THÈSE de DOCTORAT de l'UNIVERSITÉ PARIS 6

Spécialité :

Informatique

présentée

par M. Laurent Guéguen

pour obtenir le grade de DOCTEUR de l'UNIVERSITÉ PARIS 6

Sujet de la thèse :

Partitionnement maximalement prédictif sous contrainte d'ordre total

Applications aux séquences génétiques

soutenue le 18 janvier 2000

devant le jury composé de :

- M. Antoine Danchin
- M. Olivier Gascuel(rapporteur)
- M. Alain Guénoche (rapporteur)
- M. Jacques Lebbe (directeur)
- M. Jean-Charles Pomerol (président)
- M. Alain Viari

à Jacques,

Au lecteur impatient de lire cette thèse, je demande un instant de s'arrêter ici, car il serait vraiment dommage que je taise quelques remerciements & un hommage aussi.

Je remercie Jean-Charles POMEROL *de m'avoir accueilli sous sa responsabilité au sein du Laboratoire d'Informatique de Paris 6, & de m'avoir aidé à mener à bien cette thèse.*

Je sais gré à Olivier GASCUEL *& Alain* GUÉNOCHE *de m'avoir fait l'honneur d'être mes rapporteurs & membres de mon jury. Leurs remarques m'ont permis d'envisager la problématique de ce travail sous de nouveaux points de vue.*

Je remercie Antoine DANCHIN *de s'être intéressé à ce travail & d'avoir accepté de participer à mon jury.*

Parce que travailler avec lui est aussi intéressant qu'agréable, j'exprime toute ma sympathie & mes remerciements à Alain VIARI *pour sa serviabilité, son intelligence, & sa passion scientifique.*

Je rends hommage à Jacques LEBBE*. Il m'a offert de travailler sur un sujet qui lui tenait à cœur, & m'a permis de découvrir un pan de la recherche pour moi jusqu'alors inconnu. J'avais à son approche le plaisir d'être submergé par les flots d'idées qui jaillissaient de son imagination & de sa verve débordantes, & de me trouver emporté par son enthousiasme. Toujours pragmatique, il restait néanmoins ancré à la réalité, où ses intuitions trouvent encore un sol fertile, dans les multiples champs d'étude qu'il chérissait.*

Je tiens aussi à remercier les gens du tout frais Laboratoire d'Informatique & de Systématique.

En particulier, je remercie Régine VIGNES-LEBBE *qui m'a toujours supporté dans ce travail. En dépit de toutes ses difficultés, elle a toujours fait de son mieux pour m'apporter son aide & me faire partager sa rigueur scientifique.*

Je remercie Bernard LEPEN*, grâce à qui les multiples machines & systèmes d'exploitation du laboratoire coexistent dans un silico-système viable. J'ai copieusement abusé de sa constante disponibilité pour mettre à mon profit sa compétence, sa sérénité, & sa persévérance.*

J'exprime ma gratitude à tous ceux à qui j'ai fait subir ma prose, ma musique, & autres désagréments d'existence. Dans le désordre, Jérôme, Robert (dit Bob), André, Pierre, Marie-Agnès, Cyril (& ses mitochondries), Guillaume, Guillaume.

Je remercie en outre tous les gens qui m'ont apporté leur aide, Jean BROUTIN*, Ghislaine* MARY*, Joël* POTHIER*, & ceux que j'oublie...*

Enfin, je serai éternellement reconnaissant à mes parents, à mes frangins, à ma famille, & à mes ami(e)s, de leur amour & de leur affection.

Table des matières

Introduction

De plus en plus, l'informatique doit relever de nombreux défis qui lui sont posés par la biologie. Ces défis sont suscités par les possibilités de calcul et de communication offertes aujourd'hui par les outils informatiques, et qui fertilisent l'imagination et piquent la curiosité des scientifiques. L'informatique permet déjà d'aider considérablement la recherche biologique, dans des champs d'études aussi vastes et dissemblables que la systématique, l'imagerie médicale, la génétique, la biologie moléculaire... Comme toute discipline scientifique digne de ce nom, elle est aussi son propre terreau. Dans le domaine de la biologie moléculaire, l'utilisation qui en est faite participe du séquençage, de la gestion et de la mise à disposition de quantités faramineuses d'informations sur les génomes et les protéines. Pour pouvoir analyser toutes ces données, l'informatique est devenue un outil indispensable au biologiste.

Parmi les disciplines dont l'essence même est justement d'analyser n'importe quel type d'ensembles de données, la classification – ou science de bien classer – se trouve au confluent des statistiques, des mathématiques et de l'informatique. Sur un ensemble de données, la classification vise à construire de façon méthodique des classes de manière à révéler une structure de cet ensemble. Elle a de nombreuses applications en biologie, dans des registres aussi variés que l'analyse d'images, l'écologie, la reconstruction phylogénétique, ou encore l'analyse de séquences protéiques et génétiques. En général, l'utilisation qui est faite de la classification sur des séquences biologiques concerne le regroupement et l'organisation de séquences entre elles, selon des critères de ressemblance structurelle, fonctionnelle, ou de phylogénie.

L'originalité de notre approche est de chercher à construire des classifications *dans* des séquences, et non *entre* des séquences. Notre volonté est de définir des critères et des méthodes satisfaisants pour opérer des partages en segments de séquences qualitatives, de telle sorte que ces segments décrivent au mieux la structure de l'ensemble des objets de la séquence et permettent *en même temps* une description fiable de chacun des objets de cette séquence. Ainsi, nous allons chercher à mettre au point un procédé qui puisse fournir des résumés pertinents de séquences qualitatives selon un large éventail de critères. Ces critères seront fondés sur la notion de prédiction, et nous appellerons ce procédé partitionnement maximalement prédictif sous contrainte d'ordre total. Par la conception d'un programme ad hoc, nous nous donnerons les moyens de continuer à étudier le partitionnement à l'aide d'expérimentations.

Du point de vue de la classification, partager des séquences qualitatives est un cas extrêmement particulier. En général, la classification travaille sur des ensembles d'objets extrêmement variés, tant qualitatifs que quantitatifs. Considérer des séquences d'objets qualitatifs présente justement l'intérêt d'une nouvelle approche dans un contexte extrêmement intéressant. Les séquences d'objets qualitatifs, de par leur structure séquentielle, fournissent une très grande variété de types d'organisation interne, selon les multiples règles lexicales selon lesquelles les objets peuvent se suivre dans ces séquences ; il est intéressant d'évaluer comment la classification peut être adaptée à la découverte de ces organisations. La nouveauté de cette approche suscite l'espoir de découvrir des méthodes et propriétés qui, en retour, auront leur utilité en classification dans un contexte général.

Du point de vue de l'analyse de séquences, notre approche est aussi intéressante. Beaucoup de travaux lexicaux se basent sur l'extraction de particularités ponctuelles au sein de séquences, quitte après à observer la répartition de ces particularités. Au contraire, nous cherchons à opérer des partages en segments sur lesquels il y a le moins d'anomalies en regard de certains critères. Si ces critères correspondent à des caractéristiques biologiques, chaque segment ainsi formé répond de façon homogène à une caractéristique biologique particulière. Inversement, si des critères partagent « bien » une séquence génétique, il peut être intéressant de trouver des explications statistiques et/ou biologiques à ce partage. Ainsi, on pourrait révéler et différencier des segments d'une séquence, et la description de ces segments permettrait d'extraire des connaissances sur la composition et l'organisation de la séquence étudiée. L'intérêt biologique de ceci est tel qu'il nous apparaît nécessaire de développer des outils efficaces pour réaliser des partitionnements maximalement prédictifs.

Notre travail sera progressif. Nous commençons par situer notre approche dans les deux contextes de la classification et de la redescription de séquences. Par l'étude des travaux déjà accomplis en classification, et en particulier par la notion de prédiction de Gower, nous exprimons les principes qui définissent le partitionnement maximalement prédictif. Nous envisageons aussi le contexte de la redescription de séquences de lettres. Le problème du choix du « bon » type de classification sur un ensemble est très important en classification, et nous l'abordons sous l'aspect du partitionnement de séquences.

Dans le deuxième chapitre, nous présentons l'algorithme de Fisher qui réalise le partitionnement sur des séquences en un temps quadratique avec la longueur de la séquence. Nous développons ensuite un nouvel algorithme qui réalise le partitionnement maximalement prédictif de séquences au sens où il a été défini dans la première partie. Cet algorithme présente entre autres l'avantage d'être linéaire avec la taille de la séquence. Nous concevons aussi une syntaxe qui permet de fixer un cadre précis et large de critères selon lesquels le partitionnement maximalement prédictif peut être effectué. Nous mettons enfin en évidence une relation d'équivalence sur les résultats de nos partages, de manière à pouvoir améliorer l'algorithme présenté juste avant.

Après la conception d'algorithmes et de critères rigoureux applicables à ces algorithmes, il est naturel de développer un programme qui permette d'effectuer des partitionnements selon les critères définis. Ce programme – nommé `pampres` – a été developpé. Nous en présentons dans le troisième chapitre la conception ainsi que les performances.

Grâce à son efficacité, le programme `pampres` permet de réaliser de nombreuses expérimentations. L'observation de ces expérimentations a révélé des phénomènes du partitionnement très intéressants, comme nous le présentons dans le quatrième chapitre. Dans un cadre particulier, nous montrons que les observations effectuées empiriquement s'appliquent rigoureusement. La fin de ce chapitre concerne un retour sur le problème du choix du nombre de classes, tel qu'il a été présenté à la fin du premier chapitre.

Dans le cinquième chapitre, après quelques rappels biologiques, pour situer notre approche dans le contexte de la recherche biologique qui s'en approche le plus, nous donnons un aperçu de critères et méthodes qui ont déjà servi à la détermination de segments dans les séquences génétiques. Ensuite, nous présentons quelques expérimentations du partitionnement maximalement prédictif sur des génomes. Ce chapitre vise à éclairer différents aspects caractéristiques des séquences génétiques qui justifient une utilisation plus poussée, à l'avenir, du partitionnement maximalement prédictif à l'analyse de séquences.

Chapitre 1

Contextes du partitionnement maximalement prédictif

« La simplicité n'a pas besoin d'être simple, mais du complexe resserré et synthétisé (Cf. Pataph.). »
A. Jarry
Les Minutes de sable mémorial

1.1 De la classification en général

1.1.1 Principes

1.1.1.1 Présentation

D'un ensemble d'objets, une classification vise à condenser l'information fournie par la multitude des descriptions individuelles de ces objets, en un nombre moindre de descriptions générales de groupes d'objets [FVN71]. Ou bien, en citant Gordon [Gor96c], à résoudre le problème de construire des *résumés informatifs* des relations au sein de cet ensemble, en procédant à des regroupements de ces objets. Comme l'écrit Gilmour [Gil37], la classification est une étape nécessaire dans le processus par lequel l'esprit humain induit une connaissance *ordonnée* de l'univers. Sur un ensemble donné, elle cherche à construire des classes (ou groupes) d'objets qui mettent en lumière une organisation « naturelle » [Jen68] de cet ensemble. Cette organisation peut aussi bien être au niveau des relations entre les classes, qu'au sein même de ces classes, selon les critères qui ont présidé à leur construction. Bien sûr, la notion d'organisation « naturelle » est assez floue pour laisser le plein champ à de nombreux points de vues, mais les différents concepts sous-jacents sont « liés par un air de famille » [Col86]. Dans un contexte taxonomique, Gilmour affirme qu'un système de classification est d'autant plus naturel qu'il permet d'émettre beaucoup d'hypothèses concernant les classes [Gil37].

Une classification peut aussi inciter à *discriminer* entre eux les groupes d'objets ainsi formés, de telle manière que ces groupes soient les plus différentiables les uns des autres. En effet il semble inutile de partager un groupe d'objets lorsque l'intégrité de celui-ci est satisfaisante. Et, quitte à diviser un ensemble en plusieurs classes, il peut paraître généralement plus souhaitable que cette division mette le plus possible en relief une éventuelle scission « naturelle » de l'ensemble. La notion de séparation des classes étant une notion duale de celle d'union d'une classe, les critères de séparation devront être ceux qui mettent le plus en défaut ceux d'union.

Dans la suite, classification sera utilisée au sens de classification automatique, c'est-à-dire intégrée dans un processus automatisé (informatique), dans lequel l'utilisateur n'intervient qu'au moment *initial* des choix. Ceci oblige à définir, lorsque l'on voudra effectuer une classification, de façon précise – et critique – tout ce qui se rapporte à celle-ci : l'ensemble des données, le type de classification recherché, les critères qui la guident, la méthode suivie.

1.1.1.2 Différents types de classifications

Quel que soit le critère d'unification des classes, il faut, lorsqu'il s'agit de classifier, choisir un type de classification, c'est-à-dire une organisation des classes qui la constitueront. En effet, selon la problématique, il peut sembler opportun de ranger l'intégralité de l'ensemble étudié en classes disjointes – une partition – ou créer un emboîtement de classes – une hiérarchie (cf figure 1.1). Si ces deux types de classifications sont les plus étudiés, d'autres sont possibles, comme permettre que des objets soient rangés simultanément dans plusieurs classes [JS68], ou bien créer des classes floues [Rus69, FK97]. Ainsi, certaines classifications seront déclarées admissibles car elles respectent un type précis.

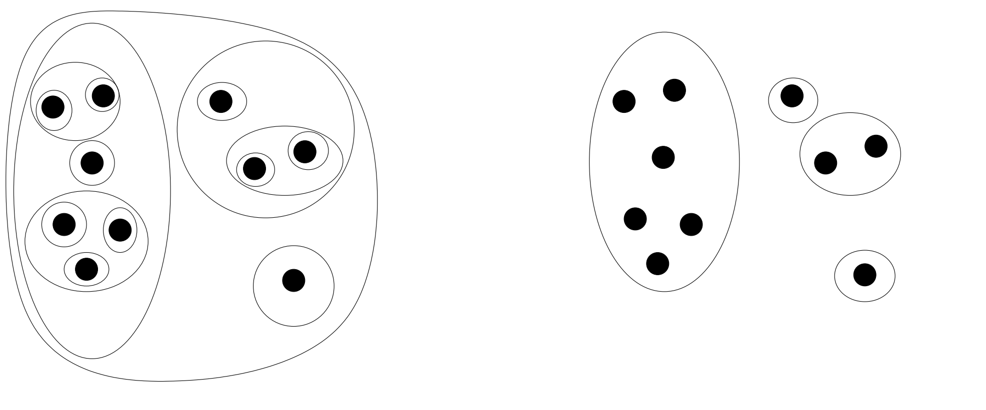


Fig. 1.1 – À gauche, une hiérarchie. À droite, une partition.

1.1.1.3 Homogénéité vs hétérogénéité

Chercher à regrouper entre eux des objets similaires, et ainsi former des classes les plus homogènes possibles, est un objectif tout-à-fait normal de la classification. En fait, il s'agit même de l'essence de la classification comme en statistique [Fis58] et en taxonomie [Gil40, Gil51].

Naturellement, le critère de séparation des groupes est alors lié à la notion d'hétérogénéité. Par exemple, dans un espace de points euclidien, une classification peut être évaluée non seulement en fonction de la concentration de chacune de ses classes, mais aussi en fonction de la dispersion de l'ensemble de ses classes [CH74, BM98].

Mais, pour importante qu'elle est, l'homogénéité n'est pas le seul critère imaginable pour évaluer une classe. Ainsi, par exemple, dans [BW77], l'objectif est de segmenter une séquence numérique en zones de variance maximale.

1.1.1.4 Étiquetage

Dans certains problèmes, une démarche classificatoire doit permettre d'accéder à une caractérisation interprétable des groupes. Ainsi, la critique souvent faite à la classification qui ne s'appuie que sur des valeurs numériques est justement qu'elle génère des classes peu *informatives* quant à leur contenu, d'une façon indépendante de l'information que l'on voudrait pouvoir extraire de chaque classe, sous la forme d'une *description* des classes [Gow74].

Pour construire une classification descriptive, il y a plusieurs façons d'aborder le problème. On peut chercher à décrire *a posteriori* les classes construites selon des critères non descriptifs, mais rien ne garantit que cela soit toujours possible de façon « simple » [Pan83, SS92] ; il semble alors plus raisonnable de considérer des caractères qui soient aisément utilisables, par exemple qualitatifs ou discrets, à la fois pour décrire les classes et évaluer la pertinence d'un objet à appartenir à telle ou telle. Ainsi, la dynamique classificatoire n'est plus axée sur les objets considérés au sein de leur ensemble, mais sur la relation qui les lie à des descriptions de classes. L'élaboration de descriptions valables est faite à partir des variables du système, ce qui peut être difficile, en particulier pour les variables continues [Dal85]. Une classification descriptive sera évaluée par la pertinence des descriptions qu'elle engendre. Dans [Col84], Colless présente la construction d'une hiérarchie basée sur la description de taxons et guidée par un critère d'information.

La notion de prédicteur de Gower [Gow74] s'inscrit dans le contexte de classification descriptive. À la suite de Gilmour [Gil51], une approche de la classification consiste à rechercher des classes en fonction de leur *pouvoir prédictif* : la connaissance *a priori* qu'un objet appartient à cette classe permet *a posteriori* de prédire des propriétés intéressantes de cet objet. Ainsi, selon ce principe, il est préférable de classifier des plantes à fleurs selon leur nombre d'étamines plutôt que selon leur nombre de feuilles, car, par exemple, une plante dont les fleurs ont trois étamines a fort vraisemblablement des feuilles à nervures parallèles, alors que très peu de propriétés sont corrélées au nombre de feuilles. Sous diverses formes, ce principe général est utilisé dans plusieurs applications de la classification à la biologie [Arc84, SH85]. Gower a proposé un sens opérationnel précis à ces recommandations générales ; il propose de mesurer la qualité d'une partition par le nombre de propriétés des objets correctement prédites par un prédicteur optimal affecté à chaque classe de la partition.

La classification conceptuelle relève d'une ambition plus grande que celle de simplement décrire des classes, puisqu'elle cherche à partager au mieux tout en assignant à chaque classe une combinaison de relations (conjonctions, disjonctions, etc) entre différentes variables, par exemple pour construire des hiérarchies [MS81] ou des partitions [MSE81]. Les techniques d'intelligence artificielle sont fortement utilisées dans cette voie [Fis87, FL86].

1.1.2 Formalisation

Nous utiliserons les notations suivantes [FB82, BF98] :

ω_i	objet
Ω	l'ensemble des objets
$\mathcal{P}(\Omega)$	l'ensemble des parties de Ω
$C \in \mathcal{P}(\Omega)\backslash\{\emptyset\}$	classe
$\mathcal{C} \subseteq \mathcal{P}(\Omega)\backslash\{\emptyset\}$	classification de Ω
$\mathbb{C}(\Omega) \subseteq \mathcal{P}(\mathcal{P}(\Omega)\backslash\{\emptyset\})$	ensemble des classifications *admissibles* de Ω
$p_{\mathcal{C}} : \mathcal{C} \longrightarrow \mathbb{R}$	évaluation (ou qualité) des classes de $\mathcal{C}$
$P : \mathbb{C}(\Omega) \longrightarrow \mathbb{R}$	évaluation des classifications de $\mathbb{C}(\Omega)$

On peut considérer le problème de classification comme un problème de maximimisation de P sur $\mathbb{C}(\Omega)$, dont on notera l'ensemble des solutions $\mathbb{M}(\Omega)$. Cette maximisation est faite selon un type de classification donné, tel que les partitions et les hiérarchies.

Définition. *Une classification* $\mathcal{C}$ *est une* **partition** *de* Ω *ssi :*
- $\cup_{C\in\mathcal{C}} = \Omega$
- $\forall C, C' \in \mathcal{C}, C \neq C' : C \cap C' = \emptyset$

On note $\mathbb{P}_k(\Omega)$ l'ensemble des k-partitions, c'est-à-dire des partitions en k classes de Ω ($|\mathcal{C}| = k$)[1], et celui des partitions de Ω, $\mathbb{P}(\Omega)$. De même, $\mathbb{M}_k(\Omega)$ est l'ensemble des k-partitions optimales

La recherche d'une partition optimale correspond à l'approche la plus naïve de ce qu'est une classification : ranger tous les éléments dans leur classe respective. Dans cette étude, nous nous intéresserons à des k-partitions construites selon un principe d'optimalité qui sera défini ultérieurement. En particulier, nous considérerons des **partitionnements**, c'est-à-dire des ensembles de k-partitions, pour k allant de 1 jusqu'à un nombre quelconque, éventuellement $|\Omega|$. Dans un partitionnement, une k-partition appartient au **niveau** k du partitionnement.

On peut vouloir gérer une « organisation » de partitions :

Définition. *Une classification* $\mathcal{C}$ *est une* **hiérarchie** *de* Ω *ssi :*
- $\Omega \in \mathcal{C}$
- $\forall \omega_i \in \Omega, \{\omega_i\} \in \mathcal{C}$
- $\forall C, C' \in \mathcal{C} : C \subseteq C'$ *ou* $C' \subseteq C$ *ou* $C \cap C' = \emptyset$

[1] $|A|$ est le cardinal de A.

On note $\mathbb{H}(\Omega)$ l'ensemble des hiérarchies de Ω. En pratique, une classification hiérarchique possède des classes qui sont l'union de quelques autres, et souvent même le type de hiérarchie considéré est celui de hiérarchie binaire, c'est-à-dire dans lequel toute classe non réduite à un élément est l'union de deux classes de la hiérarchie. Pour citer [LW66], le but d'une classification hiérarchique est de trouver la « meilleure » voie qui relie les individus isolés à la population dans sa globalité.

Une hiérarchie peut être vue comme un ensemble de partitions, ensemble dont les classes des éléments doivent être soit disjointes soient incluses entre elles ; ainsi, on voit le rapport étroit qui peut être fait entre ce type de classification et la notion de partitionnement.

1.1.3 Évaluations des classes et méthodes

Un processus classificatoire se définit par le type de classification ainsi que par le critère d'évaluation et la méthode de construction.

Souvent les fonctions d'évaluation de classification sont une somme des fonctions d'évaluation des classes.

$$P(\mathcal{C}) = \sum_{C \in \mathcal{C}} p_{\mathcal{C}}(C)$$

Pour simplifier, il est aussi possible de considérer que les évaluations des classes soient indépendantes des classifications, $\forall \mathcal{C} \in \mathbb{C}(\Omega) : p_{\mathcal{C}} = p$. Dans ce cas P sera dite **simple** :

$$P(\mathcal{C}) = \sum_{C \in \mathcal{C}} p(C)$$

Ce choix est très souvent fait, puisqu'il traduit que la valeur d'une classe ne dépend que d'elle-même, et non d'un contexte externe. C'est le cas que nous considérons.

1.1.3.1 Distance et dissimilarité

Pour classer des points d'un espace numérique, une classe peut être évaluée par la variance des distances entre les points y appartenant [Fis58]. Le travail consiste alors à miminiser la somme de ces variances au sein de l'ensemble des k-partitions. Sur la base du lemme d'additivité de Fisher [Fis58], Jensen [Jen68] propose un algorithme dynamique, toutefois très coûteux. Le même critère est aussi utilisé dans [ECS65] , cette fois pour construire une hiérarchie optimale, mais par une méthode toujours exponentielle selon le nombre d'objets.

