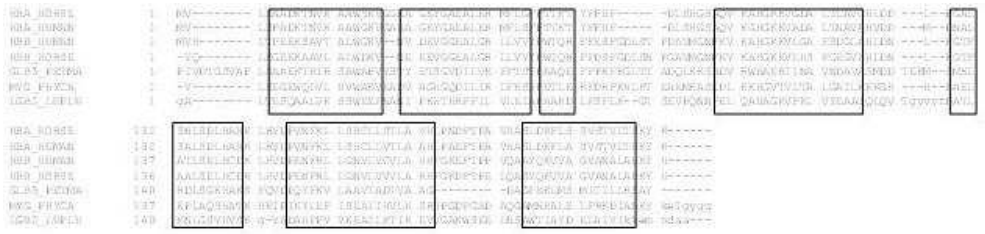


Fig. 4.20: Alignement multiple obtenu par notre méthode mixte. Les 7 hélices α recherchées sont encadrées

Cette approche générale ouvre de nombreuses possibilités, notamment dans les façons de définir les opérateurs *global*() et *local*() évoquées ci-dessus.

Une des premières considérations est la façon de représenter les groupes d'alignement. Ceux-ci peuvent être conservés tels quel, être représentés sous forme de profils, de séquences consensus... Pour chacun de ces cas, il existe un grand nombre de façons de calculer la façon de représenter un alignement multiple. Nous ne les détaillerons pas ici. Dans le cas de l'alignement local, considérant $\overline{L}$,$\overline{R}$ et $\overline{I}$, on peut appliquer l'algorithme de Smith & Waterman adapté pour un triplet de séquences, comme dans [Loytynoja and Milinkovitch, 2003].

Dans le cas de la fonction *local*(), on peut également se poser plusieurs questions entraînant de nombreuses perspectives à ce travail. Le fait de traiter deux ensembles de séquences à aligner (ou un ensemble de séquences et une seule séquence) et leur intersection soulève de nombreuses problématiques.

La première est de savoir comment traiter l'intersection ; par exemple en fonction de sa taille. Si l'intersection entre deux groupes de séquences est grande, les séquences sont proches et peuvent alors être alignées globalement plutôt que localement. Il serait donc intéressant d'affiner l'approche en déterminant un seuil permettant de tester la taille de l'intersection et d'appliquer en fonction de celle-ci soit l'opérateur *global*(), soit l'opérateur *local*().

Sur la figure 4.21 page suivante (a), est représentée l'intersection de deux alignements : ($\tilde{L}$,I) et (I,$\tilde{R}$). Un seuil α est défini pour représenter l'intersection de ces deux groupes. Si le recouvrement est faible, un alignement local est adapté (b). Réciproquement, plus le recouvrement entre les alignements est grand et plus un alignement global sera adapté. Sur la figure 4.21 page suivante , les deux cas sont illustrés au niveau de la pyramide. Si les paliers sont très proches les uns des autres en terme de niveau d'homogénéité, l'alignement local ne se justifie pas (c).

Une seconde perspective est d'utiliser l'opérateur *local*() de façon binaire, directement sur les deux groupes à aligner, et non plus de façon tertiaire, sur ces deux groupes (moins les séquences à l'intersection) et les séquences à l'intersection ; c'est à dire $local(\overline{L(p_i)|I(p_i)}, \overline{I(p_i)}, \overline{R(p_i)|I(p_i)})$. On aurait alors l'alignement local des deux groupes : $\overline{p_i} = local(\overline{L(p_i)}, \overline{R(p_i)})$. Dans ce cas, les séquences $\overline{I(p_i)}$ à l'intersection des deux groupes sont présentes dans chacun des groupes à aligner et ont donc un poids double lors de l'étape d'alignement. Cela se justifie car ces séquences ont des caractéristiques les faisant appartenir à deux classes. Il faudra cependant définir la façon de les représenter à l'issue de l'étape d'alignement : les séquences faisant le lien entre les deux classes étant présentes deux fois. Dans la pratique, nous avons observé que les deux instances d'une même séquence à la fin d'un alignement étaient identiques ; mais cela ne saurait être une règle.

Nous avons exposé un algorithme d'alignement multiple de séquences reposant sur une approche progressive mixte, ainsi que des perspectives d'amélioration. Dans la partie suivante, nous reprenons l'exemple de la famille de protéines multi-domaines chez *S.cerevisiae* et montrons l'efficacité de notre stratégie.

4.3.2 Application de cette méthode

Afin de démontrer l'intérêt de notre méthode, nous l'avons appliquée sur un cas réel d'une famille de protéines multi-domaines chez *S.cerevisiae*, exemple déjà utilisé précédemment dans ce manuscrit. Cette famille est composée de 11 protéines caractérisées par la présence de 3 domaines : un domaine "endonucléase MG^{2+}-dépendante" (représenté en vert), un domaine "Leucin-rich repeats" (LRR, représenté en bleu) et un domaine "Protéine phosphatase 2C" (représenté en jaune). Ces informations sont regroupées dans le tableau 2.1 page 50.

Afin de démontrer l'intérêt d'une approche mixte par rapport aux méthodes simples globales ou locales, nous comparons les alignements obtenus par notre méthode, ClustalW et DiAlign. Sur la figure 4.23 page ci-contre, sont représentés en (a), l'alignement multiple obtenu à partir de ClustalW et en (b), celui obtenu avec DiAlign. L'efficacité des méthodes est testée en vérifiant si celles-ci prédisent au mieux la réalité biologique. C'est pourquoi seuls les domaines caractéristiques sont représentés (un domaine correspond à une couleur donnée). Avant de décrire les résultats, nous expliquons la méthodologie nous ayant permis de calculer un alignement multiple à partir de la pyramide représentée sur la figure 4.22 page suivante.

Conformément à la méthodologie décrite ci-dessus, nous définissons un ordre d'alignement et un type d'alignement en parcourant par ordre croissant les paliers de la structure de guidage. Nous obtenons alors le tableau 4.7 page 165. Comme pour l'exemple précédent, nous utilisons

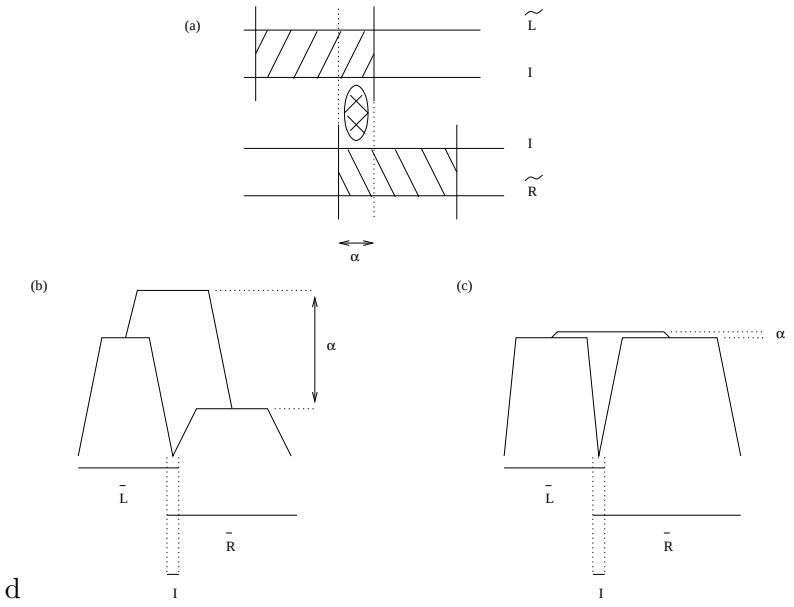


Fig. 4.21: Définition d'un seuil pour la taille de l'intersection – (a) Le seuil α peut être visualisé comme l'intersection des alignements de ($\tilde{L}$,I) et de (I,$\tilde{R}$). (b) Cas de pyramide où un alignement local est justifié. (c) Cas de pyramide où un alignement global est justifié

ClustalW pour les alignements globaux et DiAlign pour les alignement locaux. Ces logiciels permettent à l'utilisateur d'utiliser en entrée plusieurs séquences ; des alignements et des séquences ou encore plusieurs alignements.

Sur l'alignement obtenu par ClustalW, les trois types de domaine se chevauchent, c'est à dire qu'ils sont alignés les uns avec les autres. Ils n'ont pas été correctement détectés par cette

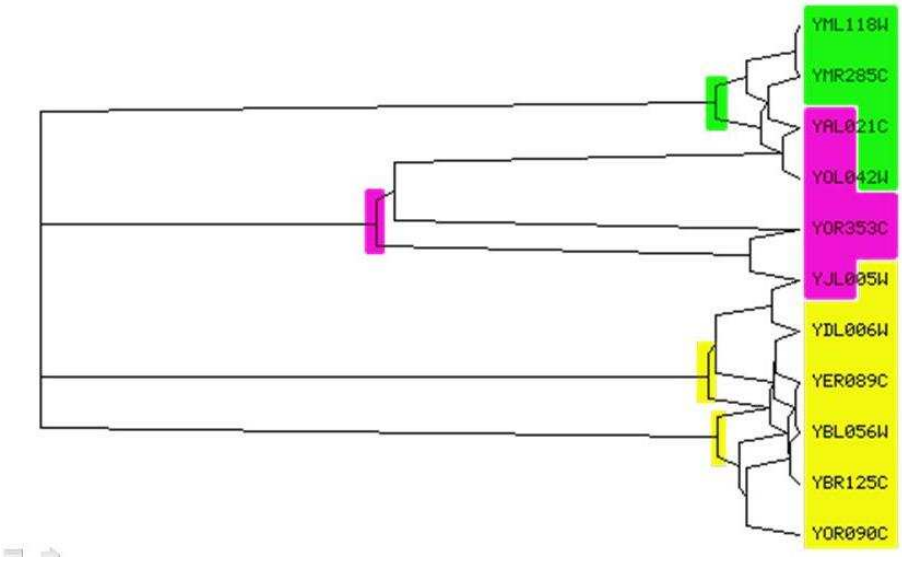
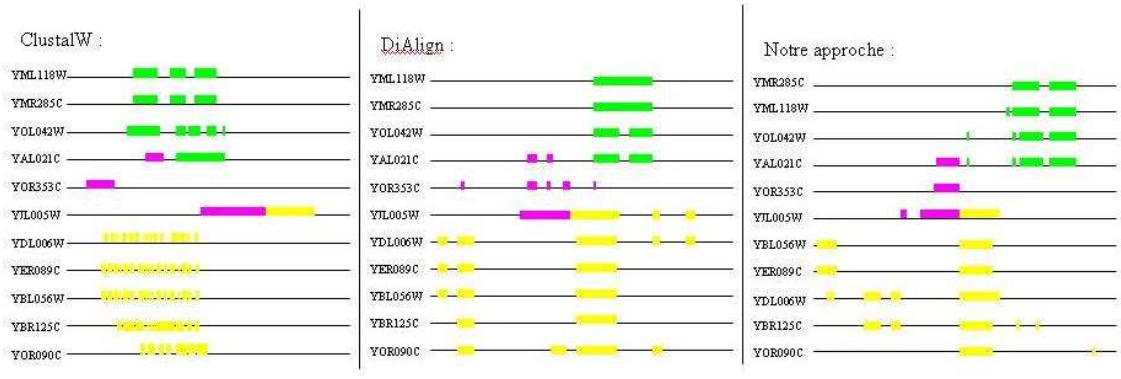


Fig. 4.22: Pyramide représentant une famille de protéines homologues.



Remarque : Les insertions de « gap » ont été conservées dans les domaines, pour représenter le cas d'une étude réelle

Fig. 4.23: Alignement multiple d'une famille de protéines multi-domaines, obtenu par notre méthode. (a) Alignement global par ClustalW, (b) Alignement local par DiALign, (c) Alignement mixte par notre méthode. En vert, le domaine Ccr4P ; en rose, le domaine LRR ; en jaune, le domaine PP2C

méthode. Cet alignement conduit à une grande fragmentation des domaines, de nombreuses brèches étant insérées pour pouvoir les aligner. Le domaine PP2C est particulièrement découpé et donc inidentifiable en tant que domaine. Sans connaissance a priori sur les séquences, il est donc impossible de prédire la présence de trois domaines caractéristiques parmi les 11 séquences.

L'alignement calculé par DiAlign est de meilleure qualité : les domaines sont moins superposés et moins fragmentés que dans celui de ClustalW. Par exemple, les domaines LRR des séquences YAL021C, YOR053C et YJL005W sont alignés alors qu'ils ne l'étaient pas du tout en utilisant une approche globale. Cependant, les domaines PP2C et CCR4P sont superposés et donc difficilement identifiables sans connaissance préalable.

Notre approche mixte permet de conserver les domaines comme de simples blocs mais est moins efficace à déterminer leurs frontières (par exemple, le coté gauche du domaine CCR4P). Certains domaines sont fragmentés. Cependant, il est à noter que les extrémités des domaines sont floues et dépendent de façon importante de l'algorithme utilisé pour les définir. Par exemple, le domaine PP2C de la protéine YBR125C est définie comme un simple bloc dans la base de données PFAM [Sonnhammer et al., 1997] et comme trois blocs dans la base de données Panther [Mi et al., 2005]. Avec notre méthode, les domaines sont clairement visibles et sans superposition.

Les méthodes ClustalW et DiAlign ne représentent pas au mieux la réalite biologique et donc fonctionnelle de cette famille de protéines. Les nombreuses "difficultés" présentes rendent l'alignement difficile : famille multi-domaines, séquences de longueurs très différentes, répétitions d'un domaine dans une séquence (YJL005W et ses nombreux domaines LRR)... Cet exemple illustre les difficultés propres aux méthodes simples globale et locale. L'alignement obtenu à partir de notre méthode est celui de meilleure qualité.

4.4 Conclusion

Dans cette partie du travail, nous avons décidé d'appliquer la classification pyramidale à l'alignement multiple de séquences, et plus particulièrement à l'approche progressive qui utilise une structure de guidage pour déterminer l'ordre des séquences à aligner. Cette méthodologie repose sur un algorithme composé de trois étapes distinctes :

1. Alignement de toutes les séquences par paires
2. Construction d'une structure de guidage
3. Calcul de l'alignement multiple final par agrégation successive des séquences selon la structure de guidage.

Les améliorations présentées dans la littérature portent principalement sur les étapes (1) et (3). Nous avons donc entrepris d'étudier plus particulièrement l'étape 2 de construction de la

structure de guidage.

Dans un premier temps, nous avons développé un algorithme permettant de transformer une pyramide en hiérachie ordonnée. Ainsi, l'étude de l'influence de l'ordre des données est possible lors de la comparaison des différentes méthodes de construction d'arbre. Pour cela, nous avons proposé un algorithme agglomératif simple et performant, respectant la contrainte suivante : conserver l'ordre des individus.

Ensuite, nous avons réalisé une étude de l'influence de la structure de guidage sur la précision d'un alignement multiple de séquences. Nous avons sélectionné un ensemble de méthodes de construction d'arbre auquel nous avons ajouté la version hiérarchisée des pyramides afin d'exploiter leur propriété d'ordre induit sur les données. Nous avons comparé ces différentes méthodes de construction d'arbre. Pour cela, nous avons utilisé une version modifiée du programme CLUSTALW pour évaluer chaque méthode avec des banques d'alignements de référence. Nous avons pu montrer le rôle majeur joué par cette étape, ainsi : (i) La méthode utilisée par défaut pour le calcul de la structure de guidage, le 'Neighbor Joining', dans la majorité des algorithmes progressifs donne des résultats non optimaux ; (ii) Le meilleur algorithme dépend de la nature des séquences initiales. (iii) L'ordre sur les données apporté par la version hiérarchisée d'une pyramide n'améliore pas la qualité des alignements. Nous avons ainsi défini deux groupes de méthodes, parmi les méthodes de construction d'arbre :

- les méthodes de classification (CAH) et le Neighbor-Joining (qui, contrairement aux méthodes suivantes, donne de moins bons résultats lorsque l'horloge moléculaire intervient)
- les méthodes de reconstruction phylogénétiques (BioNJ, Weighbor, FastME)

Les méthodes du premier groupe obtiennent de meilleurs résultats que les autres quand un maximum d'informations est intrinsèque aux séquences, c'est à dire dans les cas énumérés ci-dessous :

- Nombre de séquences moyen (compris entre 9 et 60)
- Séquences de longueur moyenne (compris entre 0 et 300 acides aminés)
- Présence de résidus hydrophobes (plus de 39%)

C'est pourquoi, quand les alignements sont durs à réaliser du fait du peu de similarité et du peu d'information contenue dans les séquences, ces méthodes offrent des résultats de qualité plus faible. Au contraire, la précision des méthodes du second groupe reste constante et celles-ci sont capables de générer de bons alignements même dans les cas difficiles. Si aucune information a priori n'est disponible sur la nature des séquences, une des méthodes de ce groupe apparait comme la plus appropriée, notamment BioNJ dont la dispersion des résultats est faible. Cette étude nous a donc permis de montrer l'importance de cette étape et le besoin d'utiliser une méthode adaptée aux données traitées.