Une extension de la distance consiste à munir l'ensemble des objets d'une fonction de dissimilarité :

Définition. *Une* **dissimilarité** *est une fonction* $d : (\Omega) \times \Omega \longrightarrow \mathbb{R}^+$ *telle que :*
- $\forall \omega_i, \omega_j \in \Omega : d(\omega_i, \omega_j) = d(\omega_j, \omega_i)$
- $\forall \omega_i \in \Omega : d(\omega_i, \omega_i) = 0$

À partir d'une dissimilarité, l'ensemble peut être considéré comme un graphe valué. Ainsi, par une méthode similaire à celle qui est présentée dans [GH61], le partage optimal en deux parties de Ω peut être obtenu avec une complexité temporelle en $O(|\Omega|^3)$. Le graphe des distances entre points d'un plan est une clique. D'une telle clique, Zahn [Zah71] opère un partage en deux, non optimal mais plus rapide, au moyen de l'arbre de recouvrement minimal. Hubert, dans [Hub74], expose différents rapports liant la théorie des graphes et la classification.

Pour construire des k-partitions optimales, Caliński et Harabasz [CH74] contruisent l'arbre de recouvrement minimal, qu'ils partagent ensuite en k classes de manière à minimiser les variations au sein des classes, par une méthode énumérative en $O(n^{k-1})$. Il est clair que construire une k-partition optimale dans un contexte encore aussi général est extrêmement coûteux, et il est plus facile de procéder de façon hiérarchique. Seulement, ceci ne garantit en général pas d'obtenir une k-partition optimale [CH74, p. 4]. Fisher et Van Ness [FVN71] décrivent les propriétés de plusieurs méthodes classificatoires face à la recherche de (k-)partitions optimales.

Deux stratégies importantes de classification hiérarchique sont la *division* et l'*agglomération*, mais Lance et Williams [LW65] reprochent à cette dernière de n'exploiter que des informations locales, et de ne pas utiliser l'information accessible en considérant la globalité de l'ensemble des objets.

Pour des raisons de complexité, il est également hors de question de construire une hiérarchie optimale dans le cas général, et les méthodes appliquées ne donnent que des résultats approchés. Dans [JS68, WLDC71] se trouvent des discussions à propos des propriétés que doit posséder un processus hiérarchique pour être acceptable. En plus de l'agglomération et de la division, d'autres méthodes, comme la greffe et l'échange, permettent de construire un arbre additif à partir d'une dissimilarité quelconque. Cependant, chaque méthode provoque un biais sur le type d'arbre obtenu [GA99], biais qui doit être pris en compte lors de l'interprétation de la classification.

1.1.3.2 Centroïde

Un autre point de vue consiste à chercher le **centroïde** d'une classe, c'est-à-dire un objet (éventuellement nouveau) qui minimise la somme des dissimilarités entre lui et les objets de la classe ; cette somme est une évaluation de la classe en question. Dans un contexte métrique, il s'agit du barycentre de la classe. Ceci est possible lorsque l'on dispose des valeurs des objets pour différentes variables et de la fonction effective de dissimilarité sur ces variables, et non seulement de la matrice des dissimilarités entre ces objets. Cette approche permet d'accéder à une description de chaque classe, aussi bien sur des données continues que qualitatives. À partir d'un centroïde, on peut aussi considérer la perte d'information (au sens de Shannon [Sha48] ou bien de Fisher [CG78]) occasionnée par le fait d'assimiler tous les objets de chaque classe au centroïde de celle-ci, comme un critère à minimiser.

1.1.3.3 Prédiction

Dans la logique d'au mieux construire des classes pour lesquelles on veut pouvoir disposer d'un résumé satisfaisant, la prédiction est une méthode plus proche du désir d'étiquetage et de description des classes que les approches précédentes.

On gère un ensemble fini D appelé ensemble de **prédicteurs admissibles**. Cet ensemble correspond aux descriptions selon lesquelles on veut construire des classes informatives.

Pour $\delta \in D$, on note $\pi_\delta : \Omega \longrightarrow \mathbb{R}$ la **prédiction** de δ sur les objets de Ω.

Par extension de π_δ sur les parties de Ω :

$$\begin{array}{rccl} \pi_\delta : & \mathcal{P}(\Omega) & \longrightarrow & \mathbb{R} \\ & A & \longmapsto & \sum_{\omega_i \in A} \pi_\delta(\omega_i) \end{array}$$

Pour la cohérence des notations, on voit que $\forall \omega_i \in \Omega, \pi_\delta(\{\omega_i\}) = \pi_\delta(\omega_i)$. Plus un prédicteur est adapté à la description de l'ensemble des objets de A, meilleure y est sa prédiction.

L'ensemble $\hat{\pi}_D(A)$ des **prédicteurs optimaux de** D d'une partie A est l'ensemble des prédicteurs de D dont la prédiction associée sur A est la plus élevée des descriptions possibles, c'est-à-dire :

$$\hat{\pi}_D(A) = \text{argmax}\{\pi_\delta(A) : \delta \in D\}$$

La prédiction optimale de A sur D est notée $\pi_D(A)$.

Au demeurant, une classe peut admettre plusieurs prédicteurs optimaux, si bien que la donnée seule de la classe ne suffit pas à savoir quel prédicteur a été choisi pour la classification, et toute interprétation ultérieure de cette classification peut en être gênée. Pour permettre de travailler sans ambiguïté sur une classe et *un* prédicteur optimal particulier, nous introduisons la notion de classe prédictive, c'est-à-dire de classe à laquelle est adjoint un prédicteur. Formellement, si D est l'ensemble des prédicteurs admissibles, une **classe prédictive** (C, δ) est un élément de $\{\mathcal{P}(\Omega)\backslash\{\emptyset\}\} \times D$. Comme le prédicteur d'une classe prédictive est fixé, la prédiction sur cette classe prédictive se définit d'elle même comme la prédiction de son prédicteur sur la classe.

Si C est une classe prédictive de prédicteur δ, on note $\pi_C = \pi_\delta$.

Une **classification prédictive**, notée $\mathcal{C}^D$, est un ensemble de classes prédictives. Les exemples de classifications donnés précédemment s'accordent au concept de classe prédictive. En particulier

Définition. *Une* k-**partition prédictive** $\mathcal{C}^D$ *de* Ω *sur l'ensemble de prédicteurs* D *est un élément de* $\mathcal{P}(\{\mathcal{P}(\Omega)\backslash\emptyset\} \times D)$ *de cardinal* k *tel que :*
- $\cup_{(C,\delta)\in\mathcal{C}^D} C = \Omega$
- $\forall (C, \delta), (C', \delta') \in \mathcal{C}^D : C \neq C' \Rightarrow C \cap C' = \emptyset$

On note $\mathbb{P}_k^D(\Omega)$ l'ensemble des k-partitions prédictives définies sur l'ensemble de prédicteurs D. De même, on définit $\mathbb{M}_k^D(\Omega)$ l'ensemble des k-partitions prédictives optimales de Ω basées sur l'ensemble de prédicteurs D. M_k^D est l'évaluation de ces optimaux.

Comme une partition prédictive $\mathcal{C}$ associe un prédicteur et un seul à chaque objet, on note $\hat{\pi}_{\mathcal{C}}(\omega_i)$ le prédicteur de ω_i dans $\mathcal{C}$ et $\pi_{\mathcal{C}}$ la fonction de prédiction définie par $\mathcal{C}$:

$$\begin{aligned} \forall \omega_i \in \Omega, \pi_{\mathcal{C}}(\omega_i) &= \pi_{\hat{\pi}_{\mathcal{C}}(\omega_i)}(\omega_i) \\ &= \pi_\delta(\omega_i) \text{ où } \omega_i \in C \text{ avec } (C,\delta) \in \mathcal{C} \end{aligned}$$

La qualité d'une partition est évaluée par la somme des prédictions que l'on peut faire sur tous les objets en utilisant pour chaque classe son prédicteur optimal :

$$\forall \mathcal{C} \in \mathbb{P}_k(\Omega) : \pi_D(\mathcal{C}) = \sum_{C \in \mathcal{C}} \pi_D(C)$$

Ceci mesure l'homogénéité à l'intérieur des classes qui constituent la partition vis-à-vis des descriptions choisies.

L'objectif de la classification ici définie est de trouver les k-partitions qui maximisent ce critère de prédiction.

Cette notion de prédiction sur une partition s'adapte à celle de prédiction sur une partition prédictive. La prédiction sur une partition prédictive est la somme des prédictions sur les classes prédictives qui la composent :

$$\forall \mathcal{C}^D \in \mathbb{P}_k^D(\Omega) : \pi(\mathcal{C}^D) = \sum_{(C,\delta) \in \mathcal{C}^D} \pi_\delta(C)$$

Cette approche ne mesure plus l'homogénéité des classes qui constituent la partition, mais seulement la bonne adéquation des prédicteurs aux classes prédictives. En revanche, si l'objectif est de construire les partitions prédictives définies sur l'ensemble de prédicteurs D qui maximisent ce critère, cette approche rejoint la précédente. En effet, entre deux classes prédictives définies sur D, (C, δ) et (C, δ'), où δ' est un prédicteur optimal de C dans D, par définition $\pi_\delta(C) \leqslant \pi_{\delta'}(C)$. Ainsi, les classes prédictives d'une partition prédictive optimale ont toujours un prédicteur optimal qui leur est attribué.

$$\forall \mathcal{C}^D \in \mathbb{M}_k^D(\Omega), \forall (C,\delta) \in \mathcal{C}^D : \delta \in \hat{\pi}_D(C)$$

Comme, par définition de π_D, si $\delta \in \hat{\pi}_D(C)$ alors $\pi_\delta(C) = \pi_D(C)$, on obtient

$$\forall \mathcal{C}^D \in \mathbb{M}_k^D(\Omega), \pi(\mathcal{C}^D) = \sum_{(C,\delta) \in \mathcal{C}^D} \pi_D(C)$$

Le **partitionnement maximalement prédictif** est un ensemble de k-partitions prédictives optimales, pour k entre 1 et une valeur quelconque.

Dans [Gow74], Gower définit la notion de prédiction sur des données qualitatives. L'ensemble des prédicteurs admissibles est l'ensemble des attributs que peut avoir un objet, et la prédiction d'un prédicteur sur un objet est le nombre d'attributs partagés par le prédicteur et l'objet. Par notre définition, nous pouvons considérer d'autres contextes. Par exemple, si les objets de Ω sont des points d'un espace, les prédicteurs de D correspondent à des parties de cet espace. La fonction de prédiction de $\delta \in D$ est en quelque sorte la fonction d'appartenance à une partie de cet espace, partie décrite par δ. Cette fonction peut être simplement une fonction binaire, mais aussi continue.

En opposition à l'homogénéité définie à partir d'un ensemble de prédicteurs D, une notion d'hétérogénéité peut être définie à partir de ce même ensemble. Ainsi, GOWER [GOW74] gère en outre un critère de séparation des classes par le fait que le prédicteur d'une classe doit au plus mal prédire les objets des autres classes.

Ici encore, la complexité de l'espace de recherche est beaucoup trop grande pour espérer, si le nombre d'objets est important, pouvoir construire des k-partitions optimales.

En poussant plus loin la logique de représentation d'une classe, on aborde le domaine de la classification conceptuelle. Dans [MSE81, FIS87], les concepts se trouvent au centre du travail (aussi bien par la méthode que pour l'évaluation). Chaque classe est définie par un concept, et les objets sont classés selon le concept qui les décrit au mieux. Ceci complique la problématique par rapport au cadre de la prédiction, auquel cette étude est limitée.

1.2 De la classification sous contraintes

Jusqu'à présent, hormis le type de classification choisi, l'ensemble des classes admissibles est totalement libre. Cependant, il est possible que ce ne soit pas le cas, par décision de l'utilisateur, par la nature du problème ou bien de l'ensemble des données. Parfois, cela permet de travailler dans un espace de recherche plus maniable – plus petit, mieux structuré – et donc d'élaborer des algorithmes moins complexes.

On appelle **classification sous contraintes** une recherche de classification dans laquelle des contraintes limitent l'ensemble $\mathbb{C}(\Omega)$ des classifications admissibles.

1.2.1 Approche générale

Choisir un type de classification *a priori* peut déjà être vu comme une contrainte, décidée par l'utilisateur selon ses besoins. Mais les contraintes possibles sont innombrables [DM84, GOR96C, BF98]. L'essentiel est de bien savoir si elles nuisent à la qualité du résultat par rappport à l'objectif premier ou bien si elles sont « naturelles » vis-à-vis du problème. On peut ainsi faire la différence entre **contraintes internes**, qui sont inhérentes au problème, obligatoires et irréprochables, et les **contraintes externes**, ajoutées à la problématique initiale. Néanmoins, une contrainte peut être considérée comme interne dans un contexte précis, et externe dans un autre, alors que le problème purement de classification est dans les deux cas identique. Ainsi, par exemple, on peut de force munir Ω d'une structure, quitte à estimer cette structure [HS76], ou bien utiliser des données structurelles naturelles de Ω, comme une structure d'ordre, cyclique, de plan, etc. Un type de contrainte est, si Ω est muni d'une structure, de limiter la forme des classes. Ainsi, par exemple, dans un espace affine, on peut ne chercher que des classes rondes, carrées, convexes... ou encore connexes. Nous étudierons ce dernier cas plus loin.

Lorsque l'on estime disposer d'un espace de recherche correct pour pouvoir procéder à un processus classificatoire, les méthodes générales peuvent être appliquées, peut-être sous certaines conditions. Mais dans certains cas, les contraintes sont telles que de nouvelles méthodes doivent être utilisées. Ainsi, ce n'est pas parce qu'un espace de recherche est plus petit qu'il est plus facile d'y trouver une solution. En effet, sa géométrie peut interdire certains algorithmes qui étaient utilisables sur un plus grand espace. Par exemple, pour effectuer une dichotomie sur un graphe à arêtes valuées, un algorithme optimal proposé dans [GH61] a une complexité temporelle en $O(|\Omega|^3)$, alors qu'avec une contrainte sur la taille des classes, un algorithme a dû être conçu [CB76] dont la complexité est en moyenne $O(|\Omega|^{4,9})$.

1.2.2 Contraintes de contiguïté

Les contraintes de contiguïté sont très importantes, aussi bien d'un point de vue théorique que pratique. Soit $\mathcal{T}$ une topologie sur Ω, une classification qui respecte les contraintes de contiguïté sur Ω se définit dans :

$$\mathbb{C}^{\mathcal{T}}(\Omega) = \{\mathcal{C} \subseteq \mathcal{P}(\Omega) : \forall C \in \mathcal{C}, C \text{ est connexe selon } \mathcal{T}\}$$

Ainsi, on peut définir une relation de voisinage sur Ω à partir de $\mathcal{T}$ et, comme on travaille en pratique sur des ensembles finis, cette relation est représentée par un graphe. Sur certains ensembles, comme par exemple des points du plan [GS69, GF85, GOR96C], il n'est pas forcément trivial de définir une relation de voisinage.

Même sous contrainte de contiguïté, il est encore généralement difficile de construire des classifications optimales ; mais aussi bien sur des plans [NG80] que sur des graphes, des algorithmes « rapides » presque optimaux de partitions ou de hiérarchies existent. CASELDINE et GORDON [CG78] proposent une méthode de construction de hiérarchies sur des arbres.

Certaines conditions doivent être satisfaites pour que les procédures de classification dans un cadre général soient applicables dans un contexte de contiguïté [FB82, MUR85, GOR96C]. À l'inverse, parfois, la fonction de dissimilarité est telle que l'on peut sans restriction ramener le contexte initial du problème à un contexte contraint, pour pouvoir ainsi utiliser des méthodes plus efficaces [HWA81, HSY85, BKČK94]. Une autre approche consiste à construire, en classant les objets, une structure sur l'ensemble. Ainsi, par exemple, la classification pyramidale, qui est une généralisation de la hiérarchie :

Définition. *Une classification $\mathcal{C}$ est une* **pyramide** *de Ω ssi :*
- $\Omega \in \mathcal{C}$
- $\forall \omega_i \in \Omega, \{\omega_i\} \in \mathcal{C}$
- $\forall C, C' \in \mathcal{C} : C \cap C' = \emptyset$ *ou* $C \cap C' \in \mathcal{C}$
- *Il existe un ordre sur Ω tel que toute classe de $\mathcal{C}$ est un intervalle.*

Dans [BD90] est présenté un algorithme – en $o(|\Omega|^4)$ – qui construit une classification pyramidale d'un ensemble d'objets à partir d'une fonction de dissimilarité, tout en établissant un ordre total sur Ω. Mais celui-ci n'est pas fixé *a priori*; il est découvert au fur et à mesure de la classification. Ainsi, une structure séquentielle sur l'ensemble et une classification dont toutes les classes sont contiguës selon cette séquence sont construites simultanément.

En classification sous contraintes, intuitivement, le problème n'est plus simplement de regrouper des objets selon un critère de rapprochement, mais plutôt de trouver des *zones*[2] dont les objets respectent au mieux ce critère. Cependant, dans une zone, peuvent être « coincés » des objets qui s'écartent fortement de ce critère par rapport à la majorité des points de cette zone. Aussi apparaît une différence fondamentale avec les classifications qui ne sont pas soumises à une contrainte de contiguïté. La classification pratique un **lissage** de l'ensemble des objets, cherchant à construire des zones les plus lisses possibles, selon un critère de lissage bien défini (l'évaluation des classes), et organisées de manière bien déterminée (le type de classification). La classification est alors une approche particulière de traitement d'images [HD75, NG80] ou de signaux. Ce dernier correspond à la classification sous contrainte d'ordre total.

1.2.2.1 Contrainte d'ordre total

Nous appelons **contrainte d'ordre total** la contrainte selon laquelle, si Ω est une séquence déjà fixée, les classes contiguës à former sont des segments. Nous appellerons **césures** les positions où les classes se rejoignent. En spécialisant le lemme de Fisher [Fis58], la construction de k-partitions optimales sur un critère d'évaluation simple est faisable par un algorithme dont la complexité en temps est $O(k.|\Omega|^2)$ [HM73, BW77, LV94, Gor96c]. Cet algorithme dynamique sera présenté dans le chapitre suivant (page 36).

Comme une classification vise à classer des objets, normalement, une classe non homogène devrait pouvoir être partagée en deux classes de façon profitable. Ce principe est au cœur des méthodes de construction hiérarchiques par dichotomie. Désormais, ce gain n'est plus garanti. Dans [GB72, GB74], des comparaisons de résultat sont faites entre hiérarchie et partition, pour évaluer avec quelle précision une hiérarchie « correcte » approche une partition optimale sur des données séquentielles. Un algorithme optimal de classification hiérarchique sur des séquences est présenté dans [LV95] ; il est valable avec les critères de classification simples, et sa complexité temporelle est en $O(|\Omega|^3)$ et spatiale en $O(|\Omega|^2)$. Dans [LDL85] est présentée une méthode de partage d'une séquence chronologique qui est une modification d'une méthode de construction hiérarchique.

Par des méthodes de calcul numérique, Witkin [Wit83] construit une hiérarchie sur un signal numérique, chaque classe de cette hiérarchie étant décrite par une fonction parabolique. Churchill [Chu89, Chu92] partage une séquence de lettres grâce à des chaînes de Markov à états cachés. Chaque classe est décrite par un état, et à chaque état correspond un processus markovien (pour plus de détails, voir 1.3.2.3).

[2] ou encore des classes connexes, donc géométriquement reconnaissables

1.2.3 Étiquetage sous contrainte

L'intérêt d'une description d'une classe est aussi clair dans un contexte contraint que dans le contexte général, et les différentes notions concernant la description et la prédiction, valables dans un contexte général, le demeurent dans un contexte contraint.

Dans les cas de [Wit83, Chu92], la classification a permis d'extraire ce que nous appelons des **descripteurs structurés de classes**, c'est-à-dire des descripteurs qui sont des modèles de reconstruction de leurs classes, ces modèles étant possibles grâce à la structure séquentielle des données. Ainsi, la description d'une classe n'est pas simplement une même description de tous les objets de la classe. L'information fournie par une telle description n'est pas seulement la composition de la classe, mais une approche de la structure interne de la classe, en fonction de l'adéquation du descripteur.

À l'échelle d'un objet, comme l'ensemble est muni d'une structure, la description d'un objet peut tenir compte du voisinage de cet objet. Par exemple, dans une ville, on peut vouloir classer des gens en fonction de leurs professions et celles de leurs voisins. Dans ce cas, la description d'une classe ne donnera pas seulement une description de chaque personne en tant que telle, mais du contexte selon lequel cette personne a été classée, c'est-à-dire la profession de ses voisins. Et à l'inverse, la seule connaissance de la profession d'une personne n'explique en rien pourquoi elle a été classée dans telle ou telle classe. On parle alors de **descripteur contextuel**. Ce type de descripteur est très différent de celui de descripteur structuré, puisque tous les objets d'une même classe sont décrits de manière identique. De plus, cette notion de description est totalement indépendante de celle de contrainte de contiguïté des classes. La structure est juste une information qui sert à la classification, mais qui ne la contraint en rien.

1.2.3.1 Prédiction sur séquences

Dans le cas de la classification prédictive, la même approche que dans le cas général reste valable. Lorsque chaque classe est décrite par un seul prédicteur, auquel tout objet de la classe est comparé, on retrouve le principe de construction de classes homogènes vis-à-vis d'un ensemble de prédicteurs.

Ainsi, par exemple, si Ω est une séquence dans un espace affine et si on assimile la prédiction de $\delta \in D$ à une fonction d'appartenance à une partie de cet espace décrite par δ, construire une k-partition maximalement prédictive sur cette séquence signifie la partager en k segments tels que chacun appartienne au mieux à une des parties de cet espace décrites par les prédicteurs de D.

Dans une partition prédictive, deux classes voisines peuvent avoir la même fonction de prédiction. Nous appelons **propre** une partition prédictive dont les fonctions de prédiction sont différentes entre classes voisines.

L'introduction d'une structure sur l'ensemble Ω permet de considérer des **prédicteurs contextuels** qui fournissent aux classes prédites des descripteurs contextuels. Pour cela, on étend la notion de prédicteur d'objet définie dans la section 1.1.3.3. Si Ω est une séquence $(\omega_1, \omega_2, \dots, \omega_n)$, la prédiction de ω_i peut dépendre des prédictions sur les autres ω_j. Par simplicité, nous nous limiterons aux successeurs de ω_i dans la séquence, et on définit la fonction de prédiction d'un prédicteur contextuel de la façon suivante.

Si $(\delta_1, \dots, \delta_c)$ est ce prédicteur avec $\delta_j \in D$, sa prédiction en ω_i est nulle si au moins un des δ_j avec $j > 2$ prédit « mal » l'objet ω_{i+j-1}, c'est-à-dire que la prédiction de δ_j sur ω_{i+j-1} n'est pas strictement positive. Par cohérence avec la prédiction des prédicteurs non contextuels, la prédiction en ω_i de δ_1 est utilisée comme précédemment. Ainsi [3] :

$$\pi_{(\delta_1,\dots,\delta_c)}(\omega_i) = \pi_{\delta_1}(\omega_i).\mathbb{1}_{(\pi_{\delta_2}(\omega_{i+1})>0)\wedge(\pi_{\delta_3}(\omega_{i+2})>0)\wedge\ldots\wedge(\pi_{\delta_c}(\omega_{i+c-1})>0)}$$

Si le prédicteur contextuel est trop long $(i + c - 1 > n)$, la prédiction est nulle.

Comme cette définition correspond aussi aux prédicteurs non contextuels, nous inclurons désormais l'ensemble des prédicteurs contextuels que l'on veut considérer dans l'ensemble des prédicteurs admissibles D.

Mais on peut aussi concevoir que, sous contraintes internes, la description de la classe soit une fonction liée à un **prédicteur structuré**, au même sens que descripteur structuré. Chaque objet, selon sa *position* dans la structure (contrainte initiale), est comparé à un prédicteur particulier produit par cette fonction [GV99]. Là aussi nous considérerons un cas simple de prédicteur structuré, à savoir les motifs de prédicteurs.

Soient des prédicteurs $\delta_1, \delta_2, \dots, \delta_c$ de D. On considère le mot $\delta_1\delta_2\ldots\delta_c$ pour prédire la classe $C = \{\omega_i, \dots, \omega_j\}$. ω_i est prédit par δ_1, ω_{i+1} par δ_2, et ainsi de suite ω_{i+c-1} est prédit par δ_c ; puis on recommence en itérant le motif sur la longueur de la classe, ω_{i+c} est prédit par δ_1, etc.

On considère alors la classe prédictive $(C, \delta_1\delta_2\ldots\delta_c)$. Si la classe est un singleton, la prédiction par un prédicteur structuré y est la même que par son premier prédicteur simple. Ainsi, on peut sans problème considérer la prédiction d'un objet par un prédicteur structuré.

On considère l'opérateur de permutation circulaire ρ tel que

$$\rho(a_1a_2a_3\ldots a_p) = a_pa_1a_2\ldots a_{p-1}$$

En fin de compte, la classe prédictive $C = (\{\omega_i, \ldots \omega_j\}, \delta_1\delta_2\ldots\delta_c)$ a une prédiction :

$$\pi_C = \sum_{i \leqslant s \leqslant j} \pi_{\rho^{i-s}(\delta_1\delta_2\ldots\delta_c)}(\omega_s)$$

Si, au lieu de considérer les positions relatives des objets d'une classe, on veut en considérer les positions absolues, on utilise la notation $(C, \underline{\delta})$. Ceci signifie que δ est « calé » de telle sorte que le premier prédicteur simple du motif δ est mis en phase avec ω_1. Si $(C, \delta) = (C, \underline{\delta'})$ et si le premier objet de C est ω_i, $\delta = \rho^{i-1}(\delta')$.

Nous venons ainsi de définir dans le contexte de la classification le partitionnement maximalement prédictif sous contrainte d'ordre total. Un domaine

[3] $\mathbb{1}_{\text{vrai}} = 1$ et $\mathbb{1}_{\text{faux}} = 0$.

d'application de ce partitionnement concerne le partage de séquences de lettres en segments homogènes. En effet, si une séquence est vue comme un mot sur un alphabet, on peut considérer que cet alphabet est l'ensemble des prédicteurs. Par exemple, pour chaque prédicteur, la prédiction en une position de la séquence est la fonction caractéristique, qui vaut 1 si la lettre en cette position et le prédicteur sont semblables, 0 sinon. Dans ce cas, une k-partition optimale partage la séquence en k segments les plus homogènes possibles vis-à-vis de l'occurrence d'une même lettre dans chacun de ces segments.

Dans ce cas, la prédiction par un prédicteur contextuel en une position de la séquence signifie que le mot qui correspond à ce prédicteur est présent en cette position. On peut ainsi considérer une partition selon les occurrences de tels ou tels mots dans la séquence. La prédiction par un prédicteur structuré sur une classe correspond à l'itération du motif qui correspond à ce prédicteur sur la classe. On peut ainsi considérer une partition selon la bonne description de ces classes par des motifs.

De ce point de vue, cette méthode fait partie de celles qui cherchent à caractériser au mieux une séquence de lettres par d'autres séquences, ou par des opérations sur des mots. Nous présentons ici quelques exemples de ces méthodes. Un exposé détaillé des méthodes de comparaison sur des séquences est fourni dans la thèse de Marie-France Sagot [Sag96].

1.3 De la redescription de séquences de lettres

1.3.1 De l'alignement de séquences

Aligner au mieux des séquences est important en analyse de séquences génétiques et protéiques, car la découverte de propriétés d'une séquence peut se faire en la comparant à d'autres séquences assez proches. En général, la proximité entre deux séquences ou au sein d'un groupe de séquences est définie par la facilité de faire correspondre tout ou partie des lettres qui composent les différentes séquences, et ainsi d'établir des relations entre ces séquences. Un ensemble de séquences peut en outre être muni d'une structure, par exemple d'arbre [San75, SCL76], qui contraint les relations possibles entre ces séquences.

Nous verrons une méthode optimale issue de la programmation dynamique pour aligner globalement deux séquences, méthode dont la complexité en temps est linéaire avec le produit des longueurs des deux séquences. Cette méthode peut être étendue à l'alignement multiple de plusieurs séquences, mais la complexité est alors trop importante. Pour aligner de nombreuses séquences, par exemple au sein d'une banque de données, on fait alors appel à des méthodes heuristiques ou recherchant des alignements locaux, et qui s'inspirent aussi des méthodes de programmation dynamique.

1.3.1.1 Méthodes d'alignement global

Soient $\Omega_1 = (\omega_{1,1}, \ldots, \omega_{1,n})$ et $\Omega_2 = (\omega_{2,1}, \ldots, \omega_{2,m})$ deux séquences d'objets. L'ensemble de tous les objets est muni d'une mesure de similarité s. Le but est d'aligner au mieux Ω_1 en face de Ω_2 de manière à maximiser la somme des similarités entre les objets qui se trouvent en vis-à-vis dans l'alignement. En outre, on permet qu'il y ait des trous, notés `-`, dans l'alignement. Par exemple, comme alignement entre `ATACCGAC` et `TTCATCCGTTACG`, on peut avoir :

```
---ATACCG--AC-
TTCAT-CCGTTACG
```

Opérer une insertion, c'est-à-dire faire un trou dans une séquence, doit avoir un certain coût, que l'on note c, et que l'on peut aussi considérer comme une extension de la similarité s entre n'importe quel objet et un objet fictif figuré par le tiret. Par cohérence, la similarité entre deux objets figurés par des tirets est $2c$, puisqu'il y a alors en tout deux insertions pratiquées. Il est sensé de pénaliser l'insertion quand elle n'est pas nécessaire, et ainsi c est négatif.

Opérer un alignement global entre Ω_1 et Ω_2 revient à opérer plusieurs insertions sur chaque séquence, de manière à construire deux nouvelles séquences $\Omega'_1 = (\omega'_{1,1}, \dots, \omega'_{1,p})$ et $\Omega'_2 = (\omega'_{2,1}, \dots, \omega'_{2,p})$ $(p \geq n, m)$ de même longueur. La valeur de cet alignement est :

$$\mathcal{S}(\Omega'_1, \Omega'_2) = \sum_{1 \leq i \leq p} s(\omega'_{1,i}, \omega'_{2,i})$$

Construire un alignement global optimal revient à maximiser $\mathcal{S}$ sur l'ensemble des alignements possibles.

Par programmation dynamique [WF74], il est possible de construire l'alignement global optimal. Le principe est de lire en même temps les deux séquences, mais parfois la lecture sur une séquence est mise en « attente » pendant que la lecture sur l'autre séquence est poursuivie, et inversement. Ceci a pour conséquence d'insérer des trous dans l'une et l'autre séquence. Grâce à un tableau $\mathcal{T}$ de $n \times m$ valeurs, toutes les lectures simultanées possibles sont modélisées. La valeur $\mathcal{T}(i, j)$ est le coût de l'alignement optimal entre $(\omega_{1,1}, \dots, \omega_{1,i})$ et $(\omega_{2,1}, \dots, \omega_{2,j})$. Cet alignement peut provenir de trois alignements possibles :

- $(\omega_{1,1}, \dots, \omega_{1,i-1})$ et $(\omega_{2,1}, \dots, \omega_{2,j-1})$, auquel on ajoute celui entre $\omega_{1,i}$ et $\omega_{2,j}$, auquel cas :
$$T(i, j) = T(i-1, j-1) + s(\omega_{1,i}, \omega_{2,j})$$
- $(\omega_{1,1}, \dots, \omega_{1,i})$ et $(\omega_{2,1}, \dots, \omega_{2,j-1})$, auquel on ajoute une insertion en face de $\omega_{2,j}$, auquel cas :
$$T(i, j) = T(i, j-1) + c$$
- $(\omega_{1,1}, \dots, \omega_{1,i-1})$ et $(\omega_{2,1}, \dots, \omega_{2,j})$, auquel on ajoute une insertion en face de $\omega_{1,i}$, auquel cas :
$$T(i, j) = T(i-1, j) + c$$

Si bien qu'en résumé, en prenant le maximum :

$$T(i, j) = \max\left\{T(i-1, j) + c, T(i, j-1) + c, T(i-1, j-1) + s(\omega_{1,i}, \omega_{2,j})\right\}$$

Pour les valeurs initiales de la récurrence, que la position i d'une séquence soit alignée juste avant la position 1 de l'autre signifie que i insertions ont été faites, si bien que :

$$T(i,0) = i.c \text{ et } T(0,j) = j.c$$

L'alignement global optimal a pour coût $T(n,m)$, et le parcours en sens inverse des arguments des maximaux qui ont servi à calculer $T(n,m)$ fournit un alignement global optimal. Dans le cas où il y a plusieurs alignements globaux optimaux, ils sont de la même manière obtenus en suivant tous les parcours qui ont servi à calculer $T(n,m)$.

On peut vérifier que, par cet algorithme, l'alignement global optimal entre Ω_1 et Ω_2 est effectué en un temps $O(n.m)$. Via des évaluations de borne supérieure et inférieure du coût de l'alignement, il est possible de limiter la complexité en restreignant la part utile du tableau [Fic84, Spo91]. Pour les cas où il n'est pas souhaité de seulement trouver l'alignement optimal, des méthodes sous-optimales ont été proposées dans [Wat83, Zuk91].

Il existe d'autres mesures de l'alignement optimal. En particulier, dans un contexte biologique, une mesure d'homologie est associée aux paires de séquences. Cette mesure dépend de caratéristiques biologiques et peut, dans certains cas, être ramenée à celle de similarité entre séquences définie ci-dessus. Mais ce n'est pas toujours le cas, comme par exemple si l'homologie entre deux séquences est définie par le nombre d'évènements mutationnels nécessaires pour passer de l'une à l'autre. Une autre mesure d'homologie peut se baser sur une mesure de similarité entre lettres, mais le coût d'un trou dans une séquence ne dépend pas de la taille de celui-ci. Dans ce cas, une adaptation de l'algorithme dynamique ci-dessus peut être trouvée dans [MM89]. Elle consiste à tripler le tableau, chaque case étant triplée selon la façon par laquelle l'alignement est parvenu en cette case : par alignement de deux objets ou par insertion sur l'une ou l'autre séquence.

Dans le cas de l'alignement conjoint de k séquences, un tableau basé sur le même principe que celui exposé ci-dessus a k dimensions, et chaque case a alors $2^k - 1$ cases adjacentes, ce qui rend vite l'algorithme dynamique inapplicable. Ainsi, des méthodes sous-optimales ont été développées, qui se basent sur des approximations d'alignements globaux, sur des alignements locaux de séquences, ou sur la description de séquences par d'autres types d'objets.

La plupart des méthodes heuristiques développées cherchent à limiter l'espace de recherche par des choix liés au domaine d'application et par des choix algorithmiques. Par exemple, la connaissance ou la construction d'un arbre qui établit des liens entre les séquences permet d'accélérer cet alignement en cherchant à minimiser les transformations entre séquences reliées, de construire des séquences consensuelles, ou encore de faire les alignements progressivement séquence par séquence [WP84, HBF92].

1.3.1.2 Méthodes d'alignement local

Aligner deux séquences de façon globale s'intéresse à toutes les lettres de ces deux séquences. Un tel alignement est intéressant lorsque l'on cherche la correspondance entre deux séquences *entières*. Mais s'il s'agit seulement de chercher à faire correspondre au mieux deux parties quelconques de séquences, considérer aussi toutes les différences qui se situeront hors de ces parties risque de perturber la bonne correspondance. Un alignement local, en revanche, cherche à aligner au mieux deux sous-séquences sans tenir compte de ce qui se passe en dehors de celles-ci. Un alignement local optimal entre deux séquences correspond à trouver les parties de ces séquences qui s'alignent au mieux. Il est clair que si tous les objets différents ont une similarité positive, un alignement local gagnera toujours à être étendu à n'importe quelle paire d'objets, si bien qu'il sera en fait global, hormis aux extrémités des séquences. Ainsi, la notion d'alignement local optimal n'a d'intérêt particulier que si, pour certains couples de lettres différentes, la similarité est négative.

Par une adaptation de l'algorithme dynamique présenté ci-dessus, Smith et Waterman ont construit un algorithme qui réalise l'alignement local optimal entre deux séquences [SW81]. Les notations sont les mêmes que ci-dessus, et les formules de récurrence sont, pour tout i et j :

$$T(i,0) = 0$$
$$T(0,j) = 0$$
$$T(i,j) = \max\left\{0, T(i-1,j-1) + s(\omega_i,\omega_j), T(i-1,j) + c, T(i,j-1) + c\right\}$$

L'optimal est alors réalisé en $\operatorname{argmax}\{T(i,j) : 1 \leqslant i \leqslant n, 1 \leqslant j \leqslant m\}$. L'alignement local optimal qui correspond à cette case du tableau est celui qui a permis d'atteindre cette case depuis la dernière case de valeur 0. La complexité temporelle de cet algorithme est aussi $O(m.n)$, et il est clair que celle-ci est exponentielle avec le nombre de séquences. Et même en se contentant de l'alignement d'une séquence donnée de taille n sur un ensemble de M séquences de taille moyenne m, la complexité $O(n.m.M)$ devient vite trop importante.

Des méthodes approchées ont été développées pour permettre l'alignement local de plusieurs séquences. En particulier, lorsqu'il y a énormément de séquences à comparer, comme dans les banques de données biologiques, des méthodes appelées de criblage de banque extraient, à partir d'une séquence requête, toutes les séquences de la banque qui satisfont un certain critère de ressemblance locale.

Parmi ces méthodes, il y a celles qui se basent sur l'alignement deux à deux, et qui s'adaptent simplement à l'alignement multiple, comme le comptage de k-mots (ou mots de longueur k), développé dans les programmes FastN ou FastP[4], améliorés en FastA [Pea90, Pea91], ou comme l'algorithme du programme BLAST [AGM+90]. Étant donnée une longueur de mot k, la méthode des Fast consiste, entre la séquence requête et chaque séquence de la banque, à faire glisser l'une sur l'autre de manière à sélectionner les 10 alignements locaux sans insertion les plus similaires qui se terminent de façon identique sur des k-mots et qui ne dépassent pas un seuil de lettres différentes. Ces alignements sont ensuite évalués au moyen d'une matrice de similarité. FastA cherche en outre à joindre les alignements ainsi trouvés par des grandes zones d'insertion sur l'une

[4]FastN est le programme qui concerne les séquences génétiques, et FastP celui qui concerne les protéines.

ou l'autre des séquences. Le principe de BLAST consiste, étant donnée une longueur de mots k, à distinguer les k-mots les plus fréquents de la séquence requête et de répertorier dans l'ensemble des séquences de la banque les positions des k-mots assez similaires à ceux qui ont été distingués dans la séquence requête. Ensuite, en chaque position ainsi trouvée, le programme cherche à prolonger les alignements de manière à satisfaire au moins un certain score. Ces méthodes sont aussi au pire en $O(n.m.M)$ mais en pratique bien plus rapides que l'algorithme de programmation dynamique optimal.

Il existe aussi des méthodes approchées pour aligner directement un ensemble des séquences. Ces méthodes réduisent la complexité de l'espace de recherche par différents moyens, comme la recherche d'alignements sans insertion, la contrainte d'une taille limite des alignements locaux recherchés, ou encore le regroupement de plusieurs séquences selon leur similarité. En particulier, l'algorithme heuristique selon lequel est conçu le programme MACAW [SAL91] parvient à réaliser de façon efficace des alignements locaux sans insertion dans un ensemble de N séquences de taille moyenne n avec une complexité temporelle en $O(N^2.n^2)$. Son principe est de pratiquer des alignements par la méthode FastA entre toutes les paires de séquences, puis une recherche similaire à celle de BLAST sur l'ensemble de la banque de séquences, mais limitée à l'ensemble des mots trouvés pendant la première phase, ce qui limite considérablement la recherche.

Une autre technique pour comparer différentes séquences est de les décrire autrement puis d'utiliser cette description. Il existe d'autres procédés de redécrire une séquence de lettres qu'en s'intéressant spécifiquement aux mots et motifs qui la constituent. En particulier, l'adaptation de méthodes d'analyse de signaux numériques telles que les transformations de Fourier [McL77, LRS+86, Oll90], les ondelettes [AdCB+96], des fonctions d'auto-corrélation [Oll90, p 48], ont donné des résultats intéressants en analyse de séquences biologiques, mais comme ces méthodes sont assez éloignées de notre approche, nous n'en parlerons pas ici.

1.3.2 Autres redescriptions de séquences

1.3.2.1 Recherche de grands mots

Faire apparaître les plus grands mots répétés dans une séquence de lettres est intéressant, non seulement parce que cela permet d'en faire correspondre différentes parties similaires et de la modéliser par ses répétitions, mais aussi parce que ces mots peuvent apparaître dans d'autres séquences et ainsi une classification des séquences selon leurs grands mots peut être faite. Le passage de la recherche de répétitions de grands mots dans une seule séquence à celle d'occurences multiples de grands mots dans différentes séquences se fait naturellement en faisant la recherche sur une seule séquence constituée de toutes ces séquences concaténées, à condition d'interdire après coup les mots qui sont construits à la jonction de deux séquences.

Pour cela, une méthode consiste à regrouper les positions de la séquence en classes dans lesquelles les positions correspondent au même mot de longueur 1.

Ensuite, chaque nouvelle classe est partagée selon les mots bilitères qui commencent en ses positions. Ainsi de suite, pour k donné, on regroupe les positions selon les mots le longueur k. Par une méthode particulière due à Karp, Miller et Rosenberg (dite méthode KMR), ceci peut être fait en un temps $O(|\Omega|.\log k)$. S'il est gênant de fixer par avance une longueur de mot, il est possible avec la même complexité de trouver les plus grands mots dont le nombre d'occurences dans la séquence est supérieur à une valeur voulue.

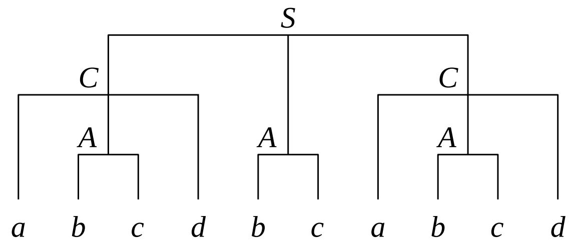
1.3.2.2 Règles de réécriture

Dans [NMW97], une séquence est décomposée selon des règles de réécriture récursives. En même temps qu'elle est lue, une séquence voit ses répétitions de mots réduites à des caractères qui décrivent, chacun, une règle de réécriture. Une séquence est ainsi réduite par une grammaire qui décompose de façon hiérarchique la séquence. Par exemple, sur la séquence :

$$abcdbcabcd$$

lue de gauche à droite, les règles de réécriture obtenues et la hiérarchie sont :

$$\begin{array}{ll} A & \rightarrow bc \\ C & \rightarrow aAd \\ S & \rightarrow CAC \end{array}$$



Cette approche peut être adaptée à la description d'une séquence à partir de règles établies par ce processus sur une autre séquence (ou sur un ensemble de séquences) pour inférer une structure à partir de règles préétablies.

1.3.2.3 Chaînes de Markov à états cachés

L'utilisation de chaînes de Markov à états cachés s'inscrit dans un cadre de modélisation probabiliste de séquences de lettres.

On veut décrire la séquence par une suite d'états probabilistes, lesquels sont liés entre eux par une chaîne de Markov. Si on note c la suite d'états – ou chemin – avec c_i le i^e état de ce chemin, c'est-à-dire l'état à la i^e position de la séquence, et $((a_{kl}))_{k,l}$ la matrice correspondant à la chaîne de Markov étudiée, les états successifs de c obéissent à la formule

$$\forall k, l : a_{kl} = P(c_i = l | c_{i-1} = k)$$

avec P désignant la probabilité.

Au sein de chaque état du processus markovien, les lettres suivent une certaine loi de probabilité qui peut correspondre à un processus markovien sur les lettres de la séquence. Elle peut aussi être plus élaborée, et suivre par exemple un processus markovien sur les mots de la séquence. Dans un cas plus simple, chaque état associe un facteur de probabilité à chaque lettre. Par souci de clarté, nous

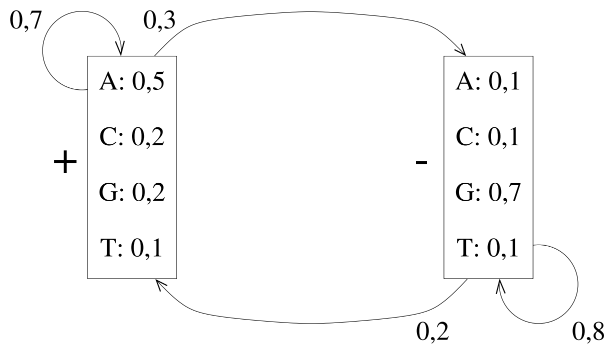


FIG. 1.2 – Chaîne de MARKOV à états cachés. Les deux états cachés (+ et -) sont caractérisés par des probabilités d'émission différentes. Par exemple, dans l'état +, la probabilité de la lettre A est $0,5$, et celle de la lettre C est $0,2$. Entre deux positions successives, la probabilité de passer de l'état + à l'état - est $0,3$, et celle de passer de l'état - à l'état + est $0,2$.

nous plaçons ici dans cette situation, comme dans l'exemple de la figure 1.2. Pour chaque état k, on note e_k la fonction de probabilité – dite probabilité d'émission – de cet état sur les lettres de la séquence : pour chaque lettre b, $e_k(b) = P(\omega_i = b|c_i = k)$.

Pour évaluer si une chaîne de MARKOV à états cachés est bien adaptée à une séquence, on peut rechercher le chemin c engendré par cette chaîne de MARKOV dont la probabilité de générer la séquence de lettres Ω est la plus forte, c'est-à-dire qui maximise (avec $c_{n+1} = 0$)

$$P(\Omega, c) = a_{0,c_1}.\prod_{i=1}^{n} e_{c_i}(\omega_i).a_{c_i,c_{i+1}}$$

Ce maximum est un indicateur de la pertinence de la chaîne de MARKOV vis-à-vis de la séquence.