Cependant, la hiérarchisation de la pyramide étant une approximation, nous avons ensuite proposé une approche mixte, basée sur les stratégies d'alignement local et global. Pour cela nous avons développé un nouvel algorithme permettant d'utiliser la propriété de recouvrement des pyramides : un objet (ici une séquence) peut appartenir à deux classes. L'utilisation des séquences appartenant à deux classes nous permet de faire le lien entre deux classes préalablement existantes et de traiter ces classes de façons distinctes en s'appuyant sur la réalité biologique. Les séquences intra-classes sont plus proches et donc plus adaptées à un alignement global. Les séquences inter-classes font le lien entre deux groupes préalablement alignés. Ces deux ensembles de séquences sont plus éloignés et donc à aligner localement. Ainsi, en tenant compte des caractéristiques biologiques des séquences mises en évidence par la pyramide, nous obtenons un ordre et un type de méthode d'alignement des séquences. Nous ne modifions donc plus uniquement l'étape de calcul de la structure de guidage de l'approche progressive mais également l'étape finale d'alignement multiple de cette approche. Cette méthode est très prometteuse comme nous avons pu le voir en l'appliquant sur des exemples concrets et offre de nombreuses perspectives.

Etapes	L(C)	R(C)	Type d'alignement
1	YMR185C	YML118W	global
2	YOL042W	YAL021C	global
3	YAL021C	YMR285C	global
4	3	1	local
5	2	3	local
6	5	6	local
7	YJL005W	YOR353C	global
8	YOR353C	2	global
9	7	8	local
10	YBL056W	YER089C	global
11	YBR125C	YBL056W	global
12	YER089C	YDL006W	global
13	YDL006W	YJL005W	global
14	11	12	local
15	10	12	local
16	14	15	local
17	YOR090C	11	global
18	17	14	local
19	18	16	local
20	15	13	local
21	16	20	local
22	9	6	local
23	19	21	local
24	23	22	local

Tab. 4.7: Tableau récapitulant l'ordre d'alignement (étapes, dont les numéros sont utilisés pour nommer les groupes de séquences), les groupes à aligner et le type d'alignement à effectuer

Conclusion & Perspectives

Devant l'importance du volume de données biologiques générés, les mathématiques et l'informatique sont devenues deux disciplines incontournables pour stocker, traiter, comparer, ces données. Avec le besoin de (re)trouver l'information parmi les données disponibles, les méthodes pour traiter et analyser de façon systématique les connaissances se développent pour extraire, classer, estimer et prédire. Nous nous sommes intéressés à la découverte d'information sans connaissance a priori et plus particulièrement au développement et l'amélioration d'une méthode de classification automatique. Ce type de méthodes a pour objectif principal de (re)trouver les similitudes présentes dans les données et de les représenter.

La première partie de ce travail concerne l'interprétation et la représentation des données biologiques. Dans un premier temps, nous avons présenté une sélection de méthodes permettant de construire un arbre à partir d'une matrice de distance. Ces méthodes sont utilisées en biologie pour traiter différents problèmes comme l'analyse de séquences d'ADN, d'ARN ou encore protéiques, l'analyse de puces à ADN... Une des caractéristiques principales de ces méthodes est la capacité à traiter un grand volume de données. Afin que ces techniques soient utilisées, il faut qu'elles soient rapides, simples et efficaces. Pour cela, elles doivent avoir une algorithmique simple au point de vue informatique tout en représentant au mieux les données. Les algorithmes agglomératifs comme la classification ascendante hiérarchique et le Neighbor-Joining (et ses variantes) répondent en effet à ce critère. Il en est de même pour les méthodes basées sur le principe d'évolution minimum.

Le sujet principal de cette thèse est la classification pyramidale : une méthode de classification généralisant les hiérarchies. Les méthodes de classification jouent un rôle proéminent dans les domaines d'application traitant d'importants volumes de données. Elle est particulièrement adaptée à la complexité des données biologiques. En effet, ses propriétés supplémentaires (représentation des classes empiétantes et ordre partiel sur les données), par rapport à la classification hiérarchique, permettent d'obtenir des représentations plus proches des données initiales.

La Classification Ascendante Pyramidale est l'algorithme le plus fréquemment utilisé pour calculer une pyramide. Cependant, un biais de construction important était induit par cet algorihtme : la présence de paliers inversés, due à une matrice de dissimilarité induite non robin-

sonienne. Nous proposons deux approches pour corriger ce biais afin d'obtenir une dissimilarité induite de Robinson, en accord avec l'extension du théorème de Johnson-Benzecri. Nous avons proposé une solution optimale avec un filtrage global, réalisé par régression isotone, et une approche heuristique, avec le filtrage local. Nous avons montré que l'utilisation de telle ou telle méthode dépend du volume des données de départ, du résultat attendu et des moyens informatiques à disposition. Nous avons également présenté un algorithme en deux étapes permettant de reconstruire une pyramide à partir de la matrice de Robinson obtenue par filtrage. Cette méthode repose sur la recherche de cliques maximales dans un graphe particulier, les cliques étant les paliers de la pyramide. Nous disposons donc à présent d'une méthode de classification fiable et robuste, offrant de nombreux avantages : représentation de classes empiétantes et ordre partiel des individus. Ses domaines d'application en biologie sont donc à étendre, par exemple dans l'analyse du transcriptome.

Dans la seconde partie de ce travail, nous avons décidé de l'appliquer à l'alignement multiple de séquences, et plus particulièrement à l'approche progressive qui utilise une structure de guidage pour déterminer l'ordre des séquences à aligner. L'alignement de séquences est une méthode permettant de prédire des informations de structure (secondaire ou tertiaire), de déterminer si une nouvelle séquence appartient à une famille de protéines ou non. L'alignement multiple de séquences est également une étape préliminaire à un grand nombre d'analyses de séquences, d'analyses phylogénétiques. L'approche progressive, offrant un bon compromis entre temps de calcul et précision, utilise une structure de guidage pour déterminer l'ordre des séquences à aligner. La première étape (*i.e.* l'alignement de toutes les paires de séquences) et la troisième (*i.e.* l'alignement progressif des séquences en suivant les branches de l'arbre) ont été très étudiées. L'ordre d'alignement des séquences est très important. Cet ordre est déterminé par la structure de guidage, et dépend donc de la technique choisie pour calculer celle-ci.

Nous avons donc utilisé l'ordre induit par la classification pyramidale pour fixer l'ordre d'alignement des séquences. Pour cela, nous avons développé un algorithme hiérarchisant une pyramide, l'ordre étant la principale contrainte à conserver. L'étude de l'influence de la structure de guidage, avec différentes méthodes, nous a permis de montrer l'importance de cette étape et le besoin d'utiliser une méthode adaptée aux données traitées. En effet, aucune des méthodes testées n'est la meilleure dans tous les cas. Certains types de méthodes sont plus adaptées dans des cas donnés. C'est pourquoi nous avons pu définir des groupes de méthodes adaptés aux séquences biologiques à analyser. Les méthodes de "classification" offrent les meilleurs résultats quand un maximum d'informations leur est donné à la base de l'analyse : plus le nombre de séquences est important, plus les séquences sont courtes ou encore plus il y a de résidus hydrophobe. Au contraire les "méthodes de reconstruction phylogénétique" restent constantes et sont capables de générer de bons alignements même dans les cas difficiles (peu de similarité et peu d'information dans les données à analyser). Si aucune information n'est disponible sur la nature des séquences, nous conseillons l'utilisation d'une des méthodes de ce groupe ; notamment

BioNJ, qui apparait comme la plus appropriée. La hiérarchisation de la pyramide étant une approximation, nous avons ensuite proposé un algorithme permettant d'utiliser la totalité de l'information contenue. En effet, non seulement, l'utilisation des séquences appartenant à deux classes, nous permet de faire le lien entre deux classes préalablement existantes, mais aussi on peut traiter ces classes de façons distinctes en s'appuyant sur la réalité biologique. Cette approche est très prometteuse comme nous avons pu le voir en l'appliquant. La classification pyramidale, de par ses propriétés, offre la possibilité de représenter au mieux la complexité des données étudiées tout en étant simple à appliquer et à analyser.

Ce travail peut être le point de départ de nombreuses perspectives :

- **d'un point de vue algorithmique**, les méthodes de filtrage pour la consolidation de la pyramide peuvent être affinées, par exemple en utilisant l'algorithme exact SIBC au lieu de l'algorithme SMOOTH qui converge vers la solution optimale. De même pour l'applicaton de la classification pyramidale dans un alignement multiple de séquences, le retour d'expérience de biologistes utilisant la méthode proposée permettra d'adapter la méthode aux données traitées. Une seconde perspective de ce travail est de tester les méthodes de construction d'arbre sur d'autres algorithmes d'alignement progressif tels que Muscle, Mafft, Probcons... Il serait intéressant de vérifier si ces résultats sont généralisables ou dépendent de l'algorithme d'alignement complet. ;
- **d'un point de vue méthodologique**, le traitement de données complexes, hétérogènes et volumineuse. Une perspective serait de développer des méthodes robustes à partir de la classification pyramidale qui s'appliquent à tous types de données. Il faudrait aussi travailler sur la compréhension et la lisibilité des connaissances découvertes, afin de mettre en avant la nouveauté et la complémentarité des ces connaissances par rapport à celle de l'expert humain.

Annexe A

DaTool

Cette annexe a pour but de présenter l'outil dans lequel nous avons développé les approches de classification que nous avons présentées dans ce manuscrit.

DaTool (Distance Analysis Tool) est une plateforme d'analyse de données biologiques définies par des distances. Notre objectif est d'intégrer, en plus des méthodes classiques, de nouvelles méthodes comme la classification pyramidale. Nous souhaitons aussi apporter un soin tout particulier à la validation des résultats ainsi qu'à leur représentation. Ainsi, les utilisateurs auront un outil complet, interactif et convivial. Un "cahier des charges" a été défini pour cerner les problèmes rencontrés et pour mettre en évidence les attentes des biologistes. Le domaine d'application d'un tel outil est vaste : comparaison de génomes microbiens, analyse comparative de protéome, Teraprot, analyse du transcriptome... Cet outil est en cours de développement. Actuellement, notre projet est composé de 31k lignes de code (équivalent 7,33 hommes/année calculé avec le logiciel SLOCCount selon le "basic cocomo model").

La figure A.1 page suivante nous présente les modules de DaTool : les librairies et les programmes utilisant ces librairies. On peut distinguer :

- un module de conversion, daConvert, gérant les différents formats de fichier
- un module de calcul, daCalc, dont les principaux algorithmes ont été détaillés dans ce manuscrit
- un module de dessin, qui sera détaillé dans cette annexe

Le projet est entièrement écrit en C. Ce langage apparaissait comme le plus proche de nos besoins. En effet, le C permet une programmation efficace.

L'utilisation des algorithmes de calcul décrits dans ce manuscrit, va produire beaucoup d'informations sur nos données de départ. Les résultats vont alors être stockés dans un fichier au format XML. Ce fichier va nous permettre de manipuler une grande diversité de type de données. Dans un premier temps nous décrirons le cadre (langages et outils) que nous avons

défini pour ce projet puis les structures de données adaptées que nous avons choisies.

Comme nous l'avons vu, ce projet est actuellement constitué de plusieurs modules, notamment de daCalc. Ces différents aspects, très détaillés dans ce manuscrit, ne seront pas repris ici. Ce logiciel devant permettre aux biologistes de traiter leurs données avec des méthodes robustes et fiables et d'interprêter directement les résultats, la partie graphique a donc, à terme, une place prépondrante dans ce projet. La réussite de cette étape est critique car elle est destinée à faciliter ce travail d'analyse et d'interprétation des biologistes. Nous décrirons donc ici le module de dessin de notre plateforme.

A.1 Langages et outils

A.1.1 Les outils de gestion de projet

Pour qu'un projet soit interactif et pérenne, plusieurs outils de gestion sont nécessaires.

Tout d'abord, pour que plusieurs utilisateurs travaillent en même temps sur un projet, un système de gestion de version est impératif. Afin de gérer les différents modules, de faire le lien entre ceux-ci et de contrôler les différentes versions, nous avons donc utilisé un serveur CVS

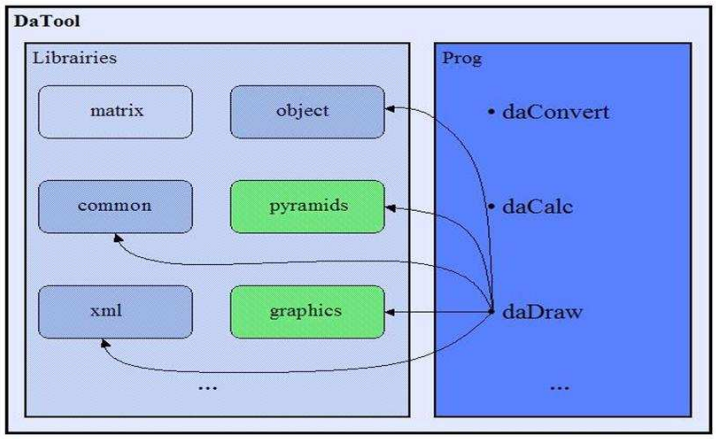


Fig. A.1: L'architecture du projet DaTool – Interactions des programmes et des librairies

(Concurrent Versions System, http ://www.cvshome.org).

Ensuite, deux points sont essentiels à la pérennité d'un projet et sa reprise par d'autres personnes :

- Les commentaires d'un code ; souvent soit le code n'est pas du tout commenté, soit il l'est trop. Dans les deux cas le résultat est le même : le code est illisible.
- La documentation détaillée d'un projet.

Nous avons choisi un outil puissant capable de générer la documentation d'un projet (format HTML ou Latex) : Doxygen (www.doxygen.org). Son fonctionnement est similaire celui de l'API Doc de java. En conséquence, le code a été commenté de telle manière qu'il soit utilisable par ce programme. La documentation générée par Doxygen est très utile pour comprendre et pérenniser le projet mais aussi pour trouver n'importe quel type de renseignement sur une fonction bien précise, de façon très rapide.

A.1.2 Les fichiers de données du projet

Les fichiers de données du projet sont stockés au format XML, en utilisant la DTD dédiée au projet : DAML. La norme XML (eXtensible Markup Language) permet avant tout de stocker dans un fichier des informations structurées. On parle alors de document XML. Ce dernier est composé de texte libre et de balises possèdant éventuellement des attributs.

A l'origine, dans les années 60, avant même la création d'Unix, on pouvait remarquer SGML parmi les langages à balises. Dans les années 80, SGML devient stable, et Internet commence se développer. En l'occurrence, dans les années 90, un sous-ensemble de SGML commence à apparaître pour créer ce qui va devenir le Word Wide Web, et se nomme HTML. Un groupe de travail du W3C (Word Wide Web Consortium www.W3C.org) reprend l'usage courant du HTML en 1994, et crée en 1996 une RFC (Request For Comments : ensemble de documents contenant des spécifications techniques) qui définit le HTML 2.0. Parallèlement à cela, en novembre 1996, un premier brouillon de XML est écrit, en tant que sous-ensemble de SGML. XML devient une recommandation du W3C en février 1998. XSLT (Extensible Style Language Transformations : langage destin transformer un fichier XML en un autre fichier XML ou HTML) date de 1999, puis on commence à penser à rapprocher HTML et XML. Le 26 janvier 2000, le W3C publie une nouvelle recommandation, XHTML-1.0, qui nest pas la réunion du XML et du HTML, mais un sous-ensemble de XML. Actuellement, XML en est toujours sa version 1.0.

DAML

Le XML définit la syntaxe d'un document, la DTD sa sémantique. La DTD est caractérisée par un ensemble de règles. Celles-ci permettent de spécifier les éléments et leurs attributs, ainsi

que leurs relations. En XML il n'existe aucune balise prédfinie, c'est à nous de les définir en mettant l'accent sur la signification des données. Le modèle que l'on a défini est présenté par la figure A.2.