Le chemin optimal peut facilement être calculé grâce à l'algorithme de VITERBI, qui utilise des procédés de programmation dynamique. De façon récursive, si on suppose connue la probabilité $v_k(i)$ du chemin qui se termine avec l'état k en position i le plus probable, on a :

$$\forall l, v_l(i+1) = e_l(\omega_{i+1}).\max_k(v_k(i).a_{kl})$$

On suppose que les séquences commencent avec l'état 0 :

$$v_0(0) = 1 \text{ et } \forall k > 0, v_k(0) = 0$$

En fin de compte, en considérant le maximum $\max_k v_k(n)$, et en parcourant en sens inverse les arguments des maximaux qui ont permis d'atteindre cette valeur, on construit le chemin le plus probable.

Une autre évaluation de l'adaptation d'une chaîne de MARKOV à états cachés sur une séquence consiste à calculer la probabilité que la séquence soit générée par cette chaîne. Cela consiste à calculer

$$\sum_c P(\Omega, c)$$

Ceci peut aussi être effectué de façon efficace, grâce à deux algorithmes issus de la programmation dynamique. Le premier – dit algorithme direct – consiste à calculer cette somme successivement sur les segments des i premières positions de la séquence, pour i allant de 1 à n. L'algorithme est similaire à celui de VITERBI en remplaçant l'opérateur de maximum par l'opérateur de somme. Le deuxième algorithme – à rebours – utilise le même principe, à la différence qu'il considère successivement selon i les segments des i dernières positions de la séquence.

En pratique, ces probabilités peuvent être très faibles, si bien qu'il est plus fiable de considérer les logarithmes de ces valeurs. Pour calculer le chemin le plus probable, on cherche

$$\max_c \big(\ln(a_{0,c_1}) + \sum_{i=1}^{n} (\ln(e_{c_i}(\omega_i)) + \ln(a_{c_i,c_{i+1}})) \big)$$

Nous pouvons alors noter que si on assimile le logarithme $\ln(e_{c_i})$ de la probabilité d'émission de l'état c_i à une fonction de prédiction des lettres de la séquence par cet état, on aboutit à un problème qui s'apparente à celui du partitionnement maximalement prédictif, à savoir ajuster au mieux une suite de fonctions de prédictions à une séquence sous une certaine contrainte. La différence essentielle est que dans le cas du partitionnement la contrainte porte sur le **nombre** de changements entre états successifs, alors que dans le cas présent du chemin le plus probable elle porte sur un **coût** de transition entre états. Ces deux approches sont voisines car leurs modèles permettent de décrire la séquence par une suite d'états de manière à optimiser un critère qui porte sur l'ensemble de la séquence. Ainsi, chercher une k-partition maximalement prédictive sur la donnée de prédicteurs qui correspondent à des états de chaînes de MARKOV, et dont les fonctions de prédiction sont les logarithmes des probabilités de ces chaînes, revient à chercher à partager au mieux une séquence en k-segments décrits chacun par une des chaînes de MARKOV disponibles.

L'approche qui consiste à calculer la probabilité de la séquence selon le modèle considéré peut aussi être adaptée à la description de celle-ci par une suite d'états. Cela consiste à ne garder, en chaque position, que l'état le plus probable.

Ceci peut être calculé simplement grâce aux algorithmes direct et à rebours. Si la séquence est ainsi décomposée en une suite d'états, cette suite ne correspond néanmoins pas à un chemin issu de la chaîne de MARKOV. Il est même possible que ce chemin soit irréalisable par cette chaîne. En revanche, en chaque position, l'état de ce chemin est le plus probable, et peut alors être considéré comme plus fiable. Le rapprochement signalé ci-dessus est apparemment perdu, puisqu'il s'agit d'un critère d'optimisation local, et non global.

Il existe différentes heuristiques pour calculer les paramètres d'un chaîne de MARKOV à états cachés. Certaines sont présentées dans [DEKM98]. Ce livre expose aussi des méthodes d'alignement de séquences au moyen de chaînes de MARKOV à états cachés. Pour notre part, nous reviendrons sur quelques aspects biologiques de ces modèles dans la section 5.2.3, page 111.

1.3.2.4 Alignement sur une expression régulière

Dans [MM89], MYERS et MILLER généralisent le problème d'aligner globalement une séquence avec une autre en cherchant à aligner au mieux une expression régulière sur une séquence donnée. Par exemple, dans le cas où la similarité d'une lettre sur elle-même rapporte 1, la similarité entre deux lettres différentes rapporte -10, et une insertion coûte -1, un alignement global optimal entre la séquence `AGCAGACGGTA` et l'expression régulière `(AG|C)*GCTA` est

```
AGCCAGACGG-TA
AGCCAG-C-GCTA
```

Leur méthode consiste à utiliser l'automate qui reconnaît l'expression rationnelle concernée. Le principe est le même que celui de l'algorithme de programmation dynamique présenté en 1.3.1.1. Le parcours du graphe et de la séquence sont faits en même temps, mais la lecture sur l'un peut être mise en attente pendant que l'autre continue, ce qui a pour conséquence d'insérer des trous dans le parcours du graphe et dans celui de la séquence. MYERS et MILLER proposent plusieurs algorithmes dynamiques pour faire cela, selon différentes gestions des insertions. Ces algorithmes construisent un alignement optimal avec une complexité en temps linéaire avec la taille de la séquence et la taille de l'expression régulière.

Nous avons remarqué que dans le cas particulier où l'alignement entre une séquence et une expression régulière ne tolère aucune insertion (ou alors, chaque insertion est tellement pénalisante qu'aucune ne va se produire dans l'alignement optimal), on construit un lissage de la séquence en segments qui correspondent aux classes d'une partition prédictive optimale.

Par exemple, aligner au mieux l'expression régulière $\texttt{(A*|C*|G*|T*)}^{\texttt{k}}$ avec une séquence construit un lissage de la séquence en k segments construits par la répétition d'une même lettre. Ceci correspond au lissage que produit une k-partition maximalement prédictive construite avec l'ensemble de prédicteurs `{A, C, G, T}`. Ainsi, sur la séquence

```
AATAACCGCCCTTCTT
```

avec l'expression régulière $\texttt{(A*|C*|G*|T*)}^{3}$, le lissage produit est

```
AAAAACCCCCCTTTTT
```

De même, le lissage selon l'expression régulière

```
(A*|(AC)*|(AG)*|(AT)*|(CA)*|C*|(CG)*|(CT)*|
                 (GA)*|(GC)*|G*|(GT)*|(TA)*|(TC)*|(TG)*|T*)
```

fournit le lissage

CACACACACACACACA

qui correspond à la description optimale sur une classe par un prédicteur parmi les prédicteurs structurés de période 2.

Nous voyons ainsi qu'en prenant tous les prédicteurs admissibles avec toutes leurs phases, et qu'en alignant la séquence sur l'expression régulière de même type que celles données en exemple, on construit un lissage en k segments, pour un certain k, et que ce lissage est celui produit par la description d'une k-partition prédictive optimale. Le fait de considérer des motifs comme prédicteurs structurés permet d'opérer un lissage qui s'adapte au mieux aux différentes fluctuations périodiques au sein de la séquence.

1.3.2.5 Partitionnement maximalement prédictif

La classification construite par partitionnement maximalement prédictif est une méthode de description de séquence qui cherche à approcher au mieux une séquence par une partition en classes prédictives. La donnée d'une suite de classes prédictives permet de décrire la séquence et permet, avec une certaine fiabilité, de prédire les objets de cette séquence. Lorsqu'il n'y a pas de prédicteurs contextuels, une k-partition maximalement prédictive produit une description similaire à un lissage sans insertion par une expression régulière bien précise, comme nous l'avons vu au paragraphe précédent. En revanche, la considération de prédicteurs contextuels ne fournit pas vraiment un alignement avec la séquence puisqu'une position de la séquence est alors prédite par un mot, et non par une simple lettre.

À la différence de l'approche de Myers et Miller, le partitionnement maximalement prédictif considère plusieurs nombres de segments pour lisser la séquence. En effet, le partitionnement est tout d'abord un processus de classification, en ce sens qu'il cherche à résumer au mieux la structure d'un ensemble d'objets. En général, tout le problème est de trouver un compromis pour le nombre de classes entre le fait de pratiquer un réel résumé de la structure de la séquence, ce qui suppose un petit nombre de classes par rapport à la taille de la séquence, et le fait de bien décrire la séquence, ce qui suppose au contraire de petites classes, c'est-à-dire un grand nombre de classes. C'est ce compromis qui guide notre recherche, en étudiant quel est le comportement successif en fonction de k des k-partitions qui constituent ce partitionnement, pour trouver un « bon » nombre de classes.

Ce dernier point peut s'avérer problématique, car difficile à connaître. En revanche, il compense l'introduction d'autres facteurs. Par exemple, si une fenêtre de lissage est utilisée, il est généralement difficile de déterminer la taille de cette fenêtre de façon non arbitraire et, souvent, prendre une même taille de fenêtre pour lisser toute une séquence qui présente de grandes différences de fréquences d'une zone à une autre gêne l'obtention d'un « bon » lissage sur l'ensemble du signal.

La recherche du nombre de classes est un des problèmes inhérents à la classification, très lié à celui de savoir ce qu'est un « bon » type de classification pour un ensemble d'objets.

1.4 Du type de classification

Dans les sections précédentes, le type d'une classification (k-partition, hiérarchie, pyramide, etc) a été présenté comme un choix, ou une connaissance *a priori* de l'expérimentateur. Souvent, il n'y a pas à faire ce choix car l'organisation des données est déterminée par un contexte extérieur. Par exemple, la phylogénie est naturellement liée à la construction de hiérarchies, et la taxonomie cherche plus volontiers à construire des partitions [Gil40]. Mais il n'est pas toujours évident de savoir quel type de classification est le mieux adapté à un problème et à un ensemble d'objets donné, ni même si cet ensemble admet d'être soumis à une quelconque classification.

1.4.1 De la validation d'une classification

Lorsqu'un choix est fait, l'adaptation d'une classification à une structure de l'ensemble étudié peut être discutable. De la même sorte, on peut être amené à se demander si le critère d'évaluation et la méthode – lorsqu'elle ne fournit qu'un résultat approché – choisis ont permis de construire une classification « valable ». En outre, comme une méthode de classification construit généralement une classification même si l'ensemble ne s'y adapte pas, il faut se demander en quelle mesure cette classification obtenue est naturelle, ou au moins quelles sont les classes construites qui sont naturelles. Enfin, lorsque plusieurs classifications sont trouvées qui ne peuvent pas être départagées par l'évaluation qui a servi à les construire, on peut aussi vouloir choisir laquelle correspond le mieux à l'ensemble.

On peut lire dans [Gor96a] une anthologie de ces différents problèmes de validation et des approches pour les résoudre.

D'un point de vue statistique, valider une classification revient à faire des tests pour confirmer ou contredire l'hypothèse de structure que tente de faire apparaître cette classification. Plusieurs hypothèses sont étudiées dans [Boc85]. En particulier, on appelle hypothèse nulle H_0 l'hypothèse selon laquelle un ensemble n'a pas de structure interne adaptée à un type de classification donné. La méthode de Monte Carlo vise à déterminer, sous une hypothèse H_1, la distribution d'une variable liée au type de classification étudié. Pour une classification construite sur un ensemble donné, cette variable prend une valeur donnée. Si cette valeur dépasse un critère $c(\alpha)$ déterminé grâce à la distribution obtenue, on peut infirmer, pour cet ensemble et cette classification, l'hypothèse H_1 avec une probabilité d'erreur inférieure à α. Par exemple, dans [BD82], sur des graphes aléatoires, les lois de probabilité de différents critères sont calculées, pour permettre d'estimer si une classe particulière formée au sein d'une partition d'un graphe se situe hors de l'hypothèse nulle, et ainsi est intéressante.

On peut aussi estimer que la validité d'une classification est directement liée à sa stabilité, ou à la stabilité de ses classes, lorsqu'elle est soumise à certaines modifications. Une approche empirique du test de Monte Carlo de l'hypothèse nulle H_0 consiste à mélanger aléatoirement une population d'ensembles d'objets à partir de l'ensemble initial, de manière à y « détruire » une éventuelle structure qui s'adapterait au type de classification recherché. Puis on compare les évaluations des classifications construites par la même méthode sur ces ensembles mélangés à celle de l'ensemble initial [GF85]. *A priori*, si un type de classification est particulièrement bien adapté à l'ensemble étudié, il sera moins efficace lorsque cet ensemble aura été perturbé, et le critère de rejet de l'hypothèse H_0 déterminé ainsi est justifié. L'écart entre ces évaluations peut même permettre de mesurer la pertinence de la classification étudiée. En suivant une idée analogue, Gabriel et Sokal [GS69] élaborent des partitions à l'aide de tests d'homogénéité selon le principe suivant : si une classe est homogène alors tout sous-ensemble de cette classe est homogène, et une classe sera déclarée telle si le calcul d'un certain critère est en deça d'une valeur seuil déterminée de façon statistique.

Dans [BM98], l'approche est différente, puisqu'il considère qu'une classe est d'autant mieux construite qu'elle est plus stable face à la perturbation de l'ensemble des données, c'est-à-dire que le processus classificatoire devrait reconstruire cette classe lorsque l'ensemble des objets a été un peu modifié (par exemple, en déplaçant, en enlevant, ou en ajoutant un objet). L'évaluation de cette stabilité se fait à partir de critères de compacité et d'isolement de cette classe. De même, dans [GKL77], la stabilité d'une classification est évaluée par sa « résistance » à des variations dans les données.

1.4.2 Du nombre de classes d'une partition dans un contexte général

Un cas particulier du problème de validation d'une classification est, lorsqu'il s'agit d'opérer une partition, de choisir en combien de classes il est « bon » de faire cette partition. Ce problème est très comparable à celui de trouver un « bon » k pour extraire une k-partition d'un partitionnement, et il se rapproche assez de celui de choisir des classes au sein d'une hiérarchie.

En ce qui concerne une hiérarchie, si elle est vue comme une voie qui relie les individus isolés à la population dans sa globalité [LW66], une fois une fonction d'évaluation choisie, la hiérarchie optimale s'affirme comme allant d'elle même. Mais lorsqu'il s'agit d'extraire une partition d'une hiérarchie, le problème se pose de savoir laquelle choisir parmi la multiplicité de partitions qu'offre la hiérarchie (figure 1.3). En outre, il n'y a pas de garantie que la k-partition extraite est optimale parmi l'ensemble des k-partitions.

Souvent, les classes qui vont constituer la partition sont sélectionnées grâce au gain d'évaluation qu'apporte leur division. Plus ce gain est faible pour une classe, plus elle a des chances d'être sélectionnée comme classe de la partition. Il faut que cette sélection prenne en compte le fait que les classes de la partition doivent être toutes deux à deux disjointes. À partir d'observations empiriques, Witkin [Wit83] décide de considérer la stabilité des classes au sein d'une hiérarhie. Lance et Williams [LW65] recherchent une partition en évaluant l'information gagnée par chaque scission de cette hiérarchie, et la variation de cette information permet de déterminer l'arrêt du processus. De même, dans [HS83], la variation de l'information taxonomique (proche de la prédiction de Gower [Gow74]) permet de choisir l'arrêt d'un processus agglomératif ou divisionnaire de hiérarchie.

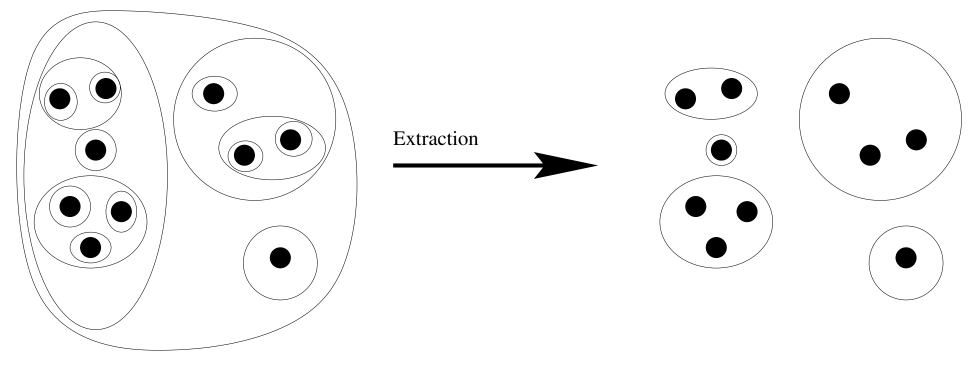


FIG. 1.3 – Extraction d'une k-partition à partir d'une hiérarchie. Le choix est fait parmi un grand nombre de possibilités.

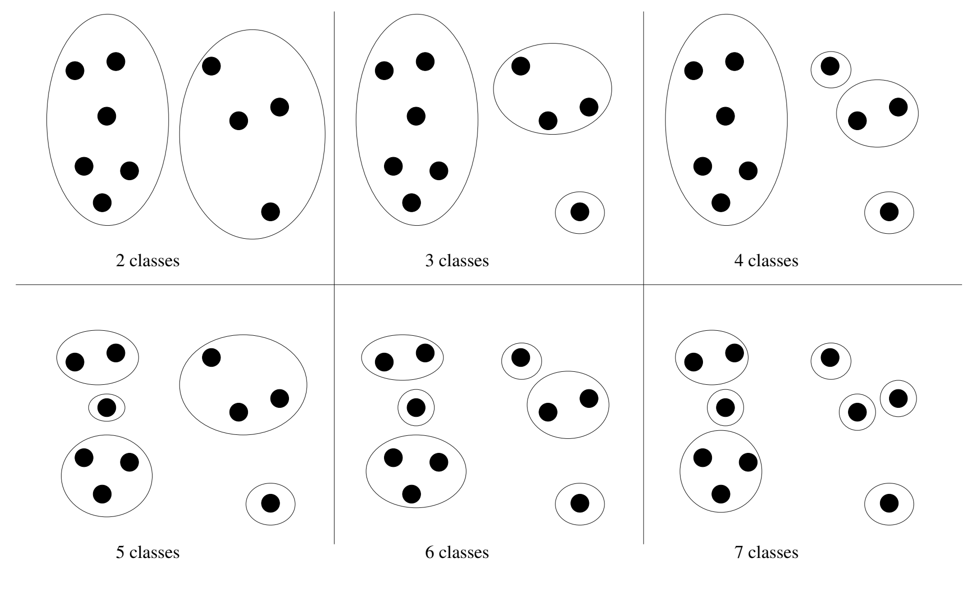


FIG. 1.4 – Extraction d'une k-partition à partir d'un partitionnement optimal. Elle est faite sur un choix restreint de possibilités. En outre, toute k-partition extraite de ce partitionnement est optimale dans l'ensemble des k-partitions.

Construire un partitionnement permet de rechercher le nombre de classes nécessaire à bien partager un ensemble, lorsqu'il n'est pas possible de connaître ce nombre au départ. Le partitionnement est soit mené jusqu'à un partage total et une étude rétrospective permet de déterminer quelle(s) partition(s) est(sont) à garder, soit jusqu'à la satisfaction d'un critère. La différence entre l'extraction d'une partition à partir d'une hiérarchie et l'extraction d'une partition à partir d'un partitionnement réside dans la différence du critère d'optimalité de ces types de classification. Extraire une partition d'un partitionnement signifie en prendre une déjà présente dans celui-ci, puisque ceci garantit son optimalité parmi toutes celles qui partagent son nombre de classes, à la différence d'une hiérarchie (figure 1.4). Ceci ne revient pas seulement à rechercher un niveau ad hoc – le nombre de classes – puisqu'à un même niveau peuvent être présentes plusieurs partitions optimales, entre lesquelles on voudra éventuellement aussi choisir.

Il est possible que l'évaluation sur laquelle s'est basée la classification suffise à déterminer précisément le nombre de classes. C'est le cas lorsque la suite des optimaux de cette fonction n'est pas monotone avec le nombre de classes. Les deux approches essentielles consistent à tenir compte, dans l'évaluation, d'une part de la pertinence des classes, et d'autre part ou bien du coût qu'occasionne la description de ces classes (point de vue du résumé de l'information) [WB68, FVN71], ou bien de l'isolement de ces classes entre elles. Ruspini [Rus69] combine ces deux critères.

Cependant, une partition avec beaucoup de classes a ses classes en moyenne plus petites qu'une partition plus grossière, si bien que ces dites classes ont en moyenne plus de chances d'être homogènes, et la partition d'autant meilleure. Par exemple pour la somme des variances des classes [Fis58, ECS65] ou bien la prédiction [Gow74], la classification globalement optimale est celle où l'ensemble est partagé en classes uniformes, ce qui présente peu d'intérêt. La variation de l'évaluation peut alors être utilisée pour déterminer l'arrêt du partitionnement. On s'arrête lorsqu'apparaît un point d'inflexion très net dans la courbe du critère de classification. Ce point d'inflexion est appelé **coude**. Une absence de coude signifierait qu'il n'y a pas de moment déterminant pour la classification, et qu'il n'y a donc pas de structure apte à être partagée. Reste à déterminer ce qu'est une absence de coude.

Un seuil qui déterminera l'arrêt du processus peut être introduit [LDL85], mais il est parfois difficile à justifier. De plus, utiliser un même seuil pour l'ensemble d'une classification suppose que ce seuil se justifie uniformément sur l'ensemble étudié, ce qui n'est pas sûr, puisque la classification peut *a priori* avoir des comportements très variables d'un groupe d'objets à un autre.

On peut aussi, en plus de l'évaluation, introduire un autre critère. Celui-ci peut prendre en compte l'information contenue dans la classification, ou bien l'isolement des classes. Dans [CH74], Caliński et Harabasz comparent la dispersion des objets entre les classes et la dispersion de ces objets à l'intérieur des classes. De nombreux critères de décision ont été proposés. Dans [MC85], trente d'entre eux sont comparés, selon leur aptitude à trouver le nombre correct de classes de quelques quatre cents classifications (variables selon l'ensemble d'objets et la méthode de classification) et le critère de Caliński et Harabasz a eu les meilleurs résultats. Dans cette étude, les classifications ont été faites sur des ensembles ayant un nombre précis de classes, mais ceci représente un cas particulier. En effet, rien ne dit qu'un *unique* nombre de classes correspond à un ensemble de données. Gordon [Gor96b] a éprouvé les cinq meilleurs critères du classement de [MC85] sur des ensembles de données constitués de quelques groupes, chacun d'eux comprenant plusieurs groupes plus restreints. La détermination des deux nombres de classes (les gros groupes, et les petits), sur des classifications opérées par plusieurs méthodes sur plusieurs ensembles plaide aussi en faveur du critère de Caliński et Harabasz. Cependant, ce résultat doit être tempéré par le fait que dans ces deux études les groupes sont nettement séparés les uns des autres, hypersphériques et que la répartition des objets au sein de chaque groupe a été faite selon une loi normale multivariée, ce qui peut particulièrement favoriser ce critère. Dans [HD99], ces critères, ainsi que d'autres méthodes, ont été testés sur des données moins « favorables » et se sont avérés bien moins efficaces.