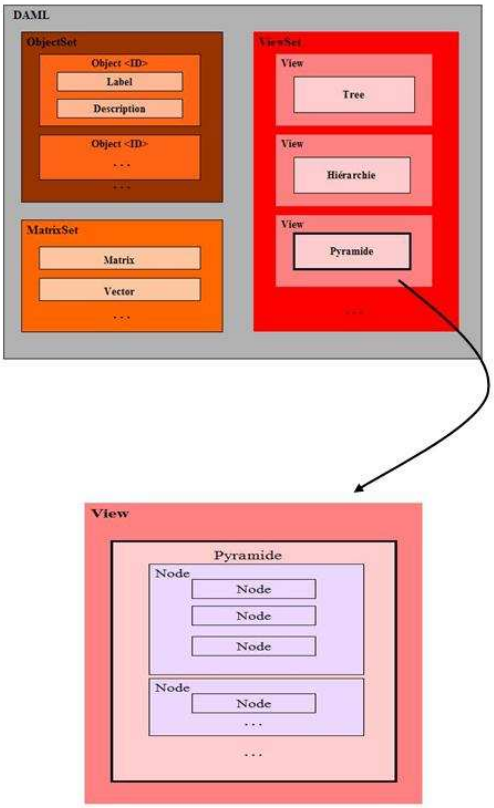


Fig. A.2: Définition de notre format DAML

Nous pouvons voir sur la figure A.2 que le document DAML est composé de trois sous-ensembles différents : ObjectSet, MatrixSet et ViewSet. Le sous-ensemble ObjectSet, est l'entité permettant de représenter une collection d'objets. Chaque objet est décrit par un identifiant unique (ID), une étiquette (label) et optionnellement une description complète (description). Le second sous-ensemble, MatrixSet, est composé de matrices triangulaires ou rectangulaires et de vecteurs utilisés au cours des calculs. C'est notamment dans ce sous-ensemble qu'est enregistrée la matrice de dissimilarité initiale sous forme d'une matrice triangulaire. Enfin, le sous-ensemble ViewSet est utilisé pour décrire les différentes représentations associées aux méthodes de classifications. Le principal intérêt du format DAML est qu'il permet l'encapsulation au sein d'un même fichier de plusieurs représentations associées à un jeu de données.

Je vais maintenant vous décrire la partie correspondant à la description des pyramides, dont voici l'extrait de la DTD. ELEMENT va définir les balises du document XML et ATTLIST en détermine les attributs. L'extrait ci-dessous nous montre que la balise ¡pyramid¿ prendre comme

attribut obligatoire (required) un identifiant unique (ID). Ainsi une pyramide est composée d'un premier noeud (pyramid.node) correspondant à la racine de la pyramide.

```
<!-- pyramid description -->
<!ELEMENT pyramid         (pyramid.node)>
<!ELEMENT pyramid.node    (\%node_desc; ,
                           ((pyramid.node.ref*, pyramid.node*) | object.ref))>
<!ELEMENT pyramid.node.ref EMPTY>

<!ATTLIST pyramid          id   ID    #REQUIRED>
<!ATTLIST pyramid.node     id   ID    #REQUIRED>
<!ATTLIST pyramid.node.ref ref  IDREF #REQUIRED>
```

Chaque noeud (identifié aussi de manière unique par un ID) est composé : d'un ensemble de noeuds ou de références vers des noeuds (décrit ci-après) s'il s'agit d'un noeud interne du graphe et d'une référence vers un objet s'il s'agit d'un noeud terminal (ou feuille) de l'arbre. Les références sont utilisées pour créer des indirections (liens). En effet, si un objet apparaît dans plusieurs représentations il est inutile de le dupliquer. Un lien vers sa déclaration dans le sous-ensemble ObjectSet assure une meilleure intégrité et supprime la redondance. Ci-dessous, un exemple de référence vers un objet X dans une pyramide.

```
<pyramid.node ID=TEST_X>
<value type=level>1.000000</value>
<object.ref REF=OBJ_X/>
</pyramid.node>
```

Les références vers les noeuds internes d'une pyramide sont utilisées dans le cas particulier où un noeud a deux prédcesseurs, comme nous le montre le code ci-dessous représentant .

```
<ViewSet>
<pyramid ID="TEST">
<pyramid.node ID="A">
        <pyramid.node ID="B">
                <pyramid.node ID="TEST_D">
                </pyramid.node>
                <pyramid.node ID="TEST_E">
                </pyramid.node>
        </pyramid.node>
        <pyramid.node ID="C">
```

```
                <pyramid.node.ref REF="E"/>
                <pyramid.node ID="F">
                </pyramid.node>
        </pyramid.node>
</pyramid.node>
</pyramid>
</ViewSet>
```

Le parseur XML

Pour lire un document XML, il existe deux méthodes assez répandues et qui répondent chacune à différents besoins : Sax et Dom.

Sax dispose d'un système de callbacks, qui sont des pointeurs de fonction. Le principe est de lire balise après balise. Suivant le type de balise, le parseur rend la main via un callback pour exécuter les opérations voulues. Cette méthode donne un grand contrôle sur la lecture du document XML et son interprétation. Ainsi un tel parseur ne prend pas beaucoup de place en mémoire contrairement à un parseur basé sur Dom. La mémoire n'est allouée qu'en fonction des besoins de l'utilisateur.

Un parseur Sax est donc intéressant pour lire des documents XML volumineux dont on souhaite extraire une partie. Il est également pratique lorsque toutes les données doivent être analysées mais ne nécessitent pas d'être en même temps en mémoire.

Un parseur **Dom** va construire en mémoire un arbre représentant le document XML. Toutes les opérations vont donc être réalisées directement sur la représentation mémoire. Dom possède donc deux avantages : l'accès aux données est rapide, car elles ont été organisées en mémoire pour être d'accès facile ; les données peuvent être relues plusieurs fois rapidement car le chargement est initial et n'est donc pas refait à chaque fois.

Pour le projet DaTool, la méthode choisie est Sax. En effet, nous n'avions pas besoin d'une représentation en mémoire de nos fichiers. De plus, ceux-ci sont volumineux (matrice, hiérarchie, pyramide...) et toutes les informations contenues ne vont pas être utilisées au même moment.

Fonctionnement du parseur

Cette étape doit permettre d'obtenir la représentation mémoire (décrite dans la section suivante) de la pyramide en lisant le fichier. Un parseur de type Sax lit les balises une à une. Le fonctionnement du parseur est donc assez simple et dépend du type de balise rencontrée, comme nous allons le voir ci-dessous.

Quand le parseur rencontre une balise ouvrante, il y a : création et allocation de la structure concernée (par exemple, création et allocation d'une structure "pyramid" pour une balise ¡pyramid¿) ; récupération des attributs (s'il y en a) pour remplir les membres de la structure.

Quand le parseur rencontre une balise fermante (caractérisée par un / avant le nom de la balise), il y a : inclusion de la structure dans celle la contenant (par exemple, à la balise ¡/object¿, on assigne un noeud l'objet lui correspondant).

Les noeuds sont un cas particulier : pour gérer les relations d'inclusion entre noeuds (relations qui représentent les liens prédécesseurs/successeurs), on utilise une pile (LIFO1). En plus des actions générales précédemment énoncées : lors d'une balise ouvrante on empile la structure ; lors d'une balise fermante on la dépile.

Exemple :

```
Code
Pile
Action
<noeud  A>
A
on empile A
       <noeud  B>
A -> B
on empile B
       </noeud  B>
A
on dépile B
       <noeud  C>
A -> C
on empile C
       </noeud  C>
A
on dépile C
</noeud  A>

on dépile A
```

Les balises "références" sont utilisées par les noeuds et les objets pour gérer le chevauchement. Afin de ne pas créer deux structures identiques, on utilise une table de hachage avec comme clé le nom du noeud et comme élément son adresse mémoire. Ainsi les relations nécessaires pourront être établies. Le parsing terminé, nous n'avons plus besoin du fichier XML : une

représentation mémoire peut contenir une (des) matrices, le graphe planaire représentant une pyramide, une hiérarchie... C'est à partir de cette représentation que les différentes opérations (calcul, dessin...) sont réalisées.

A.2 Implémentation

Tout d'abord, je vais exposer la structure de données qui concerne la représentation des pyramides, puis l'architecture du programme. Pour finir, des résultats d'applications seront présentés.

A.2.1 Structure de données représentant une pyramide

Cette partie est importante car il a fallu définir une structure efficace pour manipuler les informations nécessaires au dessin de la pyramide. Les critères qui ont participé à sa définition sont la rapidité lors du parcours du graphe et la faible empreinte mémoire. Une pyramide est un graphe planaire dont chaque noeud est défini par la structure C suivante :

```
struct pyrnode {
  char         *id;              /*!< node identifier*/
  double      level;    /*!< index of a class */
  pPyrNode  pred_left;  /*!< left predecessor, pointer on a pyrnode */
  pPyrNode  pred_right; /*!< right predecessor , pointer on a pyrnode*/
  pListX2     succ_list;        /*!< successors double linked-list */
  int            min;   /*!< minimal bound of this node */
  int            max;   /*!< maximal bound of this node */
  pObject    object;    /*!< a pointer on an object */
  pListX2    *extra;    /*!< additional data such as style ... */
};
```

Cette structure permet de définir pour un palier : un identifiant unique ; son indice pyramidal ; les relations avec ses prédcesseurs et ses successeurs ; ses bornes sur l'ensemble ordonné ; le cas échant, un pointeur vers un objet pour les noeuds terminaux. Par ailleurs une liste chaîne (extra) permet d'ajouter des informations à un noeud. Dans le graphe, il y a plusieurs types de noeuds qui se différencient par leur niveau dans la pyramide :

- Les noeuds terminaux. Ils contiennent les objets. Ces derniers n'ont pas de successeur, uniquement des prédcesseurs.
- Le noeud racine (root) : il regroupe l'ensemble des objets. Il n'a que des successeurs et aucun prédcesseur.

– Les noeuds internes : ce sont les paliers de la pyramide. Ils ont des successeurs et des prédcesseurs.

La figure suivante prsente les différents types de noeuds ainsi que les relations de prédcesseurs et successeurs.

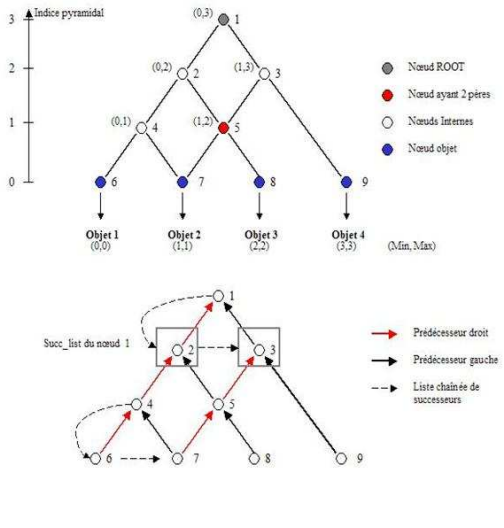


Fig. A.3: Représentation des éléments d'une pyramide

Comme on peut voir sur la figure A.3, les successeurs d'un noeud sont gérés sous la forme d'une liste chaînée. Cette liste est très simple à utiliser et permet d'obtenir tous les successeurs d'un noeud avec une complexité linéaire. En effet, la liste est constituée d'un ensemble de cellules chaînes entre elles. Il y a autant de cellules que d'éléments de la liste. Une cellule est constituée d'informations sur l'élément et d'un champ suivant (respectivement précédent) qui contient l'adresse de la cellule suivante (respectivement précédente). Le champ suivant de la dernière (respectivement première) cellule est NULL. C'est l'adresse de la première de ces cellules qui détermine la liste. Nous utilisons cette structure de données pour stocker les successeurs d'un noeud s'il en a. Cette liste contient un ou plusieurs éléments qui comportent chacun une structure noeud. Nous pouvons voir sur la figure ci-dessus que la liste chaîne du noeud 4 contient le noeud 6 ainsi que le noeud 7. De même la liste chaîne du noeud 1 contient le noeud 2 et le noeud 3.

A.2.2 Parcours d'un graphe

Pour dessiner la pyramide, on a besoin de parcourir toute la structure. Le parcours de la pyramide est un parcours récursif en profondeur : le dessin se fait à partir des noeuds de plus

bas niveau vers les niveaux supérieurs et se termine par le noeud racine (ROOT). Par exemple sur la figure A.3 page précédente, l'ordre de dessin des noeuds sera 6, 7, 4, 8, 5, 2, 9, 3 et 1. Un noeud n'est dessiné qu'une fois que tous ses successeurs sont dessinés. Le dessin de la pyramide sera fait par l'appel de la fonction parcours, dont l'algorithme en pseudo-langage est donné ci-dessous, avec le noeud racine (ROOT) comme paramètre.

```
Soit N, un noeud.
Fonction de parcours~:
parcours (N) {
        SI N a des successeurs {
        FAIRE {
        SI (prdcesseur droit = N ET prdcesseur gauche != NULL)
           ALORS ne rien faire
        SINON SI (prdcesseur droit = N)
           ALORS parcours (successeur de N)
        SINON SI (prdcesseur gauche = N)
           ALORS parcours (successeur de N)
       } TANT QUE( N a des successeurs)
       } action (N)
}
```

Les conditions mêmes de la fonction de parcours sont nécessaires pour ne pas traiter plusieurs fois un noeud.

A.3 daDraw

Le projet DaTool, dédié à l'analyse de données biologiques, comprend un module graphique daDraw permettant de visualiser ces structures. Un tel programme existait déjà, mais il a dû être entièrement réécrit pour pouvoir utiliser un nouveau type de format de description des pyramides (utilisant XML), gérer de nouveaux formats de sortie de fichier et le rendre plus fonctionnel. En amont, DaTool génère le fichier XML à partir duquel daDraw va créer une image. Ce programme se situe en fin de processus et permet de dessiner la pyramide en prenant comme fichier d'entrée la pyramide décrite au format XML.

A.3.1 La couche d'abstraction graphique

Nous avons voulu que les pyramides puissent être dessinées dans de nombreux formats bitmaps ou vectoriels. Une approche simpliste serait d'écrire une fonction par format, ce qui

introduirait une très forte redondance et nécessiterait d'importants efforts de maintenance. Une manière plus pertinente pour s'affranchir de la dépendance avec les différents formats de fichiers est d'introduire une couche d'abstraction graphique. Le rôle de celle-ci est de fournir des primitives graphiques (dessiner une droite, un texte, ...) de haut niveau, sous la forme d'une API3, l'algorithme de dessin. Un effet de bord positif de cette approche est que cette même API pourra être utilisée pour dessiner d'autres structures de classification (arbre, hiérarchie, ...). A ce stade, un prototype a été implémenté réalisant une figure au format PNG.

Les formats d'échanges de données entre programmes doivent être uniformisés. Un effort particulier a été mené pour bien séparer le code correspondant à la partie dessin (indépendante du format de sortie) du driver (réalisant les opérations de génération du graphique). Notre but est de fournir des programmes modulaires et sous forme de bibliothèques afin d'en faciliter la gestion et l'utilisation. Dadraw permet la représentation d'une hiréarchie ou pyramide sous la forme d'un graphe planaire. En ce qui concerne les pyramides on a les propriétés suivantes :

- La figure est dessinée verticalement.
- Le dessin peut être réalisé, au choix, en mode vectoriel ou bien en mode point (bitmap).
- Un objet est inclus dans au plus deux paliers.
- Chaque côté d'un palier d'une pyramide est représenté de manière oblique et orienté vers l'intérieur du palier. Pour les hiérarchies il s'agit d'une représentation horizontale.
- Les inclusions de sous-paliers intermédiaires (en sus des successeurs gauches et droits) sont représentées par des droites horizontales.
- Les paliers inversés sont indiqués en rouge (pour tester les méthodes de filtrage).
- Le programme fonctionne en ligne de commande.
- Les formats de sorties et les tailles maximales des figures sont : PNG [1024*768] PW3 [800*2400] (en fait du PNG) FIG [9540*12150] PS [9540*12150] , EPS [9540*12150] (La conversion du format FIG vers PS/EPS est ralisée par le programme Transfig distribué avec Xfig)

Le dessin est réalisé par une fonction générique qui utilise des "drivers" pour gérer chaque format de sortie.

De nombreuses variables et options graphiques ont été ajoutées, telles que : la définition d'une marge, la gestion des polices de taille proportionnelles, un cadre autour des labels, des titres/sous-titres, une échelle (pour indiquer la plage de variation de l'indice de hauteur de palier), les entêtes et pieds de pages (utile pour les formats pleine page),l'ajout d'un cadre autour de la figure, l'affichage tronqué des labels si ceux-ci sont trop grands... D'autres fonctionnalités comme l'affichage de vignettes représentatives d'une figure ou un zoom pour l'affichage dans un poster par exemple ont également été imaginées. Toutes ces fonctionnalités proviennent de besoins mis par les biologistes qui utilisent ce type de programmes.

L'implémentation de la couche d'abstraction graphique est indiquée sur la figure ci-dessus.

Elle fait partie intégrante des librairies de DaTool en s'intercalant entre la couche application et les librairies systèmes. Au niveau système une API ou une spécification permet la prise en charge des différents formats de sorties. Par exemple, l'API de la librairie GD permet de générer des fichiers au format PNG ou JPG. De même, la spécification du W3C décrit le format des fichiers SVG4. Ainsi, la couche d'abstraction est composée d'un pilote graphique générique. C'est ce pilote qui va permettre l'interface entre les primitives graphiques de haut niveau et les pilotes spécifiques à chaque API ou spécification. Dans chacun de ces pilotes sont implémentées les primitives de haut niveau. Par exemple, le pilote graphique générique comporte la fonction drawLine(x1, y1, x2, y2) pour tracer une ligne des coordonnées (x1, y1) (x2, y2). Dans le driver GD, cette fonction sera implmentée en utilisant la fonction gdImageLine. Tandis que dans le driver XML elle sera implmentée grâce à la balise ¡LINE x1= y1= x2= y2= /¿. L'encadré ci-dessous répertorie les différentes fonctions de l'API permettant de tracer des traits, des rectangles, choisir de styles de tracé (continu, pointillé, ...), des couleurs.

```
open (pGraphicDriver driver, char *file_name, int width, int height);
close (pGraphicDriver driver);
setFont (pGraphicDriver driver, char *font_name, unsigned int font_style);
setFontSize (pGraphicDriver driver, int font_size);
setFontColor (pGraphicDriver driver, const char *color_name);
drawString (pGraphicDriver driver, int x, int y, char *text);
```

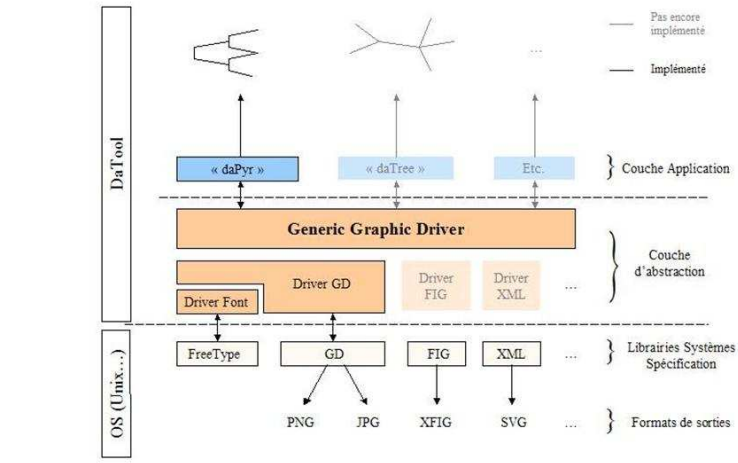


Fig. A.4: Interaction du système et des différentes couches

```
getStringHeight (pGraphicDriver driver, char *text);
getStringWidth (pGraphicDriver driver, char *text);
setLineStyle (pGraphicDriver driver, eLineStyle line_style);
setLineColor (pGraphicDriver driver, char *color_name);
setLineSize (pGraphicDriver driver, int line_size);
drawLine (pGraphicDriver driver, int x1, int y1, int x2, int y2);
setFillStyle (pGraphicDriver driver, eFillStyle fill_style);
setFillColor (pGraphicDriver driver, char *color_name);
drawRect (pGraphicDriver driver, int x1, int y1, int x2, int y2);

A ce jour, seul le pilote bitmap GD a été implmenté. Il permet la
prise en charge des formats PNG et JPG. Par ailleurs, il utilise le
pilote Font qui s'appuie sur la librairie FreeType pour la gestion des
polices TrueType. Cette couche d'abstraction va tre utilise par
l'algorithme de dessin dcrit dans la partie suivante.
```

La fonction de parcours d'un graphe ayant été détaillée ci-dessus, nous allons nous intéresser à la fonction de dessin.

```
Soit ymid(N) le milieu (sur le trait droit) du palier N.
Fonction de dessin~:
dessin (N) {
        SI (N a un prédcesseur droit)
                ALORS tracer le trait oblique vers son prdcesseur droit   (cf figure 11.a)
        SI (N a un prédcesseur gauche)
                ALORS tracer le trait oblique vers son prdcesseur gauche
                          Et tracer le trait droit, de son prdcesseur gauche  (cf figure 1
ymid (Pred_gauche(N)) = moyenne des ymid de ses successeurs
}
```

A.4 Conclusion et perspectives

Le projet de recherche DaTool est dédié à l'analyse de donnes biologiques en se basant sur des techniques de classification automatiques et non supervisées. La classification pyramidale, objet de ce mémoire, est une de ces méthodes de classification. Elle est définie comme une généralisation du modlèe de classification hiérarchique.

Nous avons pu voir, ensuite, que le dessin d'une telle pyramide devait se fixer un certain nombre de règles pour que son interprétation puisse être claire, facile et pertinente. Un cahier

des charges remplit ce rôle et donne une ligne directrice à respecter tout en permettant de prévoir dès à présent les fonctionnalités futures. Une couche d'abstraction graphique a été créée afin de s'affranchir de la dépendance avec les différents formats de fichiers. A ce jour, seul le pilote bitmap GD a été implment. Il permet la prise en charge des formats PNG et JPG. Par ailleurs, il utilise le pilote Font qui s'appuie sur la librairie FreeType pour la gestion des polices TrueType.

Enfin pour conclure, les perspectives d'un outil sont multiples et diverses. Il y a bien évidemment encore du travail à accomplir pour rendre cet outil convivial, agréable et performant pour les biologistes qui vont s'en servir. Une liste non exhaustive a déjà été établie dans le cahier des charges réalisé à cet effet. Tous ces efforts aboutiront un jour à l'achèvement de ce projet qui une fois encore, démontrera l'importance et la nécessité de l'informatique comme outil permettant de faire progresser la connaisssance des sciences du vivant.

Annexe B

Résultats des simulations

L'objectif de cette annexe est de présenter les résultats complets des simulations réalisées sur les algorithmes de filtrage que nous présentons (*cf.* chapitre 2 page 43) ; tout d'abord les figures correspondant aux simulations avec le critère d'agrégation du minimum, puis les tableaux regroupant les valeurs numériques permettant de tracer ces courbes.

B.1 Figures. Critère d'agrégation du minimum

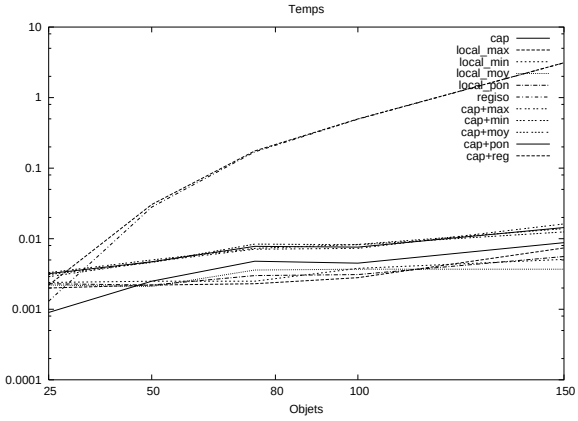


Fig. B.1: Temps de calcul des différents algorithmes

B.2 Tableaux de valeurs

B.2.1 Critère d'agrégation de la moyenne

Le tableau B.9 page 190 regroupe les valeurs du coefficient de corrélation copénétique entre les données initiales et : la pyramide avec inversions (ligne 2), la pyramide sans inversions obtenue par le seuillage avec les différents critères (lignes 3 à 6) et la pyramide sans inversions obtenue par le filtrage par régression isotone (ligne 7).

Le tableau B.10 page 191 regroupe les valeurs du coefficient de corrélation copénétique entre la matrice de dissimilarité induite et : la pyramide sans inversions obtenue par le seuillage avec avec les différents critères (ligne 2 à 5), la pyramide sans inversions obtenue par le filtrage par régression isotone (ligne 6).

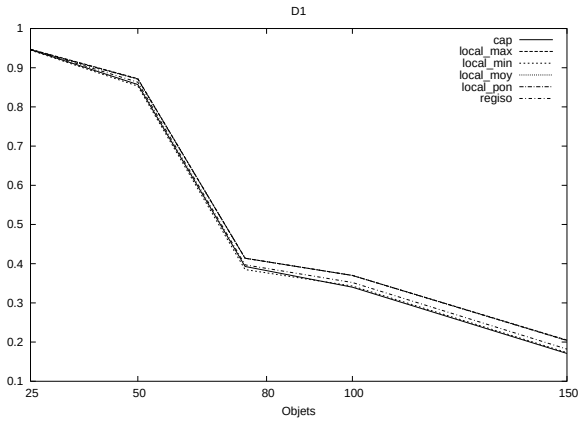


Fig. B.2: Critère D1

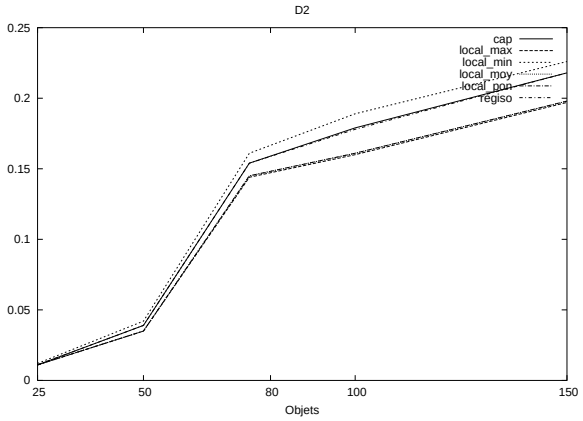


Fig. B.3: Critère D2

Algorithme	25	50	75	100	150
CAP	0.0025	0.0022	0.0046	0.0043	0.0089
Local_max	0.0024	0.0024	0.0031	0.0043	0.0056
Local_min	0.0017	0.0026	0.002	0.0035	0.0063
Local_moy	0.0029	0.0031	0.0037	0.0035	0.006
Local_pon	0.002	0.003	0.0025	0.0026	0.0062
Regiso	0.0012	0.0292	0.1765	0.4893	3.0572
CAP+Local_max	0.0049	0.0046	0.0077	0.0086	0.0145
CAP+Local_min	0.0042	0.0048	0.0066	0.0078	0.0152
CAP+Local_moy	0.0054	0.0053	0.0083	0.0078	0.0149
CAP+Local_pon	0.0045	0.0052	0.0071	0.0069	0.0151
CAP+Regiso	0.0037	0.0314	0.1811	0.4936	3.0661

Tab. B.1: Comparaison du temps CPU (en secondes) des différents algorithmes

D1	25	50	75	100	150
CAP	0.9746	0.7538	0.4764	0.4627	0.4116
CAP+Local_max	0.9743	0.7575	0.5081	0.4815	0.4437
CAP+Local_min	0.9744	0.7787	0.5219	0.5176	0.4685
CAP+Local_moy	0.9741	0.7545	0.5021	0.4767	0.4337
CAP+Local_pon	0.9741	0.7517	0.4994	0.4730	0.4300
CAP+Regiso	0.9744	0.7715	0.5200	0.5081	0.4652

Tab. B.2: Comparaison du critère D1 pour les différents algorithmes par rapport à la matrice de dissimilarité initiale

Le tableau B.11 page 191 regroupe les valeurs du coefficient de corrélation copénétique entre la matrice de dissimilarité obtenue après le filtrage par régression isotone et la pyramide sans inversions obtenue par le seuillage avec avec les différents critères.

Le tableau B.12 page 191 regroupe les distances euclidiennes entre les données initiales et : la pyramide avec inversions (ligne 2), la pyramide sans inversions obtenue par le seuillage avec les différents critères (lignes 3 à 6) et la pyramide sans inversions obtenue par le filtrage par régression isotone (ligne 7).

Le tableau B.13 page 191 regroupe les distances euclidiennes entre la matrice de dissimilarité induite et : la pyramide sans inversions obtenue par le seuillage avec avec les différents critères (ligne 2 à 5), la pyramide sans inversions obtenue par le filtrage par régression isotone (ligne 6).

Le tableau B.14 page 192 regroupe les distances euclidiennes entre la matrice de dissimilarité obtenue après le filtrage par régression isotone et la pyramide sans inversions obtenue par le

D1	25	50	75	100	150
CAP+Local_max	0.9998	0.9767	0.9227	0.8901	0.8376
CAP+Local_min	0.9997	0.9780	0.9197	0.9038	0.8838
CAP+Local_moy	0.9997	0.9810	0.9315	0.8981	0.8646
CAP+Local_pon	0.9997	0.9861	0.9381	0.9041	0.8952
CAP+Regiso	0.9999	0.9814	0.9076	0.9213	0.9044

Tab. B.3: Comparaison du critère D1 pour les différents algorithmes par rapport à la matrice de dissimilarité induite

D1	25	50	75	100	150
CAP+Local_max	0.9999	0.9946	0.9786	0.9670	0.9307
CAP+Local_min	0.9998	0.9965	0.9773	0.9807	0.9781
CAP+Local_moy	0.9998	0.9913	0.9696	0.9558	0.9134
CAP+Local_pon	0.9998	0.9864	0.9676	0.9528	0.8984

Tab. B.4: Comparaison du critère D1 pour les différents algorithmes de seuillage par rapport à la matrice obtenue avec le filtrage par régression isotone

seuillage avec avec les différents critères.

B.2.2 Critère d'agrégation du minimum

Le tableau B.9 page 190 regroupe les valeurs du coefficient de corrélation copénétique entre les données initiales et : la pyramide avec inversions (ligne 2), la pyramide sans inversions obtenue par le seuillage avec les différents critères (lignes 3 à 6) et la pyramide sans inversions obtenue par le filtrage par régression isotone (ligne 7).

Le tableau B.10 page 191 regroupe les valeurs du coefficient de corrélation copénétique entre la matrice de dissimilarité induite et : la pyramide sans inversions obtenue par le seuillage avec avec les différents critères (ligne 2 à 5), la pyramide sans inversions obtenue par le filtrage par régression isotone (ligne 6).

Le tableau B.11 page 191 regroupe les valeurs du coefficient de corrélation copénétique entre la matrice de dissimilarité obtenue après le filtrage par régression isotone et la pyramide sans inversions obtenue par le seuillage avec avec les différents critères.

Le tableau B.12 page 191 regroupe les distances euclidiennes entre les données initiales et : la pyramide avec inversions (ligne 2), la pyramide sans inversions obtenue par le seuillage avec les différents critères (lignes 3 à 6) et la pyramide sans inversions obtenue par le filtrage par régression isotone (ligne 7).

D2	25	50	75	100	150
CAP	0.003	0.167	4.384	4.603	7.133
CAP+Local_max	0.003	0.404	12.819	13.866	23.580
CAP+Local_min	0.003	0.118	3.320	3.667	5.655
CAP+Local_moy	0.003	0.364	7.379	10.582	17.577
CAP+Local_pon	0.003	0.221	7.002	8.068	11.792
CAP+Regiso	0.003	0.136	4.447	4.137	6.414

Tab. B.5: Comparaison du critère D2 pour les différents algorithmes par rapport à la matrice de dissimilarité initiale

D2	25	50	75	100	150
CAP+Local_max	$3.1\ 10^{-5}$	0.190	6.316	6.397	12.3732
CAP+Local_min	$2.4\ 10^{-5}$	0.035	0.668	0.614	0.957
CAP+Local_moy	$4.1\ 10^{-5}$	0.155	1.855	3.878	7.375
CAP+Local_pon	$4.1\ 10^{-5}$	0.025	1.619	2.575	3.011
CAP+Regiso	$7.3\ 10^{-5}$	0.028	0.465	0.464	0.707

Tab. B.6: Comparaison du critère D2 pour les différents algorithmes par rapport à la matrice de dissimilarité induite

Le tableau B.13 page 191 regroupe les distances euclidiennes entre la matrice de dissimilarité induite et : la pyramide sans inversions obtenue par le seuillage avec avec les différents critères (ligne 2 à 5), la pyramide sans inversions obtenue par le filtrage par régression isotone (ligne 6).

Le tableau B.14 page 192 regroupe les distances euclidiennes entre la matrice de dissimilarité obtenue après le filtrage par régression isotone et la pyramide sans inversions obtenue par le seuillage avec avec les différents critères.

Le tableau B.15 page 192 regroupe les valeurs du coefficient de corrélation copénétique entre les données initiales et : la pyramide avec inversions (ligne 2), la pyramide sans inversions obtenue par le seuillage avec les différents critères (lignes 3 à 6) et la pyramide sans inversions obtenue par le filtrage par régression isotone (ligne 7).

Le tableau B.16 page 192 regroupe les distances euclidiennes entre les données initiales et : la pyramide avec inversions (ligne 2), la pyramide sans inversions obtenue par le seuillage avec les différents critères (lignes 3 à 6) et la pyramide sans inversions obtenue par le filtrage par régression isotone (ligne 7).