Dans le cas particulier de la prédiction [GOW74], l'isolement d'une classe et de son prédicteur optimal est défini par la prédiction de ce prédicteur sur les autres classes de la partition : plus est mauvaise cette prédiction, plus isolée est la classe. Ainsi, GOWER définit sa variable « between », que nous adaptons à une k-partition prédictive $\mathcal{C} = \{(C, d_C)\}$:

$$B_k^D(\mathcal{C}) = \frac{1}{k-1} \sum_{C \in \mathcal{C}} \sum_{C' \in \mathcal{C}: C' \neq C} \pi_{d_C}(C')$$

La justification qu'il apporte au fait de choisir pour nombre d'arrêt k celui qui maximise $M_k^D - B_k^D$ est que rajouter une variable de description qui est uniforme sur l'ensemble des objets ne doit provoquer aucune modification. Il est possible que M_k^D soit atteinte par plusieurs k-partitions prédictives ; dans ce cas, il faut considérer *parmi celles-ci* celle qui minimise B_k^D.

1.4.3 Du nombre de classes dans le partitionnement maximalement prédictif sous contrainte d'ordre total

Dans le contexte particulier du partitionnement prédictif sous contrainte d'ordre total, l'évaluation est une fonction croissante du nombre de classes. Il faut alors chercher un moyen qui permette d'estimer si, étant donnés une séquence et un ensemble de prédicteurs, il est possible de construire une « bonne » k-partition prédictive sur cette séquence, et dans ce cas pour quel(s) k. En parallèle, il faut à l'occasion pouvoir choisir entre plusieurs k-partitions prédictives optimales.

Nous adaptons dans cette partie différentes techniques parmi celles qui sont présentées ci-dessus. Ces techniques ne doivent pas faire intervenir de notion de dissimilarité entre classes, puisque l'évaluation ne se fait pas en séparant des classes les unes des autres, mais en adaptant au mieux une classe à un prédicteur. Nous proposons aussi de nouveaux critères d'évaluation du nombre de classes.

1.4.3.1 Monte Carlo

La démarche est la même que pour le cas général. Sur une population générée aléatoirement à partir de la séquence initiale Ω, on extrait une distribution de l'évaluation. Ainsi, si la population est assez nombreuse et générée de façon uniforme, on peut tester avec fiabilité l'appartenance de la séquence aux α-percentiles, c'est-à-dire qu'elle appartient aux α % supérieurs de cette distribution. Pour garantir une suffisamment bonne fiabilité en cette appartenance, il est bon de créer une population d'au moins n individus, avec $n >> 100/\alpha$, c'est-à-dire $n \geqslant 1000/\alpha$. Si $\alpha = 5$, cela fait $n \geqslant 200$.

Bien sûr, il importe de générer la population aléatoire de façon convenable pour que la distribution empirique soit satisfaisante. Ainsi, si on veut construire une population qui permette de mettre en relief une éventuelle structure de l'ensemble d'objets étudié vis-à-vis d'un ensemble de prédicteurs, il faut générer une population dont les caractéristiques générales – indépendamment des caractéristiques structurales – soient similaires à celles de l'ensemble initial par rapport aux prédicteurs. Ainsi, il paraît raisonnable de construire chaque individu de la population aléatoirement de façon uniforme parmi les séquences dont la prédiction par n'importe quel prédicteur δ de D est assez proche de $\pi_\delta(\Omega)$. Dans le cas classique où les prédicteurs admissibles ne sont ni contextuels ni structurés, il suffit pour faire cela de mélanger de façon uniforme les éléments de la séquence. Dans le cas contraire, on pourra chercher à générer de façon uniforme des séquences qui s'approchent de façon assez satisfaisante de la séquence initiale ; la valeur de satisfaction peut être un test de χ^2 sur les prédictions pour l'ensemble des prédicteurs.

Si effectuer un partitionnement prend beaucoup de temps, devoir travailler une telle population peut être pénible, et il est clair qu'une méthode qui n'ait pas à faire cela serait préférable, soit par le calcul de la distribution, soit par un critère non statistique.

1.4.3.2 Nombre d'ex æquo

Lorsque la méthode de classification fournit toutes les k-partitions optimales, il n'est pas forcément trivial d'en choisir une. A fortiori, la présence de beaucoup d'ex æquo pour un k donné est plutôt signe de l'indécision de la classification à partager l'ensemble en k classes, et du fait qu'aucun de ces partages n'est décisif de façon satisfaisante. L'ensemble se partage mal en k classes. Au contraire, s'il y a très peu de partitions, surtout si elles sont proches les unes de autres (voir paragraphe suivant), la structure de l'ensemble est plus en accord avec un partage en k classes.

La figure 1.5, que l'on appellera **graphe des k-partitions**, décrit l'ensemble des 13-partitions optimales d'une séquence quelconque de 300 lettres pour des prédicteurs `A`, `C`, `G`, `T`.

Cette figure illustre le problème qu'il y aurait à compter les k-partitions pour s'en servir comme critère d'ambiguïté. En effet, ce nombre est le *produit* du nombre de chemins du graphe. Si par exemple, au centre, un chemin annexe au chemin des deux segments marqués `T` et `A` reliait les deux nœuds extrêmes de ce chemin, soit la création de deux classes, le nombre total de partitions optimales serait encore doublé. Au contraire, s'il existait une 13-partition de toute la séquence totalement étrangère à ce graphe, le nombre d'ex æquo serait juste incrémenté, alors qu'en fait k nouvelles classes seraient créées. Ce phénomène est disproportionné en comparaison de l'ambiguïté apportée. Ainsi, il semble plus raisonnable de compter le nombre d'arêtes de ce graphe.

Pour homogénéiser cette fonction en fonction du nombre k de classes, on considère alors la fonction du nombre d'arêtes du graphe représentant $\mathbb{P}_k$ divisé par k. Pour les partitions non ambiguës (comme $\mathbb{P}_1$ et $\mathbb{P}_{|\Omega|}$), cette fonction vaut 1, et plus elle est grande plus il y a d'ambiguïtés du choix des classes.

Ce critère n'est réellement intéressant que lorsqu'il peut souvent y avoir des ex æquo. Ceci signifie qu'il est facile à des sommes de prédictions de classes d'être

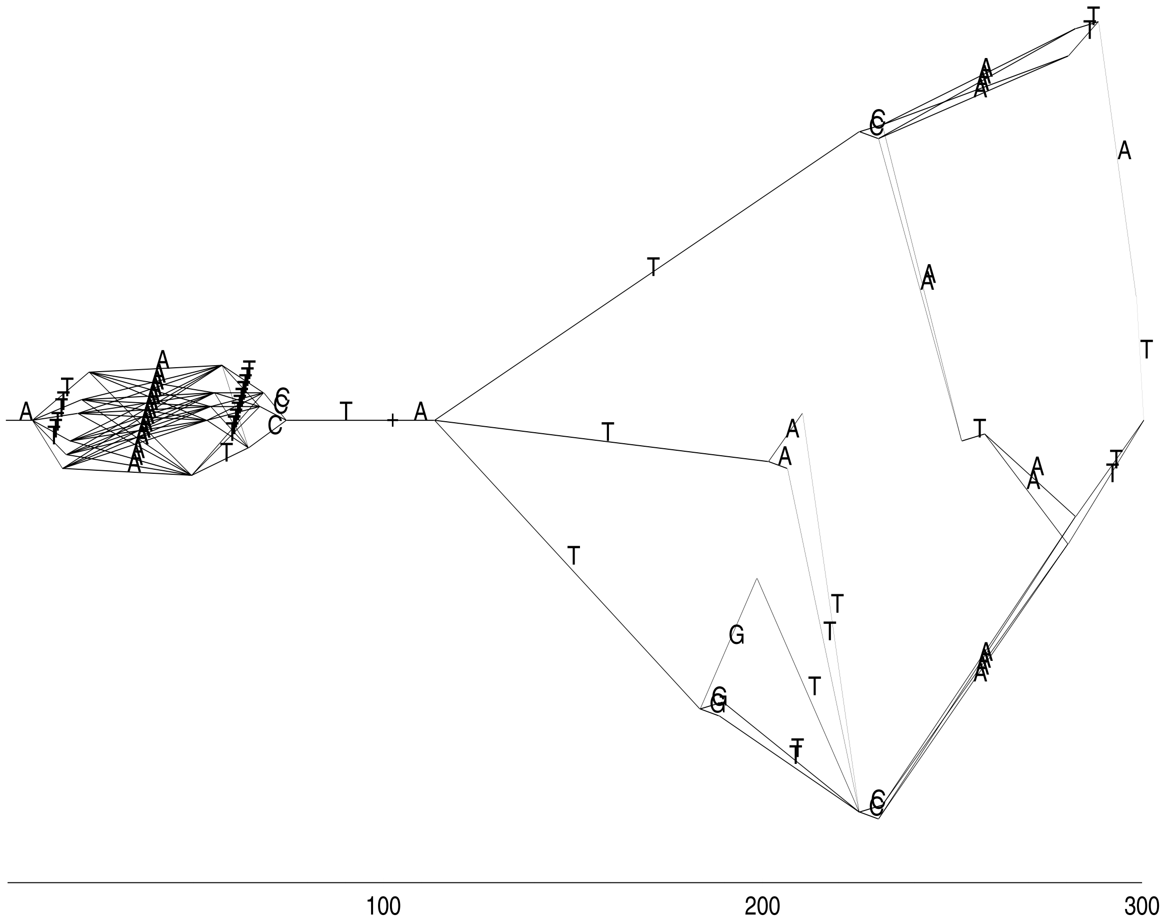


FIG. 1.5 – Graphe des 13-partitions. La séquence est représentée par le trait gradué horizontal. Une classe prédictive est représentée par un segment du graphe, la lettre située au-dessus d'elle est son prédicteur optimal. La jonction entre deux classes se situe à la verticale de sa position dans la séquence. Une partition s'obtient en parcourant le graphe dans un même sens, en choisissant aux nœuds du graphe de quelles classes prédictives elle se compose.

égales. Pour cela, comme la prédiction d'une classe est la somme de prédictions d'objets, on peut s'attendre à ce que le nombre maximum d'ex æquo soit lié au nombre de valeurs de prédictions possibles d'objets. Ce nombre doit aussi dépendre du fait que ces valeurs sont des multiples entières les unes des autres (par exemple, prédire deux fois `A` vaut autant que prédicre trois fois `C`). Ainsi, si les prédictions de `A`, `C`, `G`, `T` sont respectivement par exemple $1,123$, $1,234$, $1,345$ et $1,456$, il y a potentiellement peu d'ex æquo parmi les k-partitions optimales.

1.4.3.3 Incertitude de prédiction

Une autre vision est liée à l'objectif de prédire les objets de la séquence. Comme on veut apporter une description de chaque objet, on peut se déclarer insatisfait si cet objet n'est pas prédit précisément par la classification. Ainsi, plus est grande l'incertitude de prédiction de cet objet par l'ensemble des k-partitions prédictives optimales, plus est mauvais le niveau k pour la prédiction de cet objet. Sur chaque objet, cette incertitude peut être le nombre de prédictions différentes parmi l'ensemble des k-partitions prédictives optimales. Sur la séquence, l'incertitude de prédiction est la moyenne du nombre de prédictions sur l'ensemble des objets. S'il n'y a pas d'ambiguïté, cette moyenne vaut 1, et plus elle est grande, plus incertaines sont les prédictions sur les objets.

1.4.3.4 Densité de prédiction

Par le fait que les classes doivent être contiguës, des objets se trouvent à l'intérieur d'une classe et sont pourtant mal prédits par le prédicteur optimal de celle-ci. Globalement, sur n'importe quelle partition prédictive, la somme des mauvaises prédictions sur l'ensemble des objets de la séquence et de la prédiction de cette partition est constante, quelle que soit cette partition.

Lorsqu'on s'intéresse à une classe prédictive, en plus de sa prédiction, qui est le critère d'évaluation que nous avons choisi, on peut s'intéresser à la densité[5] de prédiction pour évaluer cette classe, c'est-à-dire sa prédiction divisée par son cardinal. Plus la proportion de bonnes prédictions sur cette classe est grande, meilleure est la classe. Notons $\tau_C = \frac{\pi_C}{|C|}$ la densité de prédiction d'une classe. Pour une partition, un critère d'évaluation consiste à faire la moyenne des densités de prédiction de ses classes. Soit $\mathcal{C} = \{(C_i, d_i)\}_{1 \leqslant i \leqslant k}$ une k-partition prédictive,

$$\tau_k^D(\mathcal{C}) = \frac{1}{k} \sum_{1 \leqslant i \leqslant k} \tau_{C_i}$$

Parmi les différentes k-partitions prédictives optimales, la valeur de τ_k^D risque d'être différente. Ainsi, la densité peut être un moyen de sélection parmi les ex æquo. En considérant le maximum de la fonction τ_k^D sur l'ensemble des k-partitions prédictives optimales, comme la somme globale des prédictions est constante sur cet ensemble, on pénalise en fait les k-partitions dont les petites classes ont de mauvaises prédictions. En effet, chaque mauvaise prédiction d'un objet a, dans la densité, une importance inversement proportionnelle à la taille

[5] ou compacité, ou à l'inverse la dispersion

de la classe à laquelle il appartient. Ainsi, pour maximiser la densité sur les partitions, mieux vaut que les mauvaises prédictions soient dans de grandes classes que dans de petites. Comme il y a autant de mauvaises prédictions sur l'ensemble de la séquence pour toutes les partitions optimales, les grandes classes sont alors en moyenne moins bien prédites. Au contraire, vouloir minimiser τ_k^D sur l'ensemble des k-partitions prédictives optimales revient à accorder moins d'importance à la prédiction des petites classes.

Le choix entre minimisation et maximisation dépend alors du point de vue. Si on considère que toutes les classes de la partition, quelles que soient leurs tailles, ont la même importance, on cherchera à maximiser la densité de prédiction. Inversement, si on considère que seules les plus grandes classes sont importantes, et que les plus petites classes ne présentent pas d'intérêt, mieux vaut minimiser la fonction τ_k^D. Si on considère que l'importance de la prédiction sur une classe est proportionnelle à la taille de cette classe, on retrouve le critère de prédiction.

En ce qui concerne le choix d'un nombre de classes par le critère de densité de prédiction, le choix parmi les k-partitions optimales peut être fait par minimisation ou par maximisation de la densité de prédiction. Comme les classes formées sont en moyenne de plus en plus petites au cours du partitionnement, la densité a tendance à augmenter avec le nombre de classes. Elle prend sa valeur maximale lorsque le partitionnement est achevé, et toutes les classes parfaitement prédites.

1.4.3.5 Isolement

Il s'agit d'adapter à ce contexte le critère d'isolement de GOWER [GOW74], qui préfère qu'une classe prédictive soit d'autant plus pénalisée qu'elle prédit bien les objets des classes voisines. Dans le contexte non contraint où se place GOWER, ceci signifie toutes les autres classes que la classe concernée. En revanche, sous contrainte de contiguïté, il n'est pas naturel de pénaliser une classe si son prédicteur prédit bien une classe géométriquement séparée de celle-ci. On peut définir la fonction d'**interaction** pour une k-partition prédictive $\mathcal{C} = \{(C, d_C)\}$, si $V(C)$ est l'ensemble des classes contiguës à C :

$$I_k^D(\mathcal{C}) = \frac{k}{\sum_{C\in\mathcal{C}}|V(C)|}\sum_{C\in\mathcal{C}}\sum_{C'\in V(C)}\pi_{d_C}(C')$$

Le facteur $\frac{k}{\sum_{C\in\mathcal{C}}|V(C)|}$ permet, de même que pour le critère de GOWER, de permettre à $M_k^D - I_k^D$ d'être indépendant d'une prédiction qui serait uniforme sur l'ensemble des objets. Si pour chaque k on considère la k-partition prédictive optimale C_k qui minimise I_k^D, le nombre de classes k est celui qui maximise $M_k^D(C_k) - I_k^D(C_k)$.

Là aussi, pour les différents éléments de $\mathbb{M}_k^D$, I_k^D pourra être différent, et il est plus judicieux de garder les partitions qui minimisent I_k^D. Dans le cas particulier d'une séquence, si $\mathcal{C} = \{(C_i, d_i)\}_{1\leqslant i\leqslant k}$, C_1 et C_k étant les classes extrêmes, C_1 n'a pour classe voisine que C_2 et C_k n'a pour classe voisine que C_{k-1}, alors que les autres classes C_i ont pour voisines C_{i+1} et C_{i-1}. Par conséquent,

$$I_k^D(\mathcal{C}) = \frac{k}{2k-2}\Big(\pi_{d_1}(C_2) + \sum_{i=2}^{k-1}(\pi_{d_i}(C_{i-1}) + \pi_{d_i}(C_{i+1})) + \pi_{d_k}(C_{k-1})\Big)$$

Dans le cas où les prédicteurs admissibles sont structurés, considérer un tel critère pose le problème de phase des prédicteurs. Par exemple, une partition optimale en deux classes de la séquence `ATATCTAGCACTCGCG` avec des prédicteurs structurés bilitères sur les lettres donne le lissage `ATATATAGCGCGCGCG`. Il est sensé que l'isolement entre ces deux classes prédictives soit mesuré à partir de chaque classe par l'itération de son prédicteur sur l'autre classe, dans la continuité de l'itération de celui-ci sur elle-même. Dans le pire des cas, deux classes prédictives dont les prédicteurs optimaux sont identiques et en phase ne seront pas du tout isolées l'une de l'autre. Cependant, il suffit qu'il y ait un objet de plus dans la séquence, à la jonction des deux classes, pour qu'il se produise un déphasage dans l'itération des prédicteurs à l'extérieur de leur classe propre, et la valeur de la fonction d'isolement peut alors être complètement modifiée. Cette instabilité peut sembler gênante, car un seul objet en plus ou en moins ne traduit pas forcément un grand changement dans la structure de la séquence. Mais il faut noter qu'un objet en plus ou en moins au début d'une classe traduit forcément un changement de phase de ses prédicteurs optimaux. Par exemple, les séquences `ATATATATATATATAT` et `ATATATATTATATAT` ne diffèrent que par une lettre de différence, et pourtant le changement de phase doit être pris en compte par le processus de classification. L'instabilité de la fonction between n'est en fait qu'un reflet de l'instabilité de la classification par des prédicteurs structurés.

Les critères présentés ci-avant obligent à construire toutes les k-partitions optimales, ce qui peut être difficile. Il serait plus économique d'avoir un critère qui se contente d'une seule partition optimale quelconque par niveau de partitionnement.

1.4.3.6 Incréments de prédiction

On considère que l'intérêt d'un partitionnement persiste tant que le gain de prédiction entre M_k^D et M_{k+1}^D est suffisamment grand, ce qui se rapproche de la notion de gain d'information taxonomique exprimée dans [HS83]. Il s'agit d'estimer ce qu'est « suffisamment grand ». Pour cela, on peut relever le maximum global du gain de prédiction. Mais si ce gain est de valeur comparable en plusieurs endroits, il n'est pas juste de ne considérer que le plus grand, et de négliger les autres. Si le partitionnement produit des gains importants et comparables jusqu'à, disons 10 classes, puis des gains négligeables à partir de 11 classes, intuitivement, il est raisonnable de retenir les 10-partitions. Ainsi, c'est quand il y a une chute du gain qu'il faut s'arrêter, ce qui revient au coude dont il a été fait mention dans la section 1.4.2, page 27. On peut voir dans la figure 1.6 deux fonctions de prédiction où un coude apparaît nettement.

Pour calculer ce coude, on peut considérer la différence entre les pentes successives de la fonction de prédiction pour les k successifs. Cependant, une

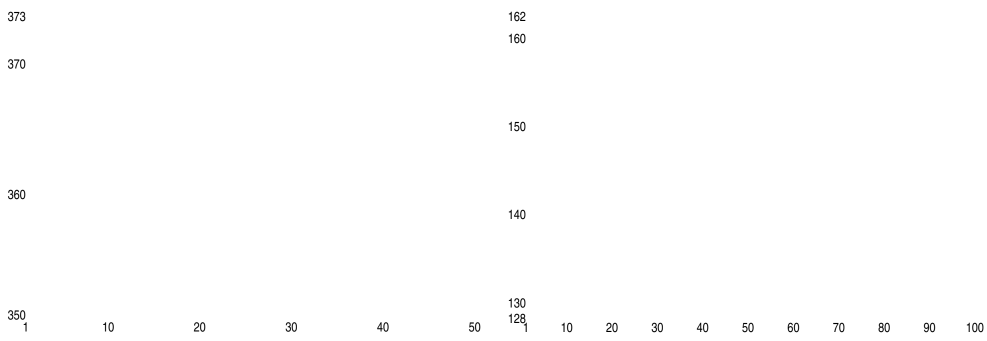


FIG. 1.6 – Deux fonctions de prédiction issues de partitionnements où le « coude » apparaît nettement. L'axe des abscisses porte le nombre de classes et celui des ordonnées porte la prédiction.

fonction qui ne prend en compte que les différences de variation entre les niveaux $k-1$, k et $k+1$ risque d'être trop soumise aux petites variations locales pour vraiment révéler un coude, comme dans le cas de la courbe de droite de la figure 1.6. Au contraire, considérer un lissage de cette courbe permettrait de sélectionner les coudes de façon sûre, mais peut-être imprécise selon le lissage pratiqué.

Dans le chapitre 4, nous continuerons la discussion au sujet du choix du nombre de classes du partitionnement. En particulier, nous mettrons en évidence des propriétés qui permettrons de restreindre la recherche du nombre de classes.

Dans ce chapitre, nous avons défini ce qu'est le partitionnement maximalement prédictif sous contrainte d'ordre total, et avons placé cette notion dans les deux contextes de la classification et de la redescription de séquences. Il faut maintenant mettre au point un algorithme efficace ainsi qu'une syntaxe de lexique pour les ensembles de prédicteurs avec lesquels on veut pratiquer le partitionnement maximalement prédictif.

Chapitre 2

Partitionnement de séquences

« Ayant donc achevé ses préparatifs, il ne voulut pas attendre davantage pour mettre à exécution son projet. »
CERVANTÈS
Don Quichotte de la Manche

2.1 Notations

Sur un ensemble $\Omega = \{\omega_1, \dots, \omega_n\}$, et un ensemble de prédicteurs D, on utilise les notations suivantes.

p	l'évaluation des classes
P	l'évaluation des classifications
$\mathbb{P}_k(\Omega)$	l'ensemble des k-partitions de Ω
$\mathbb{M}_k(\Omega)$	l'ensemble des k-partitions optimales de Ω
$\mathbb{M}(\Omega)$	l'ensemble des partitions optimales de Ω
$M_k(\Omega)$	l'évaluation des k-partitions optimales de Ω
π_δ	la fonction de prédiction de $\delta \in D$
$\mathbb{P}_k^D(\Omega)$	l'ensemble des k-partitions prédictives de Ω
$\mathbb{M}_k^D(\Omega)$	l'ensemble des k-partitions prédictives optimales de Ω
$M_k^D(\Omega)$	l'évaluation des k-partitions prédictives optimales de Ω

On note $\langle i, j\rangle = \{i, i+1, \dots, j\}$.

Désormais, Ω est muni d'une structure de séquence. On notera, pour $i \leqslant j$, $\omega_{\langle i,j\rangle} = \{\omega_i, \omega_{i+1}, \dots, \omega_j\}$.