D1	25	50	75	100	150
Pyr+Local_max	0.9999	0.9946	0.9786	0.9670	0.9307
Pyr+Local_min	0.9998	0.9965	0.9773	0.9807	0.9781
Pyr+Local_moy	0.9998	0.9913	0.9696	0.9558	0.9134
Pyr+Local_pon	0.9998	0.9864	0.9676	0.9528	0.8984

Tab. B.7: Comparaison du critère D2 pour les différents algorithmes de seuillage par rapport à la matrice obtenue avec le filtrage par régression isotone

Algorithme	25	50	75	100	150
CAP	0.0009	0.0025	0.0048	0.0045	0.0088
Local_max	0.0020	0.0022	0.0023	0.0028	0.0074
Local_min	0.0024	0.0025	0.0025	0.0038	0.0051
Local_moy	0.0022	0.0021	0.0036	0.0037	0.0037
Local_pon	0.0023	0.0022	0.0030	0.0031	0.0056
Regiso	0.0013	0.0280	0.1704	0.4945	3.1100
CAP+Local_max	0.0029	0.0047	0.0071	0.0073	0.0162
CAP+Local_min	0.0033	0.0050	0.0073	0.0083	0.0139
CAP+Local_moy	0.0031	0.0046	0.0084	0.0082	0.0125
CAP+Local_pon	0.0032	0.0047	0.0078	0.0076	0.0144
CAP+Regiso	0.0022	0.0305	0.1752	0.4990	3.1188

Tab. B.8: Comparaison du temps CPU (en secondes) des différents algorithmes

D1	25	50	75	100	150
CAP	0.946	0.857	0.393	0.340	0.171
CAP+Local_max	0.947	0.872	0.414	0.370	0.205
CAP+Local_min	0.945	0.853	0.385	0.343	0.173
CAP+Local_moy	0.946	0.871	0.414	0.370	0.204
CAP+Local_pon	0.946	0.871	0.414	0.370	0.204
CAP+Regiso	0.947	0.863	0.397	0.352	0.182

Tab. B.9: Comparaison du critère D1 pour les différents algorithmes par rapport à la matrice de dissimilarité initiale

D1	25	50	75	100	150
CAP+Local_max	0.996	0.980	0.950	0.904	0.821
CAP+Local_min	0.997	0.983	0.959	0.936	0.910
CAP+Local_moy	0.996	0.980	0.951	0.905	0.823
CAP+Local_pon	0.996	0.980	0.982	0.905	0.823
CAP+Regiso	0.998	0.992	0.970	0.962	0.944

Tab. B.10: Comparaison du critère D1 pour les différents algorithmes par rapport à la matrice de dissimilarité induite

D1	25	50	75	100	150
CAP+Local_max	0.998	0.989	0.970	0.941	0.880
CAP+Local_min	0.998	0.990	0.977	0.971	0.966
CAP+Local_moy	0.998	0.989	0.971	0.942	0.881
CAP+Local_pon	0.998	0.989	0.971	0.942	0.881

Tab. B.11: Comparaison du critère D1 pour les différents algorithmes de seuillage par rapport à la matrice obtenue avec le filtrage par régression isotone

D2	25	50	75	100	150
CAP	0.011	0.039	0.154	0.179	0.218
CAP+Locall_max	0.011	0.035	0.144	0.160	0.197
CAP+Local_min	0.012	0.042	0.161	0.189	0.226
CAP+Local_moy	0.011	0.035	0.145	0.161	0.198
CAP+Local_pon	0.011	0.035	0.145	0.161	0.198
CAP+Regiso	0.011	0.039	0.154	0.178	0.218

Tab. B.12: Comparaison du critère D2 pour les différents algorithmes par rapport à la matrice de dissimilarité initiale

D2	25	50	75	100	150
CAP+Local_max	$2.1\ 10^{-4}$	0.001	0.001	0.002	0.005
CAP+Local_min	$1.4\ 10^{-4}$	$7.4\ 10^{-4}$	$6.6\ 10^{-4}$	$6.1\ 10^{-4}$	0.001
CAP+Local_moy	$1.8\ 10^{-4}$	0.001	0.001	0.002	0.004
CAP+Local_pon	$1.8\ 10^{-4}$	0.001	0.001	0.002	0.004
CAP+Regiso	$0.5\ 10^{-4}$	$2.4\ 10^{-4}$	$1.9\ 10^{-4}$	$2.6\ 10^{-4}$	$7.0\ 10^{-4}$

Tab. B.13: Comparaison du critère D2 pour les différents algorithmes par rapport à la matrice de dissimilarité induite

D1	25	50	75	100	150
CAP+Local_max	$1.4\ 10^{-4}$	$0.8\ 10^{-4}$	0.001	0.002	0.004
CAP+Local_min	$0.9\ 10^{-4}$	$4.7\ 10^{-4}$	$4.5\ 10^{-4}$	$3.3\ 10^{-4}$	0.001
CAP+Local_moy	$1.2\ 10^{-4}$	$7.8\ 10^{-4}$	0.001	0.001	0.004
CAP+Local_pon	$1.2\ 10^{-4}$	$7.8\ 10^{-4}$	0.001	0.001	0.004

Tab. B.14: Comparaison du critère D2 pour les différents algorithmes de seuillage par rapport à la matrice obtenue avec le filtrage par régression isotone

D1	25	50	75	100	150
CAP	0.9746	0.7538	0.4764	0.4627	0.4116
pyr2hier_min	0.9743	0.7575	0.5081	0.4815	0.4437
pyr2hier_max	0.9744	0.7787	0.5219	0.5176	0.4685
pyr2hier_moy	0.9741	0.7545	0.5021	0.4767	0.4337
pyr2hier_wmoy	0.9741	0.7517	0.4994	0.4730	0.4300
cah_moy	0.9744	0.7715	0.5200	0.5081	0.4652
cah_ward	0.9744	0.7715	0.5200	0.5081	0.4652

Tab. B.15: Comparaison du critère D1 pour les différents algorithmes par rapport à la matrice de dissimilarité initiale

D2	25	50	75	100	150
CAP	0.9746	0.7538	0.4764	0.4627	0.4116
pyr2hier_min	0.9743	0.7575	0.5081	0.4815	0.4437
pyr2hier_max	0.9744	0.7787	0.5219	0.5176	0.4685
pyr2hier_moy	0.9741	0.7545	0.5021	0.4767	0.4337
pyr2hier_wmoy	0.9741	0.7517	0.4994	0.4730	0.4300
cah_moy	0.9744	0.7715	0.5200	0.5081	0.4652
cah_ward	0.9744	0.7715	0.5200	0.5081	0.4652

Tab. B.16: Comparaison du critère D2 pour les différents algorithmes par rapport à la matrice de dissimilarité initiale

Annexe C

Résultats des comparaisons des méthodes de constructions d'arbre

L'objectif de cette annexe est de présenter les résultats complets des comparaisons des méthodes de construction d'arbres (*cf.* chapitre 4 page 127) ; tout d'abord les résultats obtenus sur la base d'alignements de référence Sabmark (représentations graphiques des ACPs) puis les résultats obtenus sur la base d'alignements de référence Balibase (représentations graphiques des ACPs).

C.1 Sabmark

C.1.1 Base et références pré-définies

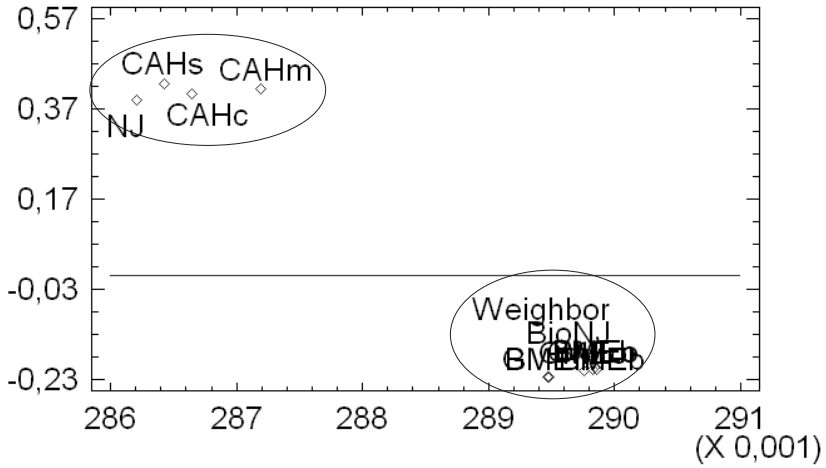


Fig. C.1: Représentation graphique d'une ACP des scores obtenus par différentes méthodes de construction d'arbre – Analyse réalisée sur la totalité de SabMark

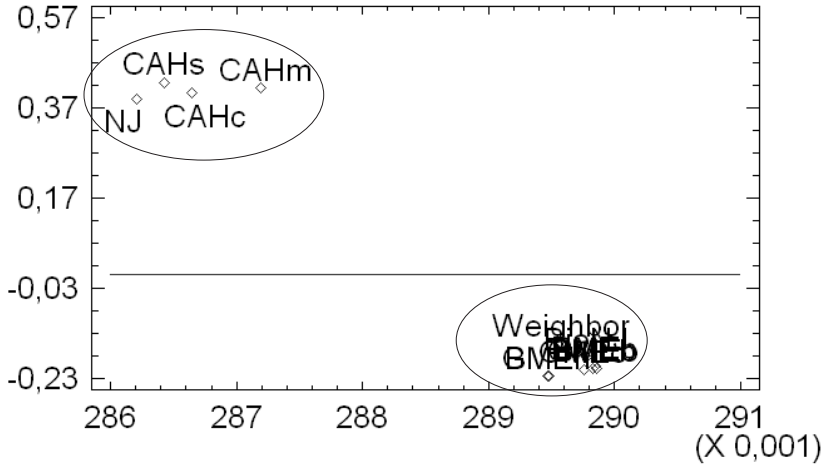


Fig. C.2: Représentation graphique d'une ACP des scores obtenus par différentes méthodes de construction d'arbre – Analyse réalisée sur la catégorie Superfamilles de SabMark

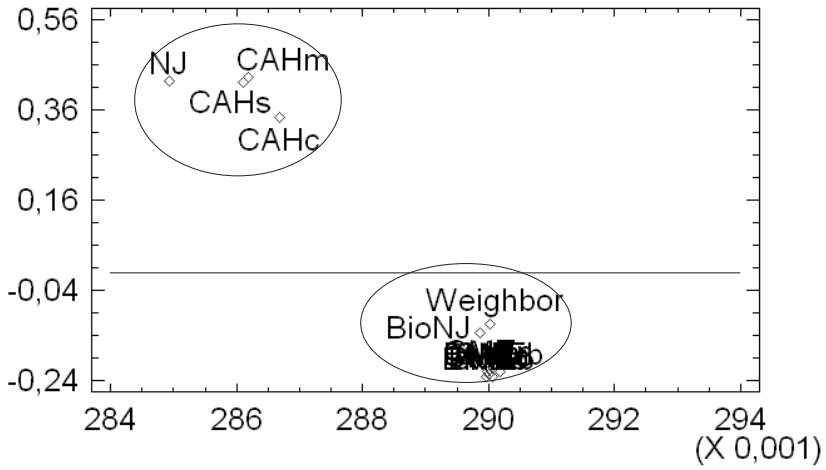


Fig. C.3: Représentation graphique d'une ACP des scores obtenus par différentes méthodes de construction d'arbre – Analyse réalisée sur la catégorie Twilight Zone de SabMark

Moins les 4

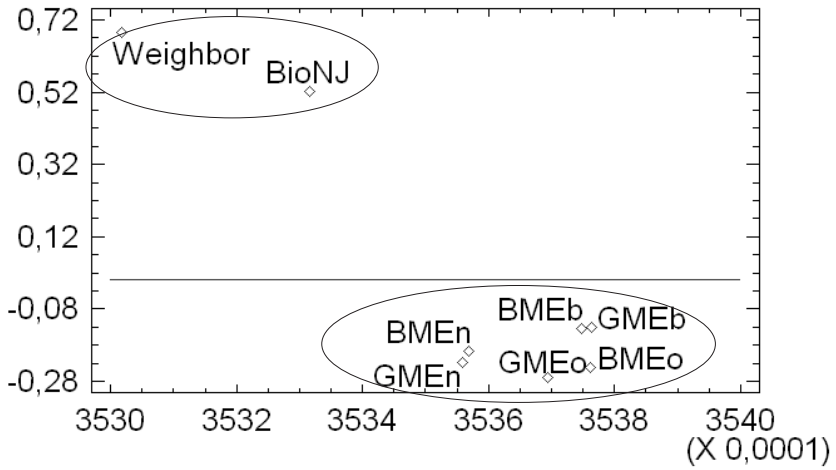


Fig. C.4: Représentation graphique d'une ACP des scores obtenus par différentes méthodes de construction d'arbre (hormis le groupe 1) – Analyse réalisée sur la totalité de SabMark

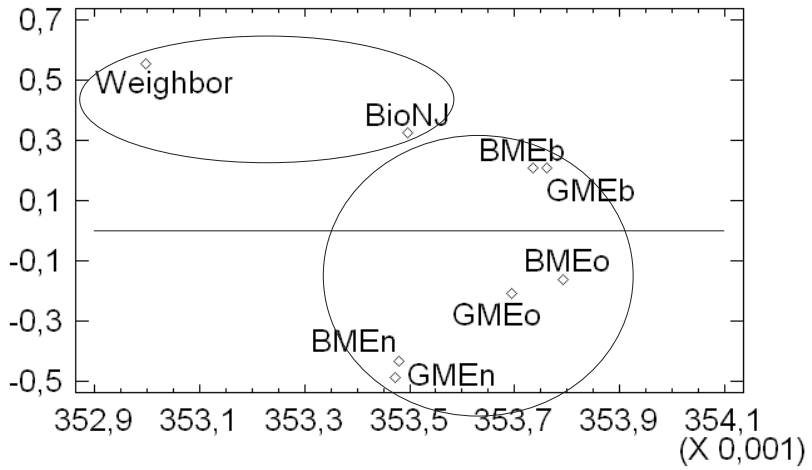


Fig. C.5: Représentation graphique d'une ACP des scores obtenus par différentes méthodes de construction d'arbre (hormis le groupe 1) – Analyse réalisée sur la catégorie Superfamille de SabMark

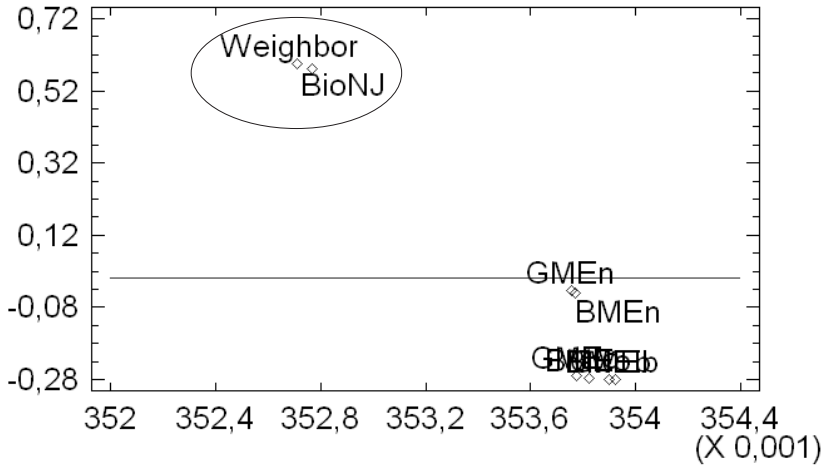


Fig. C.6: Représentation graphique d'une ACP des scores obtenus par différentes méthodes de construction d'arbre (hormis le groupe 1) – Analyse réalisée sur la catégorie Twilight zone de SabMark

C.1.2 Propriétés des séquences

Nombre de séquences

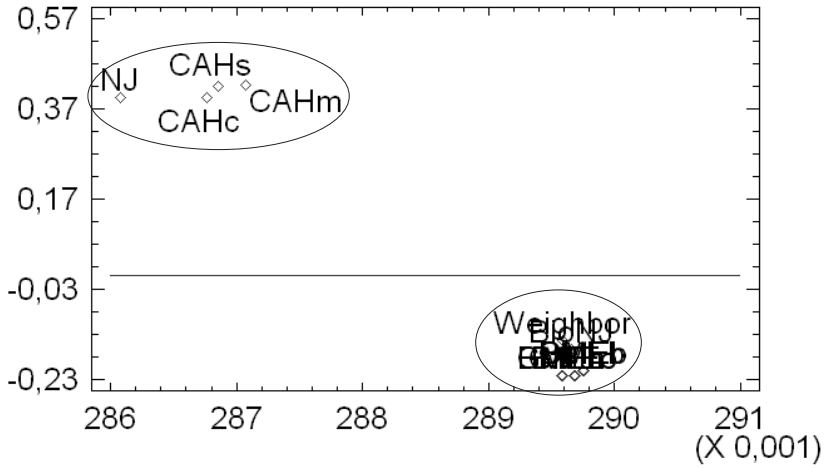


Fig. C.7: Représentation graphique d'une ACP des scores obtenus par différentes méthodes de construction d'arbre – Analyse réalisée sur les alignements comprenant de 3 à 8 séquences de SabMark

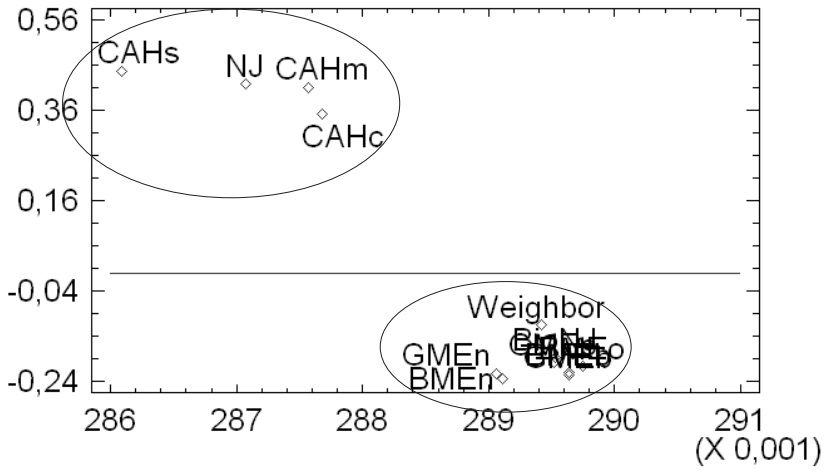


Fig. C.8: Représentation graphique d'une ACP des scores obtenus par différentes méthodes de construction d'arbre – Analyse réalisée sur les alignements comprenant de 9 à 24 séquences de SabMark

Longueur des séquences

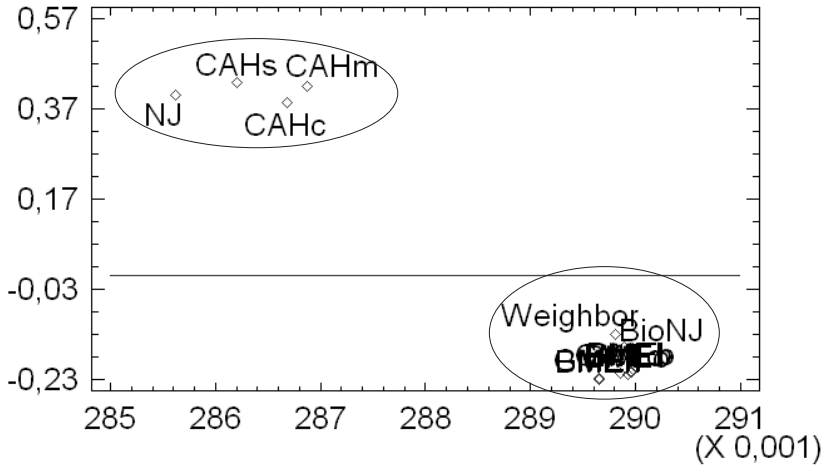


Fig. C.9: Représentation graphique d'une ACP des scores obtenus par différentes méthodes de construction d'arbre – Analyse réalisée sur les alignements de SabMark, comprenant des séquences composées de 34 à 199 acides aminés

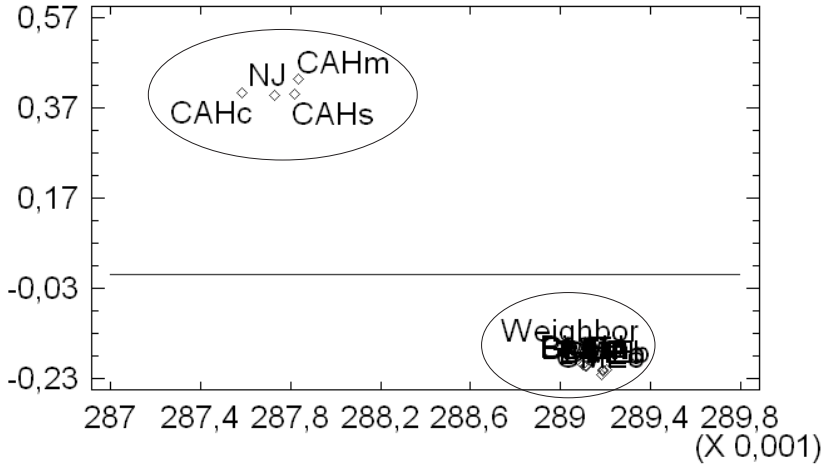


Fig. C.10: Représentation graphique d'une ACP des scores obtenus par différentes méthodes de construction d'arbre – Analyse réalisée sur les alignements de SabMark, comprenant des séquences composées de 200 à 600 acides aminés

Hydrophobicité des séquences

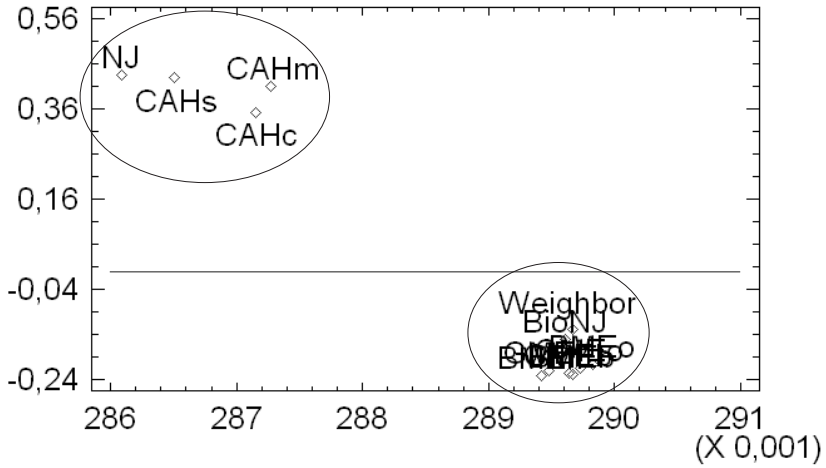


Fig. C.11: Représentation graphique d'une ACP des scores obtenus par différentes méthodes de construction d'arbre – Analyse réalisée sur les alignements de SabMark, comprenant des séquences composées de 26 à 38% de résidus hydrophobes

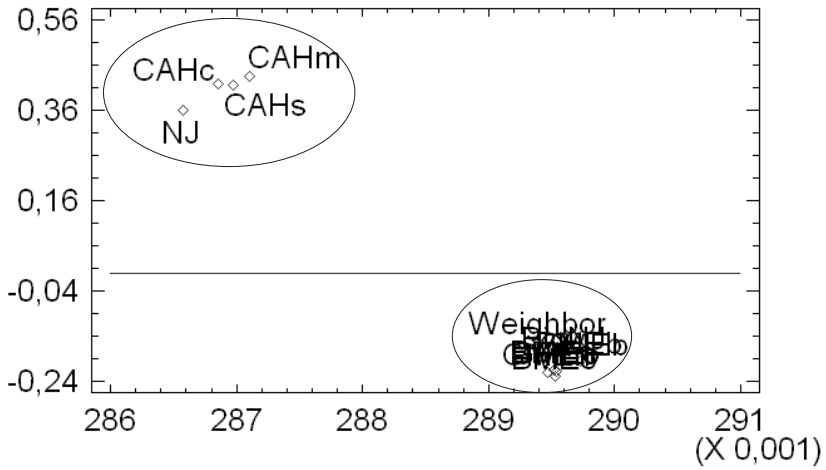


Fig. C.12: Représentation graphique d'une ACP des scores obtenus par différentes méthodes de construction d'arbre – Analyse réalisée sur les alignements de SabMark, comprenant des séquences composées de 39 à 54% de résidus hydrophobes

C.2 Balibase

C.2.1 Base et références pré-définies

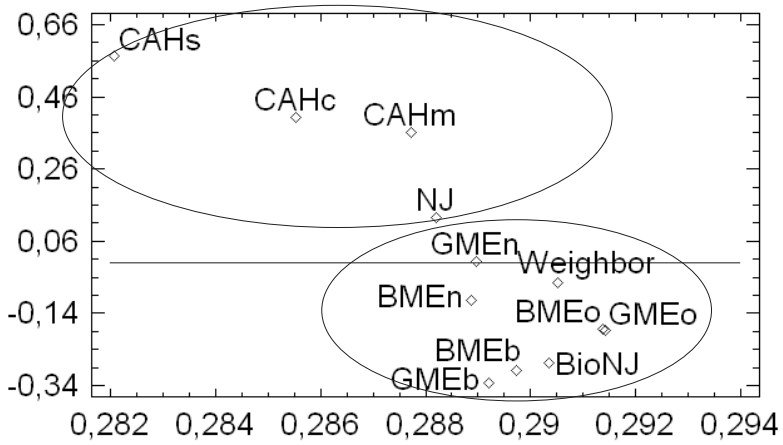


Fig. C.13: Représentation graphique d'une ACP des scores obtenus par différentes méthodes de construction d'arbre (hormis le groupe 1) – Analyse réalisée sur la totalité de Balibase

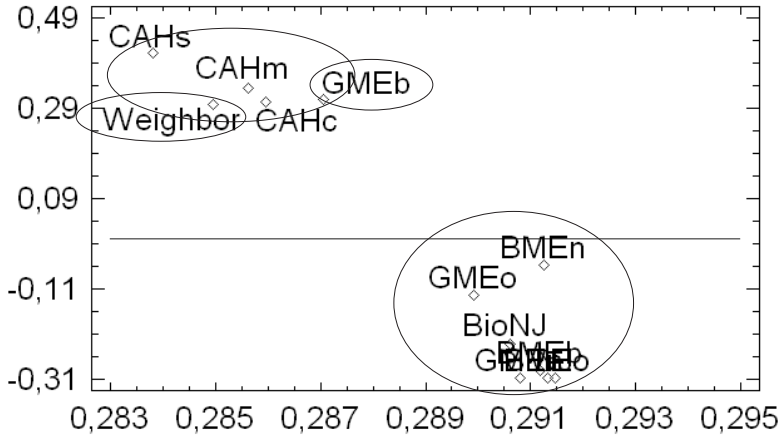


Fig. C.14: Représentation graphique d'une ACP des scores obtenus par différentes méthodes de construction d'arbre (hormis le groupe 1) – Analyse réalisée sur la référence 1_1 de Balibase

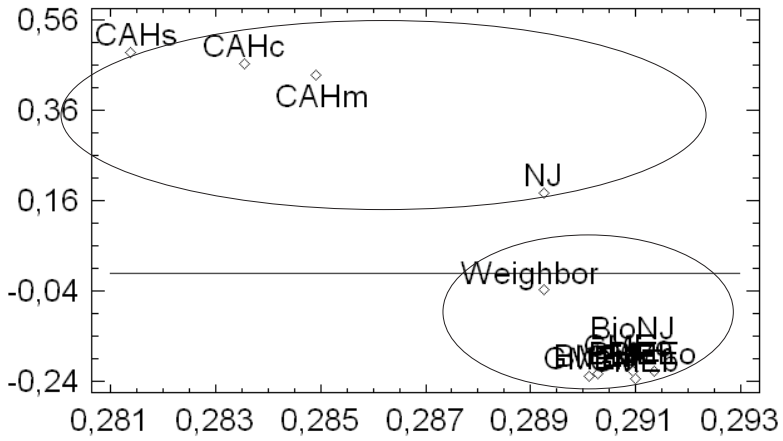


Fig. C.15: Représentation graphique d'une ACP des scores obtenus par différentes méthodes de construction d'arbre (hormis le groupe 1) – Analyse réalisée sur la référence 1_2 de Balibase

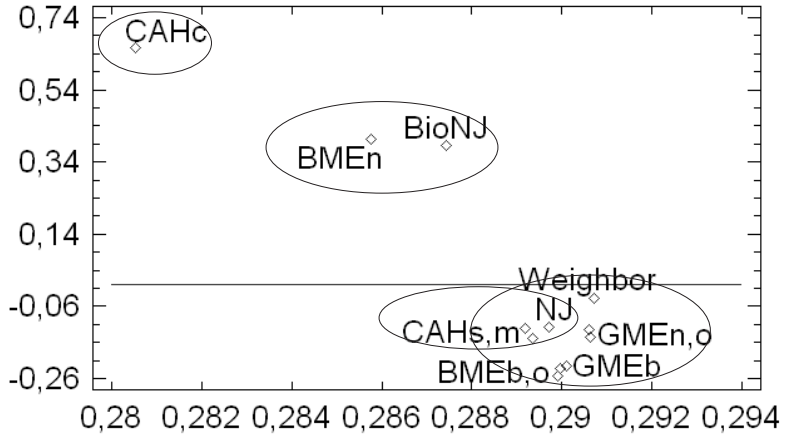


Fig. C.16: Représentation graphique d'une ACP des scores obtenus par différentes méthodes de construction d'arbre (hormis le groupe 1) – Analyse réalisée sur la référence 2 de Balibase

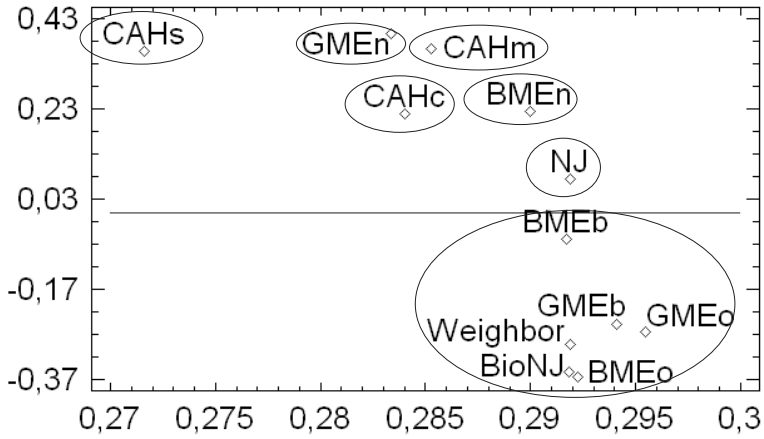


Fig. C.17: Représentation graphique d'une ACP des scores obtenus par différentes méthodes de construction d'arbre (hormis le groupe 1) – Analyse réalisée sur la référence 3 de Balibase

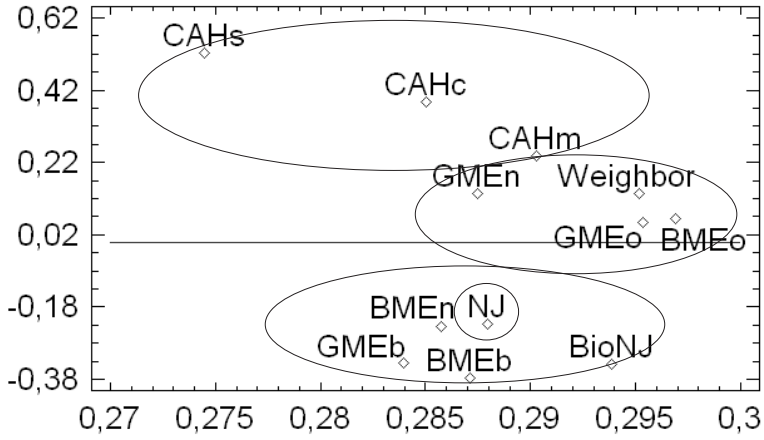


Fig. C.18: Représentation graphique d'une ACP des scores obtenus par différentes méthodes de construction d'arbre (hormis le groupe 1) – Analyse réalisée sur la référence 4 de Balibase

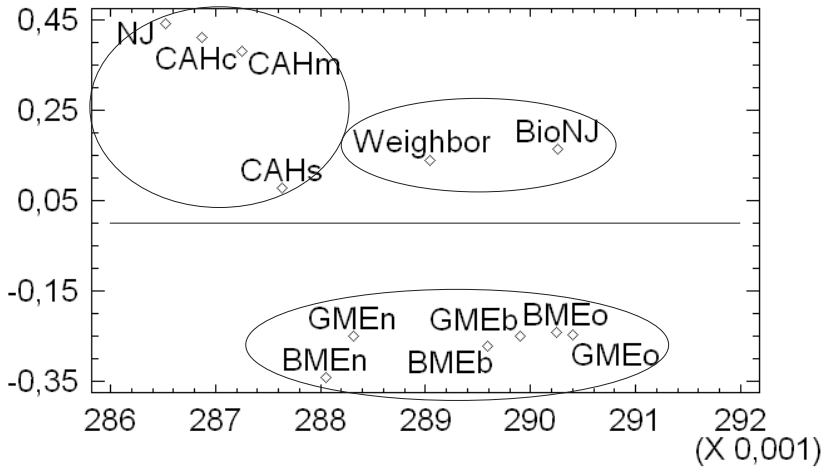


Fig. C.19: Représentation graphique d'une ACP des scores obtenus par différentes méthodes de construction d'arbre (hormis le groupe 1) – Analyse réalisée sur la référence 5 de Balibase