De même, une k-partition prédictive sur une séquence de taille n est notée $((x_0, \delta_0), (x_1, \delta_1), \dots (x_{k-1}, \delta_{k-1}))$ avec $x_0 = 0$ et $x_k = n$. Les x_i sont appelés les **césures** de la partition. Autrement dit, une classe de la partition est délimitée par ses deux césures.

Pour $y > x$, on note la prédiction de δ sur $\omega_{\langle x+1,y\rangle}$, $\pi_\delta(x, y) = \pi_\delta(\omega_{\langle x+1,y\rangle})$. Par convention, on note, si $x > y$, $\pi_\delta(x, y) = -\pi_\delta(y, x)$, et $\pi_\delta(x, x) = 0$. On voit aisément que :

$$\forall x, y, z : \pi_{\underline{\delta}}(x, z) = \pi_{\underline{\delta}}(x, y) + \pi_{\underline{\delta}}(y, z)$$

Une séquence est représentée de cette manière :

$$\overset{0}{|}\ \omega_1\ \overset{1}{|}\ \omega_2\ \overset{2}{|}\ \omega_3\ \overset{3}{|}\ \omega_4\ \overset{4}{|}\ \omega_5\ \overset{5}{|}\ \omega_6\ \overset{6}{|}\ \omega_7\ \overset{7}{|}\ \omega_8\ \overset{8}{|}\ \omega_9\ \overset{9}{|}$$

Une séquence et une partition prédictive qui prédit chaque ω_i par δ_i sont représentées ensemble ainsi :

$$\overset{0}{|}\ \frac{\omega_1}{\delta_1}\ \overset{1}{|}\ \frac{\omega_2}{\delta_2}\ \overset{2}{|}\ \frac{\omega_3}{\delta_3}\ \overset{3}{|}\ \frac{\omega_4}{\delta_4}\ \overset{4}{|}\ \frac{\omega_5}{\delta_5}\ \overset{5}{|}\ \frac{\omega_6}{\delta_6}\ \overset{6}{|}\ \frac{\omega_7}{\delta_7}\ \overset{7}{|}\ \frac{\omega_8}{\delta_8}\ \overset{8}{|}\ \frac{\omega_9}{\delta_9}\ \overset{9}{|}$$

Si f est une fonction d'un ensemble E dans $\mathbb{R}$, on note $\operatorname{argmax}\{f(x) : x \in E\}$ l'ensemble des éléments de E qui maximisent f.

Dans les algorithmes, l'opérateur $\leftarrow$ représente l'attribution.

2.2 Partitionnement sur séquences

Dans cette partie, nous présentons deux algorithmes de partitionnement sous contrainte d'ordre total. Le premier algorithme [Fis58] réalise le partitionnement dans un cadre très général d'évaluation de partition et est quadratique en temps avec la taille de la séquence. Le second algorithme [GVL98] réalise le partitionnement maximalement prédictif en un temps linéaire avec la taille de la séquence.

2.2.1 Algorithme de Fisher

L'objectif du processus de classification qui nous intéresse est de construire pour k quelconque les k-partitions de Ω qui maximisent une fonction d'évaluation simple P sur l'ensemble des k-partitions dont les classes sont des segments $\omega_{\langle i,j\rangle}$.

$$P(\mathcal{C}) = \sum_{C \in \mathcal{C}} p(C)$$

L'algorithme présenté se base sur le lemme de Fischer généralisé à des fonctions d'évaluation des classes quelconques [Fis58] :

Lemme 2.2.1. *Soient des entiers positifs k, k_1, k_2 tels que $k = k_1 + k_2$. Une k-partition optimale de Ω est l'union d'une k_1-partition optimale de Ω_1 et d'une k_2-partition optimale de Ω_2, avec Ω_1 et Ω_2 des ensembles* appropriés *tels que $\{\Omega_1, \Omega_2\}$ est une 2-partition de Ω.*

En particulier, toute k-partition optimale de Ω est la réunion d'un segment $\{\omega_{\langle i,n\rangle}\}$ et d'une partition en $k-1$ segments de $\omega_{\langle 1,i-1\rangle}$ pour un $i \in \langle 1, n\rangle$ approprié.

$$M_k(\Omega) = \max_{k-1 \leqslant i \leqslant n} (M_{k-1}(\omega_{\langle 1,i\rangle}) + p(\omega_{\langle i+1,n\rangle}))$$

Il est évident qu'il n'y a pas de k-partition de $\omega_{\langle 1,i\rangle}$ si $i < k$.

Cette propriété permet d'utiliser la programmation dynamique de manière efficace pour notre problème. En effet, la 1-partition d'un ensemble est triviale et, pour tout $i \geqslant k$, la meilleure k-partition de $\omega_{\langle 1,i\rangle}$ est obligatoirement constituée de la meilleure $(k-1)$-partition de $\omega_{\langle 1,j-1\rangle}$ à laquelle on ajoute la classe $\{\omega_{\langle j,n\rangle}\}$, pour des $j \in \langle 1,i\rangle$ appropriés. On obtient la formule de construction récursive

$$\boxed{\begin{aligned} \forall i, M_1(\omega_{\langle 1,i\rangle}) = &\quad p(\omega_{\langle 1,i\rangle}) \\ \forall k, \forall i \geqslant k, M_k(\omega_{\langle 1,i\rangle}) = &\quad \max_{k-1\leqslant j<i}(M_{k-1}(\omega_{\langle 1,j\rangle}) + p(\omega_{\langle j+1,i\rangle})) \end{aligned}}$$

Pour construire les k-partitions optimales, l'algorithme consiste alors à calculer successivement $\mathbb{M}_e$ pour e allant de 1 à k.

Le langage de programmation utilisé est le C++, dans lequel la gestion d'adresses mémoire – références – est simple. Une partition peut être vue comme une liste de césures. Comme on veut construire toutes les partitions ex æquo, on représente chaque césure par un entier qui est sa position et par une liste de références à d'autres représentations de césures. Lors de la création d'une césure, la liste est vide. L'ajout d'un élément en tête d'une liste prend le temps de création de cet élément, que l'on suppose constant. On suppose constant le temps de calcul des $p(\omega_{<i,j>})$ [1].

Voici l'algorithme :

Soit `Deb` la référence de la césure en position 0.
Soient, $\forall$`j` $\in \langle$`1,n`$\rangle$, `L[j]`, `L'[j]` des références de têtes de listes.
Soient, $\forall$`j` $\in \langle$`1,n`$\rangle$, `l[j]`, `l'[j]` des valeurs.

- Pour `j` de `1` à `n` :
  - `L[j]` $\leftarrow$ `Deb`
  - `l[j]` $\leftarrow p(\omega_{\langle \mathtt{1,j}\rangle})$
- Pour `j` de `1` à `n` :
  - `L'[j]` $\leftarrow \emptyset$
  - `l'[j]` $\leftarrow$ `0`
- Pour `e` de `2` à `k` :
  - Pour `i` de `e` à `n` :
    - `l'[i]` $\leftarrow \max_{\mathtt{e-1}\leqslant \mathtt{j}<\mathtt{i}}(\mathtt{l[j]} + p(\omega_{\langle \mathtt{j+1,i}\rangle}))$
    - `L'[i]` $\leftarrow$ la référence d'une nouvelle césure en position `i`.
    - Pour `j` de `e-1` à `i-1` :
      - si `l[j]` $+ p(\omega_{\langle \mathtt{j+1,i}\rangle}) =$ `l'[i]`, on ajoute à la liste référencée de la césure référencée par `L'[i]` la référence contenue dans `L[j]`.
  - Pour `i` de `e` à `n` :
    - `L[i]` $\leftarrow$ `L'[i]`
    - `l[i]` $\leftarrow$ `l'[i]`
  - Pour `i` de `e-1` à `n` :
    - `L'[i]` $\leftarrow \emptyset$
    - `l'[i]` $\leftarrow$ `0`
  - On enregistre `L[n]` comme la référence des e-partitions optimales de Ω, et `l[n]` leur valeur.

[1] Si ce n'est pas le cas, ces valeurs peuvent être stockées dans un tableau $n \times n$, et l'accès aux valeurs de ce tableau est constante.

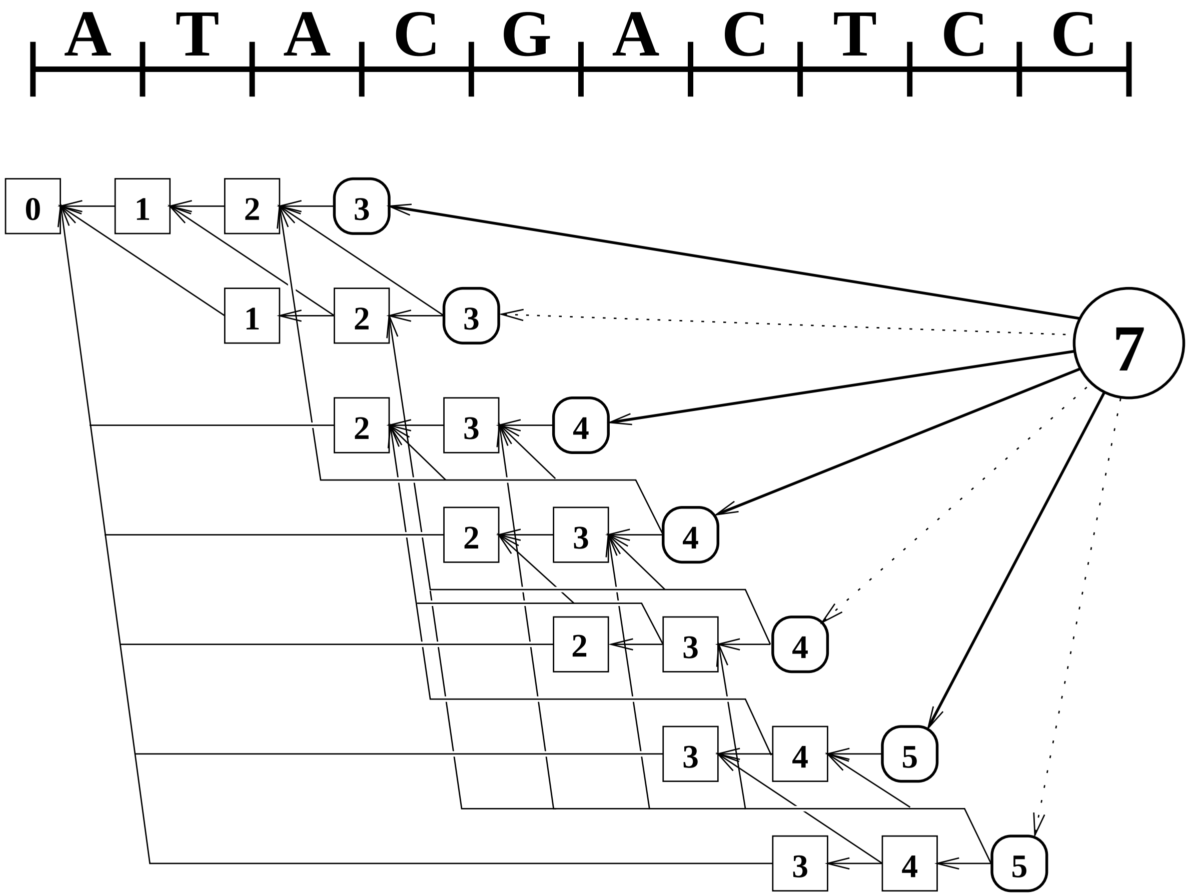


FIG. 2.1 – Principe de fonctionnement de l'algorithme de FISHER. Dans cet exemple, l'évaluation d'une classe est faite par prédiction sur cette classe. Les fonctions de prédiction sont les fonctions caractéristiques. Chaque 3-partition optimale est représentée par un parcours dans le sens des flèches à partir des carrés arrondis. Dans chaque carré est écrite la valeur des partitions qui y aboutissent. La 4-partition de la séquence (qui se termine sur le cercle) est de valeur 7. Elle est construite en recherchant les 3-partitions adéquates des différents préfixes de la séquence. Les flèches en pointillées sont les tentatives faites non retenues, et les quatre flèches pleines et épaisses sont les quatre possibilités retenues.

Pour tout $e \in \langle 2, k\rangle$, pour tout $i \in \langle e, n\rangle$ le calcul de `l[i]` et de la liste `L[i]` nécessite $(i-1-e)$ étapes. Si bien que le temps de calcul à chaque itération de la variable `e` est de l'ordre de $\sum_{i=e}^{n}(i-1-e)+(n-e) = O(n^2)$. Par conséquent, les k-partitions optimales s'obtiennent en un temps en $O(k.|\Omega|^2)$.

Cette méthode a été utilisée dans différents cas de partitionnement de séquences [Gor96c]. Puisque l'évaluation des classes doit juste être simple, elle peut servir à la construction de partitions maximalement prédictives. Cependant, lorsque la séquence est très grande, la complexité quadratique avec la taille de la séquence devient handicapante.

2.2.2 Avec la prédiction

Nous avons alors conçu un autre algorithme qui a l'avantage d'être linéaire avec la taille de la séquence, mais qui est aussi linéaire avec le nombre de prédicteurs admissibles. De plus, il nécessite une connaissance au préalable de ces prédicteurs admissibles.

Pour simplifier la présentation, nous nous intéressons en premier lieu au cas où l'ensemble D des prédicteurs admissibles des classes ne contient pas de prédicteur structuré.

Pour $\delta \in D$, $\pi_\delta(\omega_{\langle i,j\rangle})$ est la prédiction de δ sur $\omega_{\langle i,j\rangle}$. On retrouve la formule suivante.

$$p(\omega_{\langle i,j\rangle}) = \max_{\delta\in D} \pi_\delta(\omega_{\langle i,j\rangle})$$

L'algorithme s'appuie sur l'observation suivante : tous les objets d'une même classe sont comparés à un même prédicteur. Si on note $(i \geqslant k)$

> $\mathbb{M}_k^\delta(\omega_{\langle 1,i\rangle})$ les meilleures k-partitions de l'intervalle $\omega_{\langle 1,i\rangle}$ telles que le prédicteur de la classe possédant ω_i est δ,
> leur évaluation est $M_k^\delta(\omega_{\langle 1,i\rangle})$.

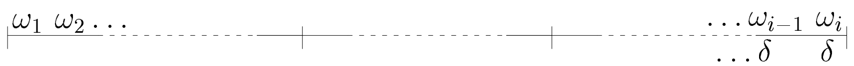
La classe qui possède ω_i d'une k-partition prédictive se voit attribuer un prédicteur, si bien que

$$M_k^D(\omega_{\langle 1,i\rangle}) = \max_{\delta\in D} M_k^\delta(\omega_{\langle 1,i\rangle})$$

Par rapport à l'algorithme précédent, cet algorithme exploite le fait que la fonction d'évaluation d'une classe prédictive est une fonction *locale*, en ce sens qu'il est suffisant de connaître la prédiction en chacune de ses positions pour évaluer la classe, et qu'il n'est nullement besoin d'autres données globales, telles que la taille de la classe. Ainsi l'apport de la prédiction d'un objet à une classe prédictive ne dépend que du prédicteur de cette classe.

Une k-partition de $\omega_{\langle 1,i\rangle}$ pour laquelle le prédicteur de la classe de ω_i est δ répond à l'une de ces deux éventualités,

— soit ω_{i-1} et ω_i appartiennent à la même classe, auquel cas le prédicteur de la classe de ω_{i-1} est aussi δ :



et la prédiction est :

$$M_k^\delta(\omega_{\langle 1,i\rangle}) = \pi_\delta(\omega_i) + M_k^\delta(\omega_{\langle 1,i-1\rangle})$$

— soit ω_{i-1} et ω_i appartiennent à deux classes différentes. Dans ce cas, la k^{e} classe de $\mathbb{M}_k^\delta(\omega_{\langle 1,i\rangle})$ est $\{\omega_i\}$, et les $k-1$ autres classes constituent une $(k-1)$-partition optimale de $\omega_{\langle 1,i-1\rangle}$,

$$\omega_1\ \omega_2 \dots \qquad \dots \omega_{i-1}\ \omega_i$$
$$\dots \delta' \quad \delta$$

et la prédiction est :

$$M_k^\delta(\omega_{\langle 1,i\rangle}) = \pi_\delta(\omega_i) + M_{k-1}^D(\omega_{\langle 1,i-1\rangle}).$$

Le maximum de ces deux formules s'écrit :

$$M_k^\delta(\omega_{\langle 1,i\rangle}) = \pi_\delta(\omega_i) + \max\Big(M_{k-1}^D(\omega_{\langle 1,i-1\rangle}), M_k^\delta(\omega_{\langle 1,i-1\rangle})\Big),$$

qui, puisqu'il n'y a pas de partition en k classes de $\omega_{\langle 1,k-1\rangle}$, se réduit à

$$M_k^\delta(\omega_{\langle 1,k\rangle}) = \pi_\delta(\omega_k) + M_{k-1}^D(\omega_{\langle 1,k-1\rangle})$$

En résumé :

$$\boxed{\begin{aligned}
\forall i, M_1^D(\omega_{\langle 1,i\rangle}) &= \max\nolimits_{\delta\in D} \pi_\delta(\omega_{\langle 1,i\rangle})\\
\forall\delta, \forall k, M_k^\delta(\omega_{\langle 1,k\rangle}) &= \pi_\delta(\omega_k) + M_{k-1}^D(\omega_{\langle 1,k-1\rangle})\\
\forall\delta, \forall k, \forall i > k, M_k^\delta(\omega_{\langle 1,i\rangle}) &= \pi_\delta(\omega_i) +\\
&\qquad \max\left(M_{k-1}^D(\omega_{\langle 1,i-1\rangle}), M_k^\delta(\omega_{\langle 1,i-1\rangle})\right)\\
\forall k, \forall i \geqslant k, M_k^D(\omega_{\langle 1,i\rangle}) &= \max\nolimits_{\delta\in D} M_k^\delta(\omega_{\langle 1,i\rangle})
\end{aligned}}$$

Proposition 2.2.1. *Ces formules permettent de calculer la prédiction des k-partitions maximalement prédictives.*

Pour construire les k-partitions optimales, l'algorithme consiste alors à calculer successivement $\mathbb{M}_e^D$ pour e allant de 1 à k, par l'intermédiaire des $\mathbb{M}_e^\delta$.

Une partition est représentée comme une liste de césures. Appelons **adjacence** une liste de références de césures couplées avec des prédicteurs. Un tel prédicteur est le prédicteur de la classe prédictive qui suit la césure référencée. La référence d'une adjacence est la référence de la tête de la liste concernée. On représente chaque césure par un entier et une liste de références à des adjacences. Cette liste est créée pendant la phase qui correspond au calcul $M_k^D(\omega_{\langle 1,i\rangle}) = \max_{\delta\in D} M_k^\delta(\omega_{\langle 1,i\rangle})$. Sa longueur est au maximum $|D|$. Les césures sont référencées dans l'algorithme ci-après par les variables `L'[i]` et `L[i]`.

En tant que listes, deux adjacences peuvent être incluse l'une dans l'autre. Si une adjacence `A` est référencée dans une césure `C`, ceci signifie que *toutes* les

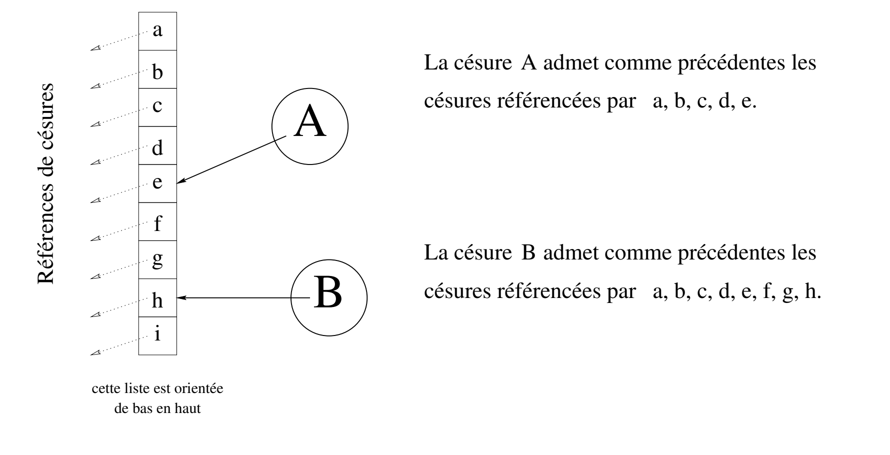


FIG. 2.2 – Fonctionnement des adjacences. L'adjacence pointée par la césure `A` est figurée par la liste orientée de bas en haut en commençant par la lettre `e`. Cette adjacence est incluse dans l'adjacence pointée par la césure `B` et figurée par la liste orientée de bas en haut en commençant par la lettre `h`.

césures qui sont référencées dans `A` *après* le nœud de la liste qui correspond à la tête de `A` sont des césures de partitions optimales passant par `C` (figure 2.2).

Pour $\delta \in D$, les césures extrêmes des $\mathbb{M}_k^\delta$ sont représentées dans une adjacence référencée par la variable $\mathtt{L}^\delta$. Cette liste est modifiée pendant la phase qui correspond au calcul $M_k^\delta(\omega_{\langle 1,i\rangle}) = \pi_\delta(\omega_i) + \max\left(M_{k-1}^D(\omega_{\langle 1,i-1\rangle}), M_k^\delta(\omega_{\langle 1,i-1\rangle})\right)$. Si le maximum n'est atteint que par $M_k^\delta(\omega_{\langle 1,i-1\rangle})$, la liste n'est pas modifiée. Si le maximum n'est atteint que par $M_{k-1}^D(\omega_{\langle 1,i-1\rangle})$, $\mathtt{L}_\delta$ référence une nouvelle adjacence dont le seul élément contient la prédicteur δ couplé à la référence de `L[i-1]`. Si le maximal est atteint par les deux variables, on ajoute en tête de la liste déjà existante le prédicteur δ couplé à la référence de la césure référencée par `L[i-1]`.

Voici l'algorithme :

Soit `Deb` la référence de la césure en position 0.
Soient, $\forall \mathtt{i} \in \langle \mathtt{1}, \mathtt{n} \rangle$, `L[i], L'[i]` des références à des césures.
Soient, $\forall \mathtt{i} \in \langle \mathtt{1}, \mathtt{n} \rangle$, `l[i], l'[i]` des valeurs.
Soient, $\forall \delta \in \mathtt{D}$, $\mathtt{L}^\delta$ des références à des adjacences.
Soient, $\forall \delta \in \mathtt{D}$, $\mathtt{l}^\delta$ des valeurs.