C.2.2 Propriétés des séquences

Nombre de séquences

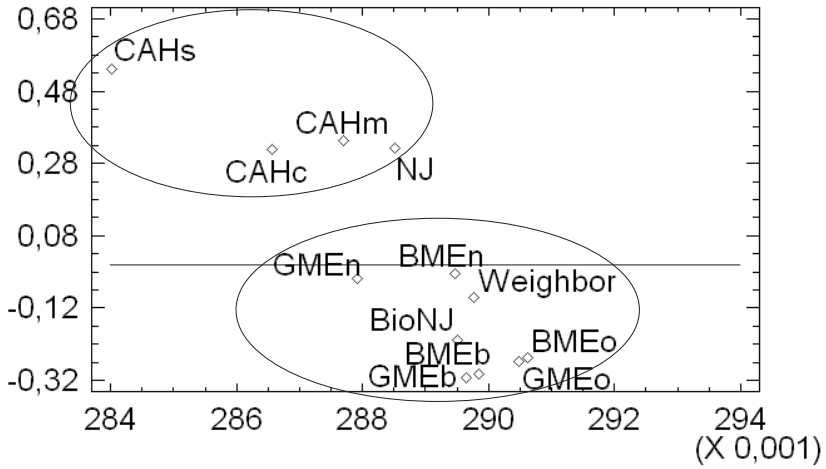


Fig. C.20: Représentation graphique d'une ACP des scores obtenus par différentes méthodes de construction d'arbre – Analyse réalisée sur les alignements de Balibase, composés de 4 à 19 séquences

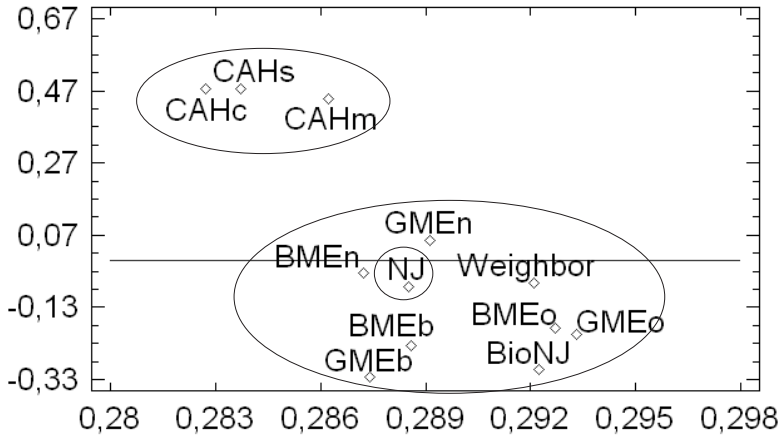


Fig. C.21: Représentation graphique d'une ACP des scores obtenus par différentes méthodes de construction d'arbre – Analyse réalisée sur les alignements de Balibase, composés de 20 à 59 séquences

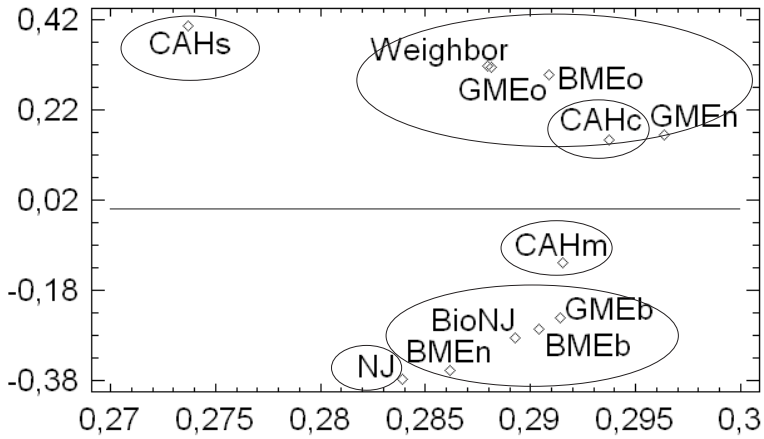


Fig. C.22: Représentation graphique d'une ACP des scores obtenus par différentes méthodes de construction d'arbre – Analyse réalisée sur les alignements de Balibase, composés de 60 à 142 séquences

Longueur des séquences

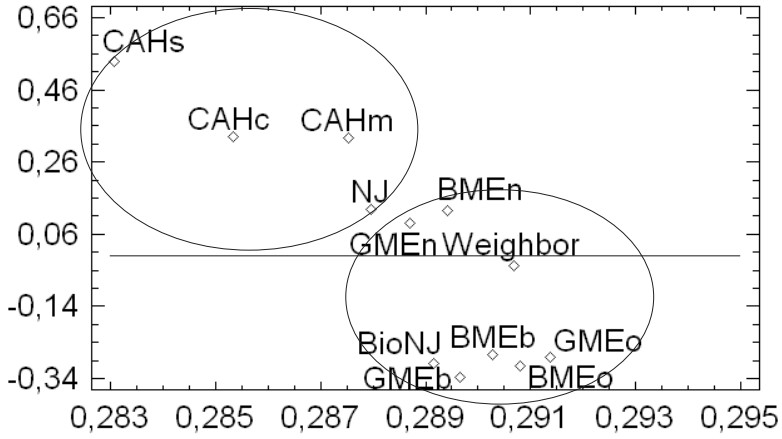


Fig. C.23: Représentation graphique d'une ACP des scores obtenus par différentes méthodes de construction d'arbre – Analyse réalisée sur les alignements de Balibase, comprenant des séquences composées de de 0 à 300 acides aminés

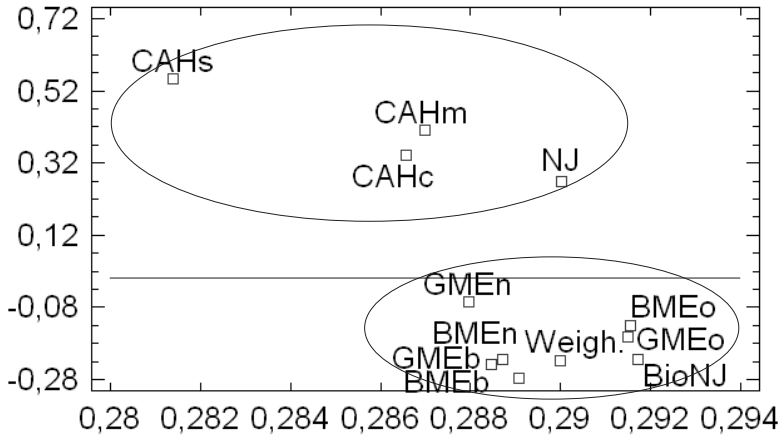


Fig. C.24: Représentation graphique d'une ACP des scores obtenus par différentes méthodes de construction d'arbre – Analyse réalisée sur les alignements de Balibase, comprenant des séquences composées de de 300 à 600 acides aminés

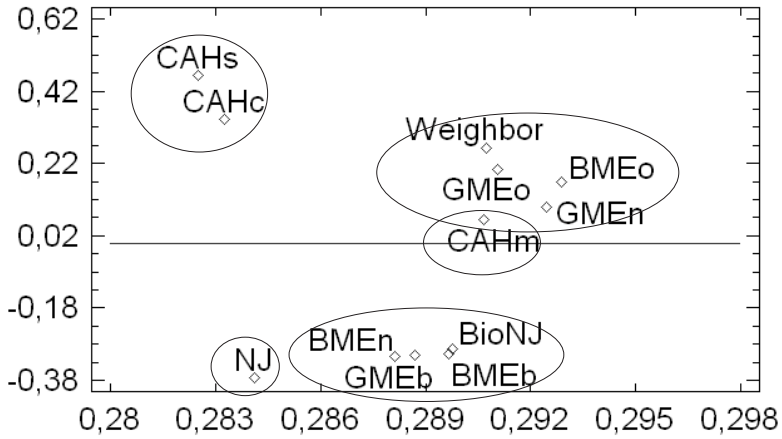


Fig. C.25: Représentation graphique d'une ACP des scores obtenus par différentes méthodes de construction d'arbre – Analyse réalisée sur les alignements de Balibase, comprenant des séquences composées de 600 à 1630 acides aminés

Hydrophobicité des séquences

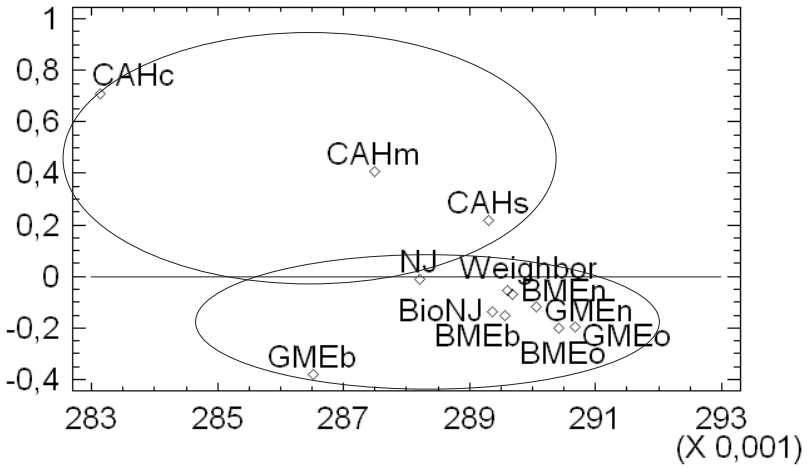


Fig. C.26: Représentation graphique d'une ACP des scores obtenus par différentes méthodes de construction d'arbre – Analyse réalisée sur les alignements de Balibase, comprenant des séquences composées de 24 à 37% de résidus hydrophobes

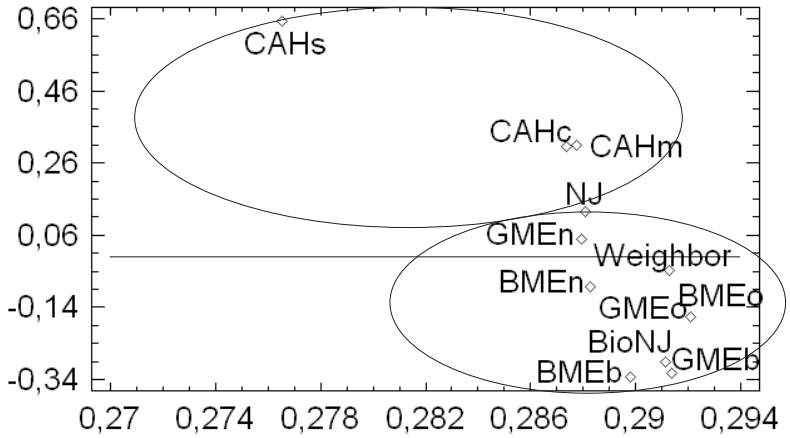


Fig. C.27: Représentation graphique d'une ACP des scores obtenus par différentes méthodes de construction d'arbre – Analyse réalisée sur les alignements de Balibase, comprenant des séquences composées de 38 à 45% de résidus hydrophobes

Table des figures

Liste des tableaux

Liste des Algorithmes

Bibliographie

[Ahola et al., 2006] Ahola, V., Aittokallio, T., Vihinen, M., and Uusipaikka, E. (2006). A statistical score for assessing the quality of multiple sequence alignments. *BMC Bioinformatics*, 7(484) :1408–1414.

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

[Aude, 1999] Aude, J.-C. (1999). *Analyse de génomes microbiens. Apports de la classification pyramidale*. Thèse de doctorat, Université Paris IX.

[Aude et al., 1999] Aude, J.-C., Diaz-Lazcoz, Y., Codani, J.-J., and Risler, J.-L. (1999). Application of the pyramidal clustering method to biological objects. *Computer and Chemistry*.

[Balaji et al., 2001] Balaji, S., Sujatha, S., Sai Chetan Kumar, S., and Srinivasan, N. (2001). Pali : a database of phylogeny and alignment of homologous protein structures. *Nucleic Acids Research*, 29(1) :61–65.

[Barton and Sternberg, 1987] Barton, G. and Sternberg, M. J. E. (1987). A strategy for the rapid multiple alignment of protein sequences : confidence levels from tertiary structure comparisons. *Journal of molecular biology*, 198 :327–337.

[Batbedat, 1990] Batbedat, A. (1990). *Les approches pyramidales dans la classification arborée*. Masson, Paris.

[Batzoglou, 2005] Batzoglou, S. (2005). The many faces of sequence alignment. *Briefings in Bioinformatics*, 6(1) :6–22.

[Benzecri, 1973] Benzecri, J. P. (1973). *L'analyse des données : 1 - la taxonomy*. Dunod, Paris.

[Benzecri, 1976] Benzecri, J. P. (1976). *Histoire et préhistoire de l'analyse des données*. Dunod, Paris.

[Benzécri, 1973] Benzécri, J. (1973). *L'analyse des données*, chapter 1-La taxonomie. Paris, dunod edition.

[Bertrand, 1986] Bertrand, P. (1986). *Etude de la représentation pyramidale*. Thèse de 3ème cycle, Université Paris IX - Dauphine.

[Bertrand, 2000] Bertrand, P. (2000). Set systems and dissimilarities. *Europ. J. Combinatorics*, 21 :727–743.

[Bertrand and Diday, 1990a] Bertrand, P. and Diday, E. (1990a). Une généralisation des arbres hiérarchiques : les représentations pyramidales. *Rev. Statistique Appliquée*, 38(3) :53–78.

[Bertrand and Diday, 1990b] Bertrand, P. and Diday, E. (1990b). Une généralisation des arbres hiérarchiques : les représentations pyramidales. *Rev. Statistique Appliquée*, 38 :53–78.

[Brenner et al., 1998] Brenner, S., C., C., and T.J.P., H. (1998). Assessing sequence comparison methods with reliable structurally identified distant evloutionnary relationships. *Proceedings of the National Academy of Sciences of the United States of America*, 95 :6073–6078.

[Briffeuil et al., 1998] Briffeuil, P., Baudoux, G., Lambert, C., De Bolle, X., Vinals, C., Feytmans, E., and Depiereux, E. (1998). Comparative analysis of seven multiple protein sequence alignment servers : clues to enhance reliability of predictions. *Bioinformatics*, 14(4) :357–366.

[Brito, 1991] Brito, P. (1991). *Analyse de données symbolique, pyramides d'héritage*. Thèse de doctorat *es* sciences, Université Paris IX - Dauphine.

[Brossier, 1980] Brossier, G. (1980). Représentation ordonnée des classifications hiérarchiques. *Statistiques et Analyse de données*, pages 31–44.

[Brucker, 2001] Brucker, F. (2001). *Modèles de classification en classes empiétantes*. PhD thesis, E.H.E.S.S.

[Bruno et al., 2000] Bruno, W., Socci, N., and A.L.Halpern (2000). Weighted neighbor joining : A likelihood-based approach to distance-based phylogeny reconstruction. *Molecular Biology and Evolution*, 17 :189–197.

[Carillo and Lipman, 1988] Carillo, H. and Lipman, D. (1988). The multiple sequence alignment problem in biology. *SIAM J. Appl. Math.*, 48(5) :1073–1082.

[Chakrabarti et al., 2006] Chakrabarti, S., Lanczycki, C., Panchenko, A., Przytycka, T., Thiessen, P., and Bryant, S. (2006). State of the art : refinement of multiple sequence alignments. *BMC Bioinformatics*, 7(499) :1408–1414.

[Cline et al., 2002] Cline, M., Hughey, R., and Karplus, K. (2002). Predicting reliable regions in protein sequence alignments. *Bioinformatics*, 18(2) :306–314.

[Codani et al., 1999] Codani, J.-J., Comet, J.-P., Aude, J.-C., Glémet, E., Wozniak, A., Risler, J.-L., Hénaut, A., and Slonimski, P. P. (1999). Automatic analysis of large scale pairwise alignments of protein sequences. *Methods in Microbiology*, 28 :229–244.

[Comet et al., 1999] Comet, J.-P., Aude, J.-C., Glémet, E., Hénaut, A., Risler, J.-L., Slonimski, P. P., and Codani, J.-J. (1999). Significance of z-value statistic of smith-waterman scores for protein alignments. *Special issue of Computers and Chemistry*, 23 :317–331.