- Pour `i` de `1` à `n` :
  - `L[i]` $\leftarrow$ `Deb`
  - `l[i]` $\leftarrow p(\omega_{\langle \mathtt{1},\mathtt{i}\rangle})$
- Pour `i` de `1` à `n` :
- Pour `e` de `2` à `k` :
  - Pour $\delta \in \mathtt{D}$:
    - $\mathtt{L}^\delta \leftarrow \emptyset$
    - $\mathtt{l}^\delta \leftarrow \mathtt{0}$
  - Pour `i` de `e` à `n` :
    - Pour $\delta \in D$:

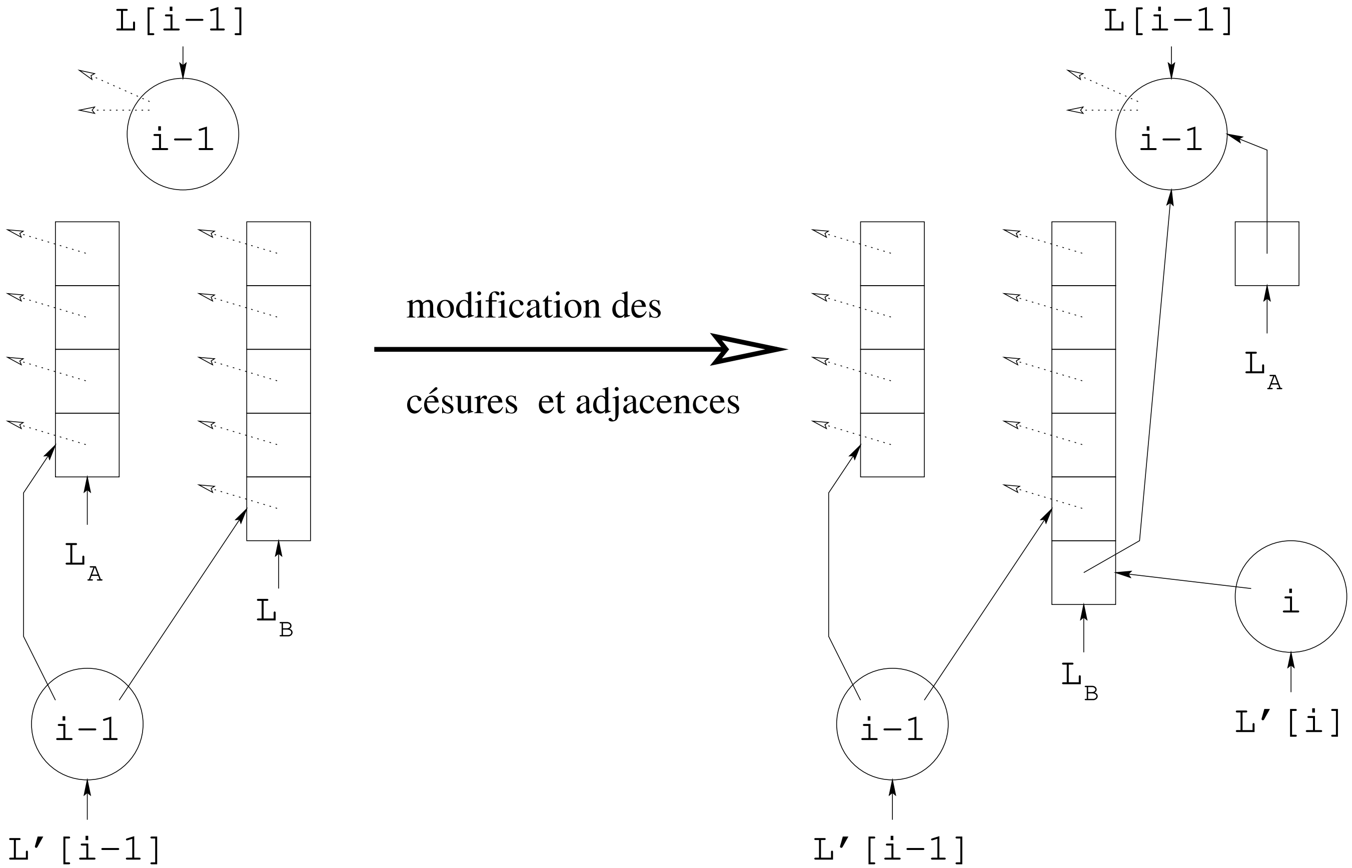


FIG. 2.3 – Modification des césures et des adjacences lors de la lecture d'une position de la séquence. Les adjacences sont représentés comme dans la figure 2.2 par des piles orientées de bas en haut. Les césures (avec leur position) représentées par des cercles sont celles référencées par `L[i-1]`,`L'[i-1]` et `L'[i]`. Dans cet exemple, il n'y a que deux prédicteurs, `A` et `B`. Avant modification les variables $\mathtt{L_A}$ et $\mathtt{L_B}$ référencent des adjacences que référence aussi la césure de `L'[i-1]`. On suppose que $\mathtt{l[i-1]} > \mathtt{l_A}$ et $\mathtt{l[i-1]} = \mathtt{l_B}$. Par conséquent, $\mathtt{L_A}$ référence une nouvelle adjacence qui référence la césure de `L[i-1]` et $\mathtt{L_B}$ référence une nouvelle tête de liste de son adjacence. Cette tête contient la référence de la césure de `L[i-1]`. Ensuite on crée une nouvelle césure de position `i` référencée par `L'[i]`. Si, par exemple, après les calculs de $\mathtt{l_A}$ et $\mathtt{l_B}$, $\mathtt{l_A} < \mathtt{l_B}$, on ajoute dans cette césure seulement la référence contenue dans $\mathtt{L_B}$.

      - si `l[i-1]` $>$ `l`$^\delta$,
        - `L`$^\delta \leftarrow$ `L[i-1]` et δ.
        - `l`$^\delta \leftarrow$ `l[i-1]` $+ \pi_\delta(\omega_{\mathtt{i}})$
      - sinon
        - si `l[i-1]` $=$ `l`$^\delta$, on ajoute en tête de la liste de `L`$^\delta$: `L[i-1]` et δ.
        - `l`$^\delta \leftarrow$ `l`$^\delta + \pi_\delta(\omega_{\mathtt{i}})$
    - `l'[i]` $\leftarrow \max_{\delta\in D}$ `l`$^\delta$
    - `L'[i]` $\leftarrow$ la référence d'une nouvelle césure en position `i`.
    - Pour $\delta \in D$:
      - si `l`$^\delta =$ `l'[i]`, on ajoute à la liste `L'[i]` : `L`$^\delta$.
  - Pour `i` de `e` à `n` :
    - `L[i]` $\leftarrow$ `L'[i]`
    - `l[i]` $\leftarrow$ `l'[i]`
  - On enregistre `L[n]` pour les k-partitions optimales, et `l[n]` leur valeur.

Pour tout $e \in \langle 2, k\rangle$, pour tout $i \in \langle e, n\rangle$, pour tout $\delta \in D$, la modification de `L`$^\delta$ est faite en temps constant (au plus, création d'une tête de liste et modification de sa référence). En outre, pour tout $i \in \langle e, n\rangle$, au plus $|D|$ éléments sont mis dans la liste `L[i]`. Par conséquent, le temps de calcul à chaque itération de la variable `e` est de l'ordre de $\sum_{i=e}^{n} |D| = |D|.(n-e+1)$. Par conséquent, les k-partitions optimales s'obtiennent en un temps en $O(k.|\Omega|.|D|)$.

Lorsque des prédicteurs sont structurés, il faut revoir le calcul des $\mathbb{M}_k^\delta(\omega_{\langle 1,i\rangle})$. En effet, désormais, pour voir s'il est judicieux de prendre δ comme prédicteur de la classe où est ω_i, on doit gérer la phase de δ sur laquelle il faut se caler. Par exemple, si on veut calculer la prédiction de `CT` sur la séquence `ATATATATA`, faut-il estimer que le lissage produit est `CTCTCTCTC` ou bien `TCTCTCTCT` ? Nous résolvons ce dilemne en considérant l'ensemble de toutes les phases.

Si le prédicteur étudié est $\delta_1\delta_2\ldots\delta_j$ avec δ_i prédicteur non structuré, on note $\mathbb{P}_k^{\delta_1\delta_2\ldots\delta_j}(\omega_{\langle 1,i\rangle})$ l'ensemble des k-partitions prédictives de la sous séquence $\omega_{\langle 1,i\rangle}$ telles que le prédicteur de la classe où est ω_i est $\delta_1\delta_2\ldots\delta_j$, **« calé » de telle manière que δ_j est le prédicteur de** ω_i, δ_{j-1} celui de ω_{i-1}, et ainsi de suite en réitérant le motif si la classe est assez longue, de telle sorte que δ_j est aussi le prédicteur de ω_{i-j+1}.

$\mathbb{M}_k^{\delta_1\delta_2\ldots\delta_j}(\omega_{\langle 1,i\rangle})$ est l'ensemble des k-partitions prédictives optimales de $\mathbb{P}_k^{\delta_1\delta_2\ldots\delta_j}(\omega_{\langle 1,i\rangle})$.

$M_k^{\delta_1\delta_2\ldots\delta_j}(\omega_{\langle 1,i\rangle})$ est la prédiction de ces partitions.

De même que ci-dessus, il y a deux cas pour le calcul de $M_k^{\delta_1\delta_2\ldots\delta_j}(\omega_{\langle 1,i\rangle})$:

— soit ω_i et ω_{i-1} sont dans la même classe, et donc δ_{j-1} est comparé à ω_{i-1} au sein du motif $\delta_1\delta_2\ldots\delta_j$

$$\begin{array}{lcccc} \omega_1\ \omega_2\ldots & | & | & \ldots\omega_{i-2}\ \omega_{i-1}\ \omega_i \\ & & & \ldots\delta_{j-2}\ \delta_{j-1}\ \delta_j \end{array}$$

et la prédiction est

$$M_k^{\delta_1\delta_2\ldots\delta_j}(\omega_{\langle 1,i\rangle}) = \pi_{\delta_j}(\omega_i) + M_k^{\delta_j\delta_1\delta_2\ldots\delta_{j-1}}(\omega_{\langle 1,i-1\rangle}),$$

— soit ω_{i-1} et ω_i ne sont pas dans la même classe

et la prédiction est

$$M_k^{\delta_1\delta_2\ldots\delta_j}(\omega_{\langle 1,i\rangle}) = \pi_{\delta_j}(\omega_i) + M_{k-1}^{D}(\omega_{\langle 1,i-1\rangle}).$$

Ainsi, en prenant le maximum de ces deux possibilités, en rappelant que ρ est l'opérateur de permutation circulaire tel que

$$\rho(a_1a_2a_3\ldots a_p) = a_pa_1a_2\ldots a_{p-1}$$

on a

$$M_k^{\delta_1\delta_2\ldots\delta_j}(\omega_{\langle 1,i\rangle}) = \pi_{\delta_j}(\omega_i) + \max(M_k^{\rho(\delta_1\delta_2\ldots\delta_j)}(\omega_{\langle 1,i-1\rangle}), M_{k-1}^{D}(\omega_{\langle 1,i-1\rangle}))$$

En résumé :

$$\boxed{\begin{aligned}
\forall i, M_1^D(\omega_{\langle 1,i\rangle}) =&\ \ \max\nolimits_{\delta\in D} \pi_\delta(\omega_{\langle 1,i\rangle})\\
\forall \delta, \forall k, M_k^\delta(\omega_{\langle 1,k\rangle}) =&\ \ \pi_\delta(\omega_k) + M_{k-1}^D(\omega_{\langle 1,k-1\rangle})\\
\forall \delta, \forall k, \forall i > k, M_k^\delta(\omega_{\langle 1,i\rangle}) =&\ \ \pi_\delta(\omega_i) +\\
&\qquad \max\left(M_{k-1}^D(\omega_{\langle 1,i-1\rangle}), M_k^{\rho(\delta)}(\omega_{\langle 1,i-1\rangle})\right)\\
\forall k, \forall i \geqslant k, M_k^D(\omega_{\langle 1,i\rangle}) =&\ \ \max\nolimits_{\delta\in D} M_k^\delta(\omega_{\langle 1,i\rangle})
\end{aligned}}$$

Pour que l'algorithme présenté en 2.2.2 soit utilisable dans ce cas, il suffit d'opérer une permutation circulaire qui correspond à l'opérateur ρ sur les références des variables $\mathtt{L}_\delta$ et $\mathtt{l}_\delta$. Le temps de cette opération est de l'ordre de $|D|$, si bien que la complexité de l'algorithme n'est pas augmentée. En outre, pour homogénéiser les membres d'une adjacence, si elle est référencée par δ au moment où un membre césure lui est ajouté, on met le prédicteur $\underline{\delta}$ dans ce membre.

2.3 Réduction des k-partitions optimales

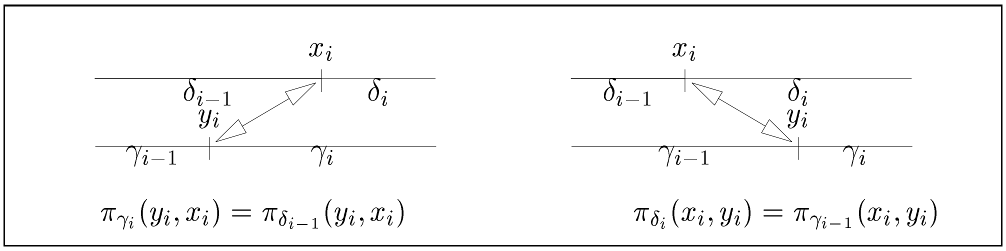
Lorsque l'évaluation d'une classification admet des valeurs non entières, on peut espérer qu'il y ait peu de partitions optimales ex æquo. Mais quand les valeurs de la fonction d'évaluation sont entières, les k-partitions optimales peuvent en revanche être très nombreuses. Un cas particulièrement courant est le suivant : pour partager la séquence `AACACC` en deux classes avec `A` et `C` comme prédicteurs, deux 2-partitions sont optimales : `AAAACC` et `AACCCC`, c'est-à-dire qu'il y a une ambiguïté quant à la frontière entre deux classes, due à l'interversion locale des lettres par rapport à l'ordre des prédicteurs optimaux des classes.

Dans ce cas, nous allons, en établissant entre les partitions prédictives optimales une relation d'équivalence, chercher à réduire le nombre de partitions à traiter. Pour que cette relation soit intéressante, il faut qu'elle permette, à partir d'un seul représentant de chaque classe d'équivalence, de retrouver tous ses représentants par un algorithme « simple ». De plus, il faut que ces classes soient facilement constructibles, ce qui est réalisable par une modification de l'algorithme du 2.2.2.

2.3.1 Relation d'équivalence

Définition. *Soit* $X = ((x_i, \delta_i))_{i \in \langle 0,k-1 \rangle}$ *et* $Y = ((y_i, \gamma_i))_{i \in \langle 0,k-1 \rangle}$ *des éléments de* $\mathbb{P}_k^D(\Omega)$*, on note* $\sim$ *la relation définie ainsi :*

$$X \sim Y \iff \forall i \in \langle 1, k-1 \rangle, \left| \begin{array}{l} \pi_{\delta_i}(x_i, y_i) = \pi_{\gamma_{i-1}}(x_i, y_i) \\ \textit{ou } \pi_{\gamma_i}(x_i, y_i) = \pi_{\delta_{i-1}}(x_i, y_i) \end{array} \right.$$



Cette définition signifie que deux k-partitions prédictives sont dans une même classe si les prédictions des intervalles entre leurs césures de même rang sont égales.

On rappelle que si une classe prédictive est notée $(C, \underline{\delta})$, cela signifie que le prédicteur de cette classe coïncide parfaitement sur cette classe avec le mot $\underline{\delta}$ répété à partir de la première position de la *séquence* (voir 1.2.3.1).

Lorsque l'on considère des partitions prédictives sur une séquence, on peut, de même que ci-dessus, considérer $\mathbb{P}_k^{\underline{\delta_0},\dots,\underline{\delta_{k-1}}}(\Omega)$ l'ensemble des k-partitions prédictives de Ω pour lesquelles le prédicteur de la première classe (où est la première position) est $\underline{\delta_0}$, celui de la deuxième $\underline{\delta_1}$, et ainsi de suite celui de la k^{e} $\underline{\delta_{k-1}}$. L'ensemble $\mathbb{M}_k^{\underline{\delta_0},\dots,\underline{\delta_k}}(\Omega)$ est l'ensemble des optimaux de $\mathbb{P}_k^{\underline{\delta_0},\dots,\underline{\delta_{k-1}}}(\Omega)$.

$\sim$ est une relation d'équivalence sur $\mathbb{M}_k^{\underline{\delta_0},\dots,\underline{\delta_{k-1}}}(\Omega)$ car :

- $X \sim X$ car $\forall i, \pi_{\underline{\delta_i}}(x_i, x_i) = 0 = \pi_{\underline{\delta_{i-1}}}(x_i, x_i)$
- Si $X \sim Y$: $\forall i, \pi_{\underline{\delta_i}}(x_i, y_i) = \pi_{\underline{\delta_{i-1}}}(x_i, y_i)$. Donc $\forall i, \pi_{\underline{\delta_i}}(y_i, x_i) = \pi_{\underline{\delta_{i-1}}}(y_i, x_i)$ et $Y \sim X$.
- Si $X \sim Y$ et $Y \sim Z$: $\forall i, \pi_{\underline{\delta_i}}(x_i, y_i) = \pi_{\underline{\delta_{i-1}}}(x_i, y_i)$ et $\pi_{\underline{\delta_i}}(y_i, z_i) = \pi_{\underline{\delta_{i-1}}}(y_i, z_i)$ donc, par la somme, $\pi_{\underline{\delta_i}}(x_i, z_i) = \pi_{\underline{\delta_{i-1}}}(x_i, z_i)$ et $X \sim Z$.

Si $X \in \mathbb{M}_k^{\underline{\delta_0},\dots,\underline{\delta_k}}(\Omega)$, on note $\tilde{X}$ sa classe d'équivalence dans $M_k^{\underline{\delta_0},\dots,\underline{\delta_k}}$.

2.3.2 Reconstruction

Avant d'aborder la construction des classes d'équivalence, il faut s'assurer de pouvoir reconstruire simplement tous les éléments d'une classe à partir d'un seul de ses représentants.

Soit $X = ((x_i, \underline{\delta_i}))_{i \in \langle 0,k-1 \rangle}$ ce représentant.

Pour $i \in \langle 1, k-1 \rangle$, on note :

$$X_i = \{x \in \langle 0, n \rangle : \pi_{\underline{\delta_i}}(x_i, x) = \pi_{\underline{\delta_{i-1}}}(x_i, x)\}$$

Proposition 2.3.1.

$$\begin{aligned}\tilde{X} = \quad & \{((y_0, \underline{\delta_0}), \ldots, (y_{k-1}, \underline{\delta_{k-1}})) \in \mathbb{P}_k^{\underline{\delta_0}, \ldots, \underline{\delta_{k-1}}}(\Omega) : \\ & (y_0, \ldots, y_{k-1}) \in X_0 \times \ldots \times X_{k-1} \& \forall i, y_{i-1} < y_i\}\end{aligned}$$

Pour vérifier cette proposition, on utilise le résultat suivant :

Lemme 2.3.1. *Si* $X = ((x_i, \underline{\delta_i}))_{i \in \langle 0,k-1 \rangle} \in \mathbb{M}_k^{\underline{\delta_0}, \ldots, \underline{\delta_{k-1}}}(\Omega)$ *et* $Y = ((y_i, \underline{\delta_i}))_{i \in \langle 0,k-1 \rangle} \in \mathbb{P}_k^{\underline{\delta_0}, \ldots, \underline{\delta_{k-1}}}(\Omega)$ *avec* $X \sim Y$*, alors* $Y \in \mathbb{M}_k^{\underline{\delta_0}, \ldots, \underline{\delta_{k-1}}}(\Omega)$.

■ **Démonstration du lemme 2.3.1** :

$P(Y) = \sum_{i=0}^{n-1} \pi_{\underline{\delta_i}}(y_i, y_{i+1})$

$$\text{Or,} \qquad \begin{aligned}\pi_{\underline{\delta_i}}(y_i, y_{i+1}) &= \pi_{\underline{\delta_i}}(y_i, x_i) + \pi_{\underline{\delta_i}}(x_i, x_{i+1}) + \pi_{\underline{\delta_i}}(x_{i+1}, y_{i+1}) \\ &= \pi_{\underline{\delta_i}}(y_i, x_i) + \pi_{\underline{\delta_i}}(x_i, x_{i+1}) + \pi_{\underline{\delta_{i+1}}}(x_{i+1}, y_{i+1})\end{aligned}$$

car $X \sim Y$.

Donc ($x_0 = y_0$ et $x_k = y_k$),

$$\begin{aligned}P(Y) = \; & \pi_{\underline{\delta_0}}(x_0, x_1) + \pi_{\underline{\delta_1}}(x_1, y_1) + \\ & \pi_{\underline{\delta_1}}(y_1, x_1) + \pi_{\underline{\delta_1}}(x_1, x_2) + \pi_{\delta_2}(x_2, y_2) + \\ & \vdots \\ & + \pi_{\underline{\delta_{k-2}}}(y_{k-2}, x_{k-2}) + \pi_{\underline{\delta_{k-2}}}(x_{k-2}, x_{k-1}) + \pi_{\underline{\delta_{k-1}}}(x_{k-1}, y_{k-1}) + \\ & \pi_{\underline{\delta_{k-1}}}(y_{k-1}, x_{k-1}) + \pi_{\underline{\delta_{k-1}}}(x_{k-1}, x_k) \\ = \; & \pi_{\underline{\delta_0}}(x_0, x_1) + \pi_{\underline{\delta_1}}(x_1, x_2) + \\ & \vdots \\ & + \pi_{\underline{\delta_{k-2}}}(x_{k-2}, x_{k-1}) + \pi_{\underline{\delta_{k-1}}}(x_{k-1}, x_k) \\ = \; & P(X)\end{aligned}$$

Comme $X \in \mathbb{M}_k^{\underline{\delta_0}, \ldots, \underline{\delta_{k-1}}}(\Omega)$, $Y \in \mathbb{M}_k^{\underline{\delta_0}, \ldots, \underline{\delta_{k-1}}}(\Omega)$.

□

■ **Démonstration de la proposition 2.3.1** :

Notons B l'ensemble qui est à droite de l'égalité.

Par définition, si $Y \in \tilde{X}$ alors $Y \in B$.

Si $Y \in B$, $X \sim Y$. Donc, grâce au lemme, $Y \in \mathbb{M}_k^{\underline{\delta_0}, \dots, \underline{\delta_{k-1}}}(\Omega)$ et donc $Y \in \tilde{X}$.

□

Grâce à cette proposition, $\tilde{X}$ se construit sans problème à partir de X.

Définition. *Soit* $Y = ((y_i, \underline{\delta_i}))$ *dans* $\tilde{X}$, Y *est dit justifié à gauche (resp. à droite) dans* $\tilde{X}$ *si,* $\forall ((y'_i, \underline{\delta_i})) \in \tilde{X}$ *et* $\forall j \in \langle 0, k\rangle, y_j \leqslant y'_j$ *(resp.* $y_j \geqslant y'_j$*).*

La construction de $\tilde{X}$ justifie l'existence de ces partitions.

2.3.3 Partitionnement en classes d'équivalences

Maintenant, par une adaptation de l'algorithme 2.2.2, nous allons voir comment construire directement les classes d'équivalence, ou plutôt un seul des représentants de chaque classe d'équivalence.

Par rapport à l'algorithme précédent, il s'agit de tenir compte de l'existence de plusieurs représentants – doublons – d'une même classe d'équivalence. Il faut faire un choix parmi ces doublons. Ce choix est déterminé par la sélection de la partition prédictive qui a les plus petites positions de césures, de manière à obtenir celle qui est justifiée à gauche (voir définition 2.3.2).