[Corpet, 1988] Corpet, F. (1988). Multiple sequence alignment with hierarchical clustering. *Nucleic Acids Research*, 16(22) :10881–10890.

[Corter, 1996] Corter, J. E. (1996). Tree models of similarity and association. *Sage University Papers series : Quantitative Applications in the Social Sciences*, 112.

[Dayhoff et al., 1979] Dayhoff, M., Schwartz, R., and Orcutt, B. (1979). *Atlas of Protein Sequence and Structure*. Dayhoff M.O., Editor, National Biomedical Research Foundation : Washington DC.

[Dempster et al., 1977] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm (with discussion). *Journal of the Royal Statistical Society*, 39.

[Desper and Gascuel, 2002] Desper, R. and Gascuel, O. (2002). Fast and accurate phylogeny reconstruction algorithms based on the minimum-evolution principle. *Journal of Computational Biology*, 19 :687–705.

[Diday, 1984a] Diday, E. (1984a). Une représentation visuelle des classes empiétantes : les pyramides. Rapport de recherche No. 291, INRIA, Paris.

[Diday, 1984b] Diday, E. (1984b). Une représentation visuelle des classes empiétantes : les pyramides. rapport de recherche 291, INRIA.

[Do et al., 2005] Do, C., Mahabhasyam, M., Brodno, M., and Batzoglou, S. (2005). Probcons : Probabilistic consistency-based multiple sequence alignment. *Genome Research*, 15 :330–340.

[Doolittle, 2000] Doolittle, R. (2000). On the trail of protein sequences. *Bioinformatics*, 16(1) :24–33.

[Durand and Fichet, 1988] Durand, C. and Fichet, B. (1988). *Classification and related methods of data analysis*, chapter one-to-one correspondances in pyramidal representation : a unified approach, pages 85–90. Bock, H. H., Amsterdam, north-holland edition.

[Dykstra and Robertson, 1982] Dykstra, R. L. and Robertson, T. (1982). An algorithm for isotonic regression for two or more independent variables. *The Annals of Statistics*, 10(3) :708–716.

[Edgar, 2004a] Edgar, R. (2004a). Local homology recognition and distance measures in linear time using compressed amino acid alphabets. *Nucleic Acids Research*, 32(1) :380–385.

[Edgar, 2004b] Edgar, R. (2004b). Muscle : multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research*, 32(5) :1792–1797.

[Edgar and Sjölander, 2003] Edgar, R. and Sjölander, K. (2003). Satchmo : sequence alignment and tree construction using hidden markov madels. *Bioinformatics*, 19(11) :1404–1411.

[Eisen et al., 1998] Eisen, M. B., Spellman, P. T., Browndagger, P. O., and Dagger, D. B. (1998). Cluster analysis and display of genome-wide expression patterns. *PNAS*, 95 :14863–14868.

[Feng and Doolittle, 1987] Feng, D. and Doolittle, R. (1987). Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *Journal of Molecular Evolution*, 25 :351–360.

[Fichet, 1984] Fichet, B. (1984). Sur une extension des hiérachies et son équivalence avec certaines matrices de robinson. Journées statistiques de Montpellier.

[Friedman, 1937] Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *J. Am. Stat. Assoc.*, 32 :675–701.

[Gascuel, 1997] Gascuel, O. (1997). Bionj : an improved version of the nj algorithm based on a simple model of sequence data. *Molecular Biology and Evolution*, 14 :685–695.

[Gaul and Schader, 1994] Gaul, W. and Schader, M. (1994). Pyramidal classification based on incomplete dissimilarity data. *Journal of Classification*, 11 :171–193.

[Gil A.J. and A., 1998] Gil A.J., C. C. and A., A. (1998). On the efficiency and sensitivity of a pyramidal classification algorithm. *Journal of Economic Literature Classification*, C15 C88.

[Glémet and Codani, 1996] Glémet, E. and Codani, J.-J. (1996). Lassap, a large scale sequence comparison package. *Bioinformatics*, 13(2) :137–143.

[Gonnet et al., 1992] Gonnet, G., Cohen, M., and Benner, S. (1992). Exhaustive matching of the entire protein sequence database. *Science*, 257(5062) :1443–1445.

[Gonnet et al., 2000] Gonnet, G., Korostensky, C., and Benner, S. (2000). Evaluation measures of multiple sequence alignments. *Journal of Computational Biology*, 7 :261–276.

[Gordon, 1996a] Gordon, A. D. (1996a). *Clustering and classification*, chapter Hierarchical Classification, pages 65–121. Arabie, P. and Hubert, L.J. and De Soete, G., World Scientific.

[Gordon, 1996b] Gordon, A. D. (1996b). *Clustering and classification*, chapter Hierarchical Classification, pages 65–121. World Scientific.

[Gotoh, 1999] Gotoh, O. (1999). Multiple sequence alignment : algorithms and applications. *Advanced Biophysics*, 36 :159–206.

[Grasso and Lee, 2004] Grasso, C. and Lee, C. (2004). Combining partial order alignment and progressive multiple sequence alignment increases alignment speed and scalability to very large alignment problems. *Bioinformatics*, 20(10) :1546–1553.

[Guénoche, 1974] Guénoche, A. (1974). *Un algorithme "Branch and Bound" pour la sériation chronologique*, chapter Méthodes numériques en calcul scientifique et technique. Chatenay Malabry.

[Hartigan, 1967] Hartigan, J. A. (1967). Representation of similarity matrices by trees. *Journal of the American Statistical Association*, 62 :1140–1158.

[Henikoff and Henikoff, 1992] Henikoff, S. and Henikoff, J. G. (1992). Amino acid substitution matrices from protein blocks. *Proc. Nat. Acad. Sci.*, 89 :10915–10919.

[Holmes and Bruno, 2001] Holmes, I. and Bruno, W. (2001). Evolutionary hmms : a bayesian approach to multiple alignment. *Bioinformatics*, 17(9) :803–820.

[Hubert, 1974] Hubert, L. J. (1974). Some applications of graph theory to clustering. *Psychometrika*, 39(3) :283–309.

[Jardine and Sibson, 1968] Jardine, N. and Sibson, R. (1968). The construction of hierarchies and non-hierarchies classification. *the Computer Journal*, (11) :177–184.

[Jones, 1999] Jones, D. (1999). Protein secondary structure prediction based on position-specific scoring matrices. *J. Mol. Biol.*, 292 :195–202.

[Jones et al., 1992] Jones, D., Taylor, W., and Thornton, J. (1992). The rapid generation of mutation data matrices from protein sequences. *Comput. Appl. Biosci.*, 8(3) :275–282.

[Karplus and Hu, 2001] Karplus, K. and Hu, B. (2001). Evaluation of protein multiple alignments by sam-t99 using the balibase multiple alignment test-set. *Bioinformatics*, 17(8) :713–720.

[Katoh et al., 2005] Katoh, K., Kuma, K., Toh, H., and Miyata, T. (2005). Mafft version 5 : improvement in accuracy of multiple sequence alignment. *Nucleic Acids Research*, 33(2) :511–518.

[Katoh et al., 2002] Katoh, K., Misawa, K., Kuma, K., and Miyata, T. (2002). Mafft : a novel method for rapid multiple sequence alignment based on fast fourier transform. *Nucleic Acids Research*, 30(14) :3059–3066.

[Klipp et al., 2005] Klipp, E., Herwig, R., Wierling, C., and Lehrach, H. (2005). *Systems Biology in Practice.* Wiley, Weinheim.

[Koonin et al., 1997] Koonin, E., Mushegian, A., Galperin, M., and Walker, D. (1997). Comparison of archaeal and bacterial genomes : computer analysis of protein sequences predicts novel functions and suggests a chimeric origin for the archaea. *Mol Microbiol.*, 25 :619–637.

[Lasch, 1996] Lasch, R. (1996). Pyramidal clustering schemes. *Statistical papers 37.*

[Lassmann and Sonnhammer, 2002] Lassmann, T. and Sonnhammer, E. (2002). Quality assessment of multiple alignment programs. *FEBS letters*, 529 :126–130.

[Lebart et al., 1995] Lebart, L., Piron, M., and Morineau, A. (1995). *statistique exploratoire multidimensionnelle.* Dunod.

[Lee, 2003] Lee, C. (2003). Generating consensus sequences from partial order mulitple sequence alignment graphs. *Bioinformatics*, 19(8) :999–1008.

[Lee et al., 2002] Lee, C., Grasso, C., and Sharlow, M. (2002). Multiple sequence alignment using partial order graphs. *Bioinformatics*, 18(3) :452–464.

[Li, 2003] Li, K.-B. (2003). Clustalw-mpi : Clustalw analysis using distributed and parallel computing. *Bioinformatics*, 19(12) :1585–1586.

[Lipman et al., 1989] Lipman, D., Altschul, S., and Kececioglu, J. (1989). A tool for multiple sequence alignment. *Proc. Nat. Acad. Sci.*, 86 :4412–4415.

[Lipman and Pearson, 1985] Lipman, D. and Pearson, W. (1985). Rapid and sensitive protein similarity searches. *Science*, 227 :1435–1441.

[Louis et al., 2001] Louis, A., Ollivier, E., Aude, J.-C., and Risler, J.-L. (2001). Massive sequence comparisons as a help in annotating genomic sequences. 11 :1296–1303.

[Loytynoja and Milinkovitch, 2003] Loytynoja, A. and Milinkovitch, M. (2003). A hidden markov model for progressive multiple alignment. *Bioinformatics*, 19(12) :1505–1513.

[McClure et al., 1994] McClure, M., Vasi, T., and Fitch, W. (1994). Comparative analysis of multiple protein-sequence alignments methods. *Molecular Biology Evolution*, 11(4) :571–592.

[Mfoumoune, 1998] Mfoumoune, E. (1998). *Algorithme de classification Pyramidale*. PhD thesis, Université Paris IX Dauphine.

[Mi et al., 2005] Mi, H., Lazareva-Ulitsky, B., Loo, R., Kejariwal, A., Vandergriff, J., Rabkin, S., Guo, N., Muruganujan, A., Doremieux, O., Campbell, M., Kitano, H., and Thomas, P. (2005). The panther database of protein families, subfamilies, functions and pathways. *Nucleic Acids Research*, 33 :D284–D288.

[Morgenstern, 1999] Morgenstern, B. (1999). Dialign2 : improvement of the segment-to-segment approach to mulitple sequence alignment. *Bioinformatics*, 15(3).

[Morgenstern et al., 1996] Morgenstern, B., Dress, A., and Werner, T. (1996). Dialign : Multiple dna and protein sequence alignment based on segment-to-segment comparison. *Proc. Nat. Acad. Sci.*, 32 :571–592.

[Morrison and Ellis, 1997] Morrison, D. and Ellis, J. (1997). Effects of nucleotide sequence alignment on phylogeny estimation : A case study of 18s rdnas of apicomplexa. *Mol. Biol .Evol.*, 14(4).

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

[Notredame, 2002] Notredame, C. (2002). Recent progresses in multiple sequence alignement : a survey. *Pharmacogenomics*, 3(1) :211–218.

[Pages et al., 1979] Pages, J. P., Cailliez, F., and Escouffier, Y. (1979). Analyse factorielle : un peu d'histoire et de géométrie. *Revue de Statistique Appliquée*, 27(1) :5–28.

[Park and Teichmann, 1998] Park, J. and Teichmann, S. (1998). Divclus : an automatic method in the geanfammer package that finds homologous domains in single- and multi-domain proteins. *Bioinformatics*, 14 :144–150.

[Pascarella and Argos, 1992] Pascarella, S. and Argos, P. (1992). Analysis of insertions/deletions in protein structures. *Journal of Molecular Biology*, 224 :461–471.

[Phillips and Wheeler, 2000] Phillips, A.and Janies, D. and Wheeler, W. (2000). Multiple sequence alignment in phylogenetic analysis. *Molecular Phylogenetics and Evolution*, 16(3) :317–330.

[Qian and Eddy, 1995] Qian, S. and Eddy, W. F. (1995). An algorithm for isotonic regression on ordered rectangular grids. Technical Report 616, University of Carnegie Mellon, departement of Statistics.

[Raghava et al., 2003] Raghava, G., Searle, S., Audley, P., Barber, J., and Barton, G. J. (2003). Oxbench : A benchmark for evaluation of protein multiple sequence alignment accuracy. *BMC Bioinformatics*, 4(47).

[Reuchlin, 2003] Reuchlin, M. (2003). Contributions à l'histoire des methodes statistiques employees en psychologie. *Psychologie et Histoire*, 4(3-30).

[Robertson et al., 1988] Robertson, T., Wright, F. T., and Dykstra, R. L. (1988). *Order restricted statistical inference*. probability and mathematical statistics. John Wiley & Sons.

[Robinson, 1951] Robinson, W. S. (1951). A method for chronologically ordering archæological deposits. *Am. Antiquity*, 16(4) :293–301.

[Russell, 1914] Russell, B. (1914). *Our Knowledge of the External World as a Field for Scientific Method in Philosophy*.

[Saitou and Nei, 1987] Saitou, N. and Nei, M. (1987). The neighbor-joining method : A new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4 :406–425.

[Sammeth and Morgenstern, 2003] Sammeth, M. and Morgenstern, B.and Stoye, J. (2003). Divide-and-conquer multiple alignment with segment-based constraints. *Bioinformatics*, 19(Suppl. 2) :ii189–ii195.

[Sauder et al., 2000] Sauder, J. M., Arthur, J., and Dunbrack, R. (2000). Large-scale comparison of protein sequence alignments with structure alignments. *Proteins*, 40 :6–22.

[Schwartz and Pachter, 2006] Schwartz, A. and Pachter, L. (2006). Multiple alignment by sequence annealing. *Bioinformatics*, 23(ECCB 2006) :e24–e27.

[Smith and Smith, 1992] Smith, R. F. and Smith, T. F. (1992). Pattern-induced multi-sequence alignment (pima) algorithm employing secondary structure-dependent gap-penalties for comparative protein modelling. *Protein Engineering*, 5 :35–41.

[Smith and Waterman, 1981] Smith, T. F. and Waterman, M. S. (1981). Identification of common molecular subsequences. *Journal of Molecular Biology*, 147 :195–197.

[Sneath and Sokal, 1973] Sneath, P. and Sokal, R. (1973). *Numerical taxonomy*, pages 230–234. W.K.Freeman and Company, San Francisco.

[Sokal and Rohlf, 1962] Sokal, R. R. and Rohlf, F. J. (1962). The comparison of dendograms by objective methods. *Taxon*, 11 :33–40.

[Sonnhammer et al., 1997] Sonnhammer, E., Eddy, S., and Durbin, R. (1997). Pfam : a comprehensive database of protein domain families based on seed alignments. *Proteins*, 82(3) :405–420.

[Spearman, 1904] Spearman, C. (1904). General intelligence objectively determined and measured. *American Journal of Psychology*, 15 :201–292.

[Subramanian et al., 2005] Subramanian, A. R., Weyer-Menkhoff, J., Kaufmann, M., and Morgenstern, B. (2005). An improved algorithm for segment-based multiple sequence alignment. *BMC Bioinformatics*, 6(66).

[Taylor, 1988] Taylor, W. R. (1988). A flexible method to align large numbers of biological sequences. *Journal of Molecular Biology*, 88 :161–169.

[Teraprot, 2002] Teraprot (2002). Teraprot : une comparaison exhaustive de 70 génomes. `http ://www.infobiogen.fr/services/Teraprot/`.

[Thompson et al., 1994] Thompson, J. D., Higgins, D., and Gibson, T. (1994). Clustal w : improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22(22) :4673–4680.

[Thompson et al., 1999a] Thompson, J. D., Plewiak, F., and Poch, O. (1999a). Balibase : a benchmark alignment database for the evaluation of multiple alignment programs. *Bioinformatics*, 15(1) :87–88.

[Thompson et al., 1999b] Thompson, J. D., Plewiak, F., and Poch, O. (1999b). A comprehensive comparison of multiple sequence alignments programs. *Nucleic Acids Research*, 27(13).

[Vogt et al., 1995] Vogt, G., Etzold, T., and Argos, P. (1995). An assessment of amino acid exchange matrices in aligning protein sequences : the twilight zone revisited. *Journal of Molecular Biology*, 299(4) :816–831.

[Wallace et al., 2005] Wallace, I., O'Sullivan, O., and Higgins, D. (2005). Evaluation of iterative alignment algorithms for multiple alignment. *Bioinformatics*, 21(8) :1408–1414.

[Zhang and Kahveci, 2007] Zhang, X. and Kahveci, T. (2007). Qoma : quasi-optimal multiple alignment of protein sequences. *Bioinformatics*, 23(2) :162–168.