Nous utilisons de nouveau les variables `L[i]` et consœurs pour représenter les césures, dont la représentation contient un entier pour la position et une liste de références d'adjacences (voir page 40). Les variables $\mathtt{L}^\delta$ correspondent à des références d'adjacences. Le prédicteur qui est dans chaque élément de l'adjacence de $\mathtt{L}_\delta$ est $\underline{\delta}$.

`L[i]` est construite à partir des $\mathtt{L}_\delta$, pour les différents δ de `D`. Pour deux prédicteurs δ et δ' différents, forcément, les partitions prédictives issues de $\mathtt{L}_\delta$ et de $\mathtt{L}_{\delta'}$ ne sont pas dans la même classe d'équivalence. Par conséquent, la construction de `L[i]` peut être faite comme précédemment.

En ce qui concerne $\mathtt{L}_\delta$, les doublons étaient gérés directement en ajoutant dans l'adjacence de $\mathtt{L}_\delta$ la référence à `L[i-1]`. Désormais, il faut éviter les doublons qui peuvent résulter de cette gestion. Le problème se pose quand $\mathtt{l}_\delta = \mathtt{l[i-1]}$.

On suppose que les partitions prédictives qui aboutissent à la césure `L[i-1]` et celles qui aboutissent aux césures de $\mathtt{L}_\delta$ sont toutes justifiées à gauche dans leur classe d'équivalence. Sur le segment $\omega_{\langle 1,i-1\rangle}$, deux partitions prédictives justifiées à gauche $((x_j, \underline{\delta_j}))_{j\in\langle 0,e-2\rangle}$ et $((y_j, \underline{\gamma_j}))_{j\in\langle 0,e-2\rangle}$ sont dans la même classe d'équivalence si pour tout j dans $\langle 0, e-2\rangle$ on a simultanément $x_j = y_j$ et $\underline{\delta_j} = \underline{\gamma_j}$. `L[i-1]` correspond à au moins toutes les $(e-1)$-partitions prédictives justifiées à gauche de $\omega_{\langle 1,i-1\rangle}$ qui terminent par $(x_{e-2}, \underline{\delta_{e-2}}$. De même, $\mathtt{L}_\delta$ doit correspondre à au moins toutes les partitions justifiées à gauche de $\omega_{\langle 1,i-1\rangle}$ qui possèdent la classe $(y_{e-2}, \underline{\gamma_{e-2}})$. Ainsi, il suffit de tester $x_{e-2} = y_{e-2}$ et $\underline{\delta_{e-2}} = \underline{\gamma_{e-2}}$.

On ne garde alors que les éléments des adjacences de `L[i-1]` qui ne sont pas dans une adjacence d'une césure référencée par $\mathtt{L}_\delta$ pour éviter les doublons.

Le fait de garder les éléments de $\mathtt{L}_\delta$ garantit que les partitions conservées sont celles qui sont justifiées à gauche dans leur classe.

Ainsi, on construit une nouvelle césure de position $i-1$ qui référence des adjacences (autant d'adjacences que de prédicteurs) dont les éléments sont sélectionnés comme indiqué ci-dessus à partir des éléments des adjacences de `L[i-1]`. Dans ce processus, on compare les listes des adjacences de `L[i-1]` avec les listes d'éléments de $\mathtt{L}_\delta$. Cependant, on ne sait pas quelle est la taille des listes de `L[i-1]`, ni celle de la liste de $\mathtt{L}_\delta$, ni les tailles des listes des éléments de $\mathtt{L}_\delta$, ce qui détruit l'aspect linéaire de la complexité de l'algorithme.

En dehors de la copie des césures et la sélection dans celles-ci des références de manière à empêcher la redondance, l'algorithme est identique à celui présenté en page 41.

2.4 Lexiques de prédiction

Pour pratiquer des partitionnements de séquences de lettres avec des ensembles de prédicteurs variables, nous définissons quelles sont les fonctions de prédiction dont l'on se servira. Nous construisons également un langage de déclaration d'ensembles de prédicteurs admissibles. Pour chaque ensemble de prédicteurs, un lexique (`L`) est défini dans ce langage.

Les exemples des sections suivantes correspondent à $D = \{A, C, G, T\}$.

2.4.1 Prédicteurs simples

Le type basique d'élément de `D` est une lettre de la séquence. Ainsi, pour tous les éléments $\{\mathtt{d}_1, \mathtt{d}_2, \ldots, \mathtt{d}_1\} \subset D$ que l'on veut admettre, on note :

L = $\mathtt{d}_1\ \mathtt{d}_2\ \ldots\ \mathtt{d}_1$
et donc
D = $\{\mathtt{d}_1, \mathtt{d}_2, \ldots, \mathtt{d}_1\}$

Pour une lettre ω_i de la séquence, la prédiction par un prédicteur de ce type est la fonction caractéristique, c'est-à-dire :

$\pi_\delta(\omega_i) = \mathbb{1}_{\delta=\omega_i}$

Dans ce cas, la prédiction d'une lettre sur un segment est le nombre d'occurrences de cette lettre dans le segment.

Si toutes les lettres de la séquence sont tolérées comme prédicteurs, il peut être pénible de toutes les décrire dans `L`. La notation `*` permet d'y remédier. Dans l'exemple :

```
L = *
D = {A, C, G, T}
```

2.4.2 Utilisation des coûts et poids

On peut estimer que, dans un certain contexte, la prédiction de tel prédicteur a plus de valeur la prédiction de tel autre, et qu'il doit être gratifié d'un poids plus important. Si ce prédicteur est `d`, pour lequel la prédiction est valorisée par le facteur `x`, on écrit dans `L` : `d(x)` et $\pi_{\mathtt{d(x)}} = \mathtt{x}.\pi_{\mathtt{d}}$ [2].

Par exemple, si la prédiction de la lettre A est trois fois plus importante que celle des autres lettres, on écrit :

`L = A(3) C G T`

et donc

`D = {A(3), C, G, T}`

$\pi_{\mathtt{A(3)}}(\omega_i) = 3.\mathbb{1}_{\mathtt{A}=\omega_i}$

À l'opposé, si la prédiction d'une lettre est pénalisante, on peut lui assigner un poids négatif.

Dans la déclaration du lexique, lorsqu'un prédicteur est déclaré plusieurs fois avec des facteurs différents, on résout l'ambiguïté en ne gardant que le facteur maximum. Par exemple

```
L = A(3) *
D = {A(3), C, G, T}
```

2.4.3 Prédicteurs indifférenciés

Parfois, il peut y avoir des prédicteurs non structurés qui, vus de l'expérimentateur, ne sont pas différenciés, et doivent être tous considérés ensemble. Ainsi, un prédicteur doit être assigné à l'ensemble de ces prédicteurs. La prédiction de ce nouveau prédicteur sera le maximum des prédictions de ces prédicteurs.

Si $\{d_1, d_2, \dots, d_l\} \subset D$ sont des prédicteurs non structurés, on note $[\mathtt{d_1 d_2 \dots d_l}]$ dans `L` le fait qu'ils sont indifférenciés, et $\pi_{[\mathtt{d_1 d_2 \dots d_l}]} = \max(\pi_{\mathtt{d_1}}, \pi_{\mathtt{d_2}}, \dots, \pi_{\mathtt{d_l}})$.

Par exemple, si `A` et `T` sont indifférenciés d'une part, et que de même `C` muni du poids 2 et `G` sont indifférenciés d'autre part, on note :

`L = [AT] [C(2)G]`

et donc

`D = {[AT], [C(2)G]}`

$$\pi_{\mathtt{[AT]}}(\omega_i) = \max(\mathbb{1}_{\mathtt{A}=\omega_i}, \mathbb{1}_{\mathtt{T}=\omega_i})$$
$$\pi_{\mathtt{[C(2)G]}}(\omega_i) = \max(2.\mathbb{1}_{\mathtt{C}=\omega_i}, \mathbb{1}_{\mathtt{G}=\omega_i})$$

Si toutes les lettres de la séquence sont indifférenciées, au lieu d'avoir à les écrire toutes, on utilise le caractère joker noté point d'interrogation (`?`). Si, à l'origine, ne rien différencier ne présente pas d'intérêt, l'élaboration du lexique rend ce joker utile. Par exemple :

```
L = ?(2) A(3) [C(-1)G(3)]
D = {[A(2)C(2)G(2)T(2)], A(3), [C(-1)G(3)]}
```

En outre, pour simplifier l'écriture de prédicteurs comme `[A(2)C(2)G(2)]`, on note dans le lexique `[ACG](2)`. Ainsi, l'écriture $[\mathtt{d_1(x_1)d_2(x_2)\dots d_l(x_l)}]\mathtt{(y)}$ est résolue comme : $[\mathtt{d_1(y.x_1)d_2(y.x_2)\dots d_l(y.x_l)}]$

[2] `d(1)` revient à `d`.

2.4.4 Prédicteurs contextuels

Si on veut prédire selon la présence du mot $d_1 d_2 \ldots d_l$, on note cela

"$\mathtt{d_1 d_2 \ldots d_l}$" dans L et
$\pi_{"\mathtt{d_1 \ldots d_l}"}(\omega_i) = \pi_{\mathtt{d_1}}(\omega_i).\mathbb{1}_{(\pi_{\mathtt{d_2}}(\omega_{i+1})>0)\wedge\ldots\wedge(\pi_{\mathtt{d_l}}(\omega_{i+l-1})>0)}$

Par exemple, si on veut classer une séquence selon les mots de deux lettres qui s'y trouvent :

```
L  = "**"
D  = {"AA", "AC", "AG", "AT", "CA", "CC", "CG", "CT",
   "GA", "GC", "GG", "GT", "TA", "TC", "TG", "TT"}
```

Ces trois notations peuvent être utilisées ensemble pour permettre une grande variété de prédicteurs non structurés. Cependant, d'un point de vue complexité, comme le temps pour comparer un prédicteur simple n'est pas nul, même s'il est très faible, il est clair que le coût en temps de l'exécution est directement lié à la longueur des prédicteurs.

2.4.5 Prédicteurs structurés

Si on veut prédire une classe selon un motif, l'ensemble des prédicteurs admissibles doit être clos pour l'opérateur ρ.

Pour signifier qu'un motif $d_1 d_2 \ldots d_l$ est un prédicteur structuré admissible, on note, avec $\mathtt{d_j}$ un prédicteur non structuré écrit grâce aux notations présentées ci-dessus

$\mathtt{d_1 d_2 \ldots d_l}$ dans L, et
D possède alors les $\rho^i(\mathtt{d_1 d_2 \ldots d_l})$ pour tout i.

La taille de D n'est pas forcément augmentée de l, puisque le prédicteur structuré peut avoir une période interne, auquel cas c'est le plus petit motif qui sera retenu. Par exemple, du prédicteur CTCT, D n'a besoin de posséder que CT et TC.

Par exemple, si on veut opérer une partition avec toutes les descriptions de deux lettres, il faudra gérer 16 variables de D :

```
L = **
D = {A, AC, AG, AT, CA, C, CG, CT, GA, GC, G, GT, TA, TC, TG, T}
```

Si en outre, on veut en même temps manipuler toutes les descriptions de trois lettres, il faudra gérer $4^2 + 4^3 - 4 = 76$ éléments de D :

```
L = ** ***
D = {A, TA, AT, CA, AC, GA, AG, T, CT, TC, GT, TG, C, GC, CG, G,
      AAT, TAA, ATA, AAC, CAA, ACA, AAG, GAA, AGA, ATT, TAT, TTA,
      ATC, CAT, TCA, ATG, GAT, TGA, ACT, TAC, CTA, ACC, CAC, CCA,
      ACG, GAC, CGA, AGT, TAG, GTA, AGC, CAG, GCA, AGG, GAG, GGA,
      TTC, CTT, TCT, TTG, GTT, TGT, TCC, CTC, CCT, TCG, GTC, CGT,
      TGC, CTG, GCT, TGG, GTG, GGT, CCG, GCC, CGC, CGG, GCG, GGC}
```

Des prédicteurs de classes appartiennent à plusieurs ensembles de motifs, et il n'est pas besoin de les considérer séparément dans l'ensemble des prédicteurs. Pour cette raison, écrire `L = **** **` donne exactement le même ensemble de prédicteurs qu'écrire `L = ****`.

Pour la même raison, on peut être sûr qu'une k-partition prédictive optimale construite avec le lexique `L = **` a forcément une moins bonne prédiction qu'une k-partition prédictive optimale construite avec le lexique `L = ****`. Ceci est juste une question de tailles de motifs multiples les unes des autres. Ainsi, on ne peut, à partir de la simple comparaison d'ensembles de prédicteurs, absolument rien déduire de la comparaison des partitionnements créés sur un lexique `L = ***` et `L = **`. Sur une séquence où les lettres sont plus soumises à des répétitions de période deux, le lexique `L = **` est plus performant, mais si les lettres de cette séquence sont plus soumises à des répétitions de période trois, le lexique `L = ***` permettra de meilleures prédictions à nombre de classes égal. En particulier, la taille du lexique n'est pas forcément un critère de meilleure prédiction.

Le lexique peut comporter des parenthèses pour faciliter l'écriture de certains ensembles de prédicteurs. Par exemple, si on veut faire alterner un prédicteur quelconque avec `A` et `T` indifférenciés, ou bien avec le mot de deux lettres dont la première est un `G` et la deuxième un `C` ou un `T` indifférenciés, on note :

```
L = *([AT] "G[CT]")
D = {[AT]A, A[AT], [AT]T, T[AT], [AT]C, C[AT], [AT]G, G[AT],
    "G[CT]"A, A"G[CT]", "G[CT]"T, T"G[CT]",
    "G[CT]"C, C"G[CT]", "G[CT]"G, G"G[CT]"}
```

2.4.6 Exemples d'utilisation

Ci-dessus, nous avons défini la fonction élémentaire de prédiction comme la fonction caractéristique. En utilisant les notations définies ci-dessus, il est possible de définir des prédictions plus élaborées. Certains ensembles très simples d'écriture correspondent à des types de prédictions intéressantes, dont voici quelques exemples.

2.4.6.1 Matrices de similarité

Si la relation entre les prédicteurs et les lettres n'est plus une simple relation d'identité mais une relation de similarité, il serait intéressant de pouvoir intégrer cette similarité dans les prédictions des lettres. Ceci est aisément faisable, en utilisant des prédicteurs ambigus munis des facteurs de la matrice de similarité. Par exemple, si la matrice de similarité est

	A	C	G	T
A	1	0.3	0.4	0.5
C	0.3	1	0.7	0.3
G	0.4	0.7	1	0.2
T	0.5	0.3	0.2	1

il suffit d'assimiler chaque prédicteur admissible à la colonne de la matrice de similarité qui lui correspond. Ainsi, les prédicteurs admissibles décrits dans le tableau ci-dessus sont énumérés dans le lexique

```
L=[A(1)C(0.3)G(0.4)T(0.5)] [A(0.3)C(1)G(0.7)T(0.3)]
  [A(0.4)C(0.7)G(1)T(0.2)] [A(0.5)C(0.3)G(0.2)T(1)]
```

2.4.6.2 Chaînes de Markov

Comme nous l'avons présenté en 1.3.2.3, on peut établir un lien entre le calcul de la probabilité d'une séquence selon une chaîne de Markov à états cachés et la prédiction, la différence tenant essentiellement des contraintes de passage entre les différents états.

En ôtant cette différence, on peut concevoir qu'une séquence soit générée par une suite de k chaînes de Markov sur k segments d'une k-partition. Si chaque classe C_i est décrite par la chaîne de Markov B_i, en notant $P(C_i|B_i,\Omega)$ la probabilité que C_i soit générée par B_i dans Ω, la probabilité que la séquence soit générée par cette suite est

$$\prod_{i=1}^{k} P(C_i|B_i,\Omega)$$

Maximiser ce produit sur l'ensemble des k-partitions revient à maximiser la somme des logarithmes de ces probabilités[3]. Or, si on note $P(\omega_i|B,\Omega)$ la probabilité que la lettre en ω_i est générée par la chaîne de Markov B au sein de Ω (car cette probabilité dépend d'autant de lettres qui précèdent ω_i que l'ordre de B),

$$\begin{aligned}\log P(\omega_{\langle i,j\rangle}|B,\Omega) &= \log \textstyle\prod_{s=i}^{j} P(\omega_s|B,\Omega)\\ &= \textstyle\sum_{s=i}^{j} \log P(\omega_s|B,\Omega)\end{aligned}$$

Or, la probabilité d'occurence d'une lettre dans une séquence au sein d'un processus markovien est une fonction locale, en ce sens qu'elle ne dépend que d'un nombre borné de positions qui précèdent la lettre en question, au même titre que la prédiction par un prédicteur contextuel. Ainsi, il est possible de construire un prédicteur qui correspond aux probabilités d'une chaîne de Markov. Par exemple, si la chaîne est d'ordre 1 sur l'alphabet A, C, G, T, on note $B_{ef} = P(\omega_i = f|\omega_{i-1} = e)$ au sein de la chaîne B. Dans ce cas, le prédicteur écrit dans le lexique par

["AA"$(\log B_{AA})$"AC"$(\log B_{AC})$..."GT"$(\log B_{GT})$"TT"$(\log B_{TT})$]

prédit une classe $\omega_{\langle i,j\rangle}$ avec la valeur $\log P(\omega_{\langle i+1,j+1\rangle}|B,\Omega)$[4].

Par conséquent, si on se donne initialement plusieurs chaînes de Markov, on peut partager au mieux la séquence – au sens où sa probabilité d'être générée

[3] On suppose ces probabilités sont toutes non nulles.

[4] Ce décalage est dû au fait que les prédicteurs structurés concernent les lettres qui suivent la position considérée, alors que les chaînes de Markov ne regardent que les positions qui la précèdent. Si cela pose problème, il suffit de considérer la séquence dans le sens opposé, et d'inverser les prédicteurs structurés ("AC" au lieu de "CA").

est la plus grande – en k segments par partitionnement maximalement prédictif grâce à un lexique formé à partir de ces chaînes.

2.4.6.3 Segments sans prédicteurs

Sur une séquence, une partition maximalement prédictive opère un partage en classes prédictives, et cherche à assigner des prédicteurs à l'intégralité des lettres de la séquence. En cela réside un *a priori* sur la « bonne » organisation de cette séquence. En effet, en supposant deux zones voisines bien adaptées à la prédiction, il peut y avoir entre elles une zone intermédiaire sur laquelle aucun prédicteur ne s'adapte de façon satisfaisante. Ainsi, il faudrait que le partitionnement puisse détecter de telles zones intermédiaires. Cependant, il faut pouvoir les appréhender précisément.

Dans le contexte de la prédiction, une classe est jugée mauvaise si la meilleure prédiction sur l'ensemble des prédicteurs proposés n'est pas acceptable. Cette prédiction est dite *mauvaise*. Maximiser la prédiction revient à chercher à tout prédire, même lorsqu'il n'y a rien à prédire dans la classe concernée. Il faut alors aussi pénaliser les mauvaises prédictions. Une mauvaise zone est telle que pour tout prédicteur, la *proportion* de prédiction est en deçà d'un certain seuil. Soit τ cette proportion ; il faudrait que toute classe prédictive C ait une proportion de prédiction supérieure à τ, c'est-à-dire $\frac{\pi(C)}{|C|} > \tau$.

Pour cela, on utilise le prédicteur[5] joker `?` auquel le facteur τ est assigné, c'est-à-dire que pour toute lettre ω_i, $\pi_{?(\tau)}(\omega_i) = \tau$. Pour tout classe C

$$\pi_{?(\tau)}(C) = \tau|C|$$

si bien qu'elle n'est prédite par un autre prédicteur δ que si $\pi_\delta(C) > \tau|C|$. Si ce n'est pas le cas, son prédicteur associé est $?(\tau)$. C'est ce que l'on désire.

Le choix de τ dépend de la « pureté » de prédiction que l'on désire obtenir dans les classes. Ainsi, par exemple, si la prédiction moyenne de δ sur la séquence est $\overline{\pi_\delta}$, la reconnaissance des segments où la différence entre la prédiction de δ et la prédiction supérieure de 20 % à cette moyenne est la plus grande est faite grâce au lexique :

`L =` δ `?`$(1,2.\overline{\pi_\delta})$

2.4.6.4 Minima de densités de prédictions

Si, pour un prédicteur donné δ, on désire construire des classes dont la densité de prédiction (c'est-à-dire la prédiction divisée par la longueur de la classe) est supérieure à un seuil τ_δ, il suffit (voir le paragraphe précédent) de considérer le lexique

`L =` $\delta(\tau_\delta^{-1})$ `?`

Ainsi, si on veut réaliser une partition prédictive d'une séquence avec les prédicteurs admissibles $\delta_1, \ldots, \delta_p$ de telle manière que la densité de prédiction

[5] qui, rappelons le, est admis comme prédicteur par toute lettre et qui, *a fortiori*, ne prédit rien.

de chaque classe dont le prédicteur optimal est δ_i soit supérieure à τ_{δ_i}, il suffit de travailler avec le lexique

$$\mathtt{L = } \delta_1(\tau_{\delta_1}^{-1}) \quad \delta_2(\tau_{\delta_2}^{-1}) \quad \ldots \quad \delta_{\mathtt{p}}(\tau_{\delta_{\mathtt{p}}}^{-1}) \quad \mathtt{?}$$

Dans ce cas, il faut tenir compte des pondérations apportées à ces prédicteurs pour estimer cette classification. Ce qui détermine qu'un prédicteur est déclaré optimal pour une classe est le facteur avec lequel sa densité d'occurences dans cette classe est éloigné de la pondération qui lui est assignée.

Les algorithmes et la syntaxe de lexique présentés dans ce chapitre nous permettent de concevoir un programme réalisant le partitionnement maximalement prédictif.

Chapitre 3

Le programme pampres

Pour pouvoir réaliser des partitionnements maximalement prédictifs sur des séquences d'objets, nous avons vu dans le chapitre deuxième un algorithme linéaire avec la taille de la séquence et applicable à une vaste gamme de prédicteurs de classes. En outre, pour avoir accès à toutes les k-partitions optimales d'un partitionnement sans avoir à les créer toutes, nous avons défini une relation d'équivalence entre ces k-partitions au chapitre deux, et nous avons conçu un algorithme qui ne construit que ces classes d'équivalence lors du partitionnement. Enfin, nous avons organisé une syntaxe de lexique qui permet d'utiliser une grande variété d'ensembles de prédicteurs ; grâce à cette syntaxe, il est aisé de décrire un quelconque ensemble de prédicteurs avec lequel on voudrait réaliser le partitionnement maximalement prédictif.

Le problème que l'on traite amène à manipuler simultanément lexiques, séquences, partitions et partitionnements. Ainsi, il a semblé adéquat d'utiliser un mode de programmation par objets pour permettre une gestion facile et efficace de tous ces objets. De plus, ce mode de programmation permet qu'un développement ultérieur du programme soit aisément réalisable. Le langage C++ [Str97] a été utilisé pour représenter ces « espèces » de données et traduire les algorithmes de partitionnement. Le programme qui en a résulté s'appelle `pampres` [1].

`pampres` permet d'opérer des partitionnements maximalement prédictifs sur des séquences de lettres à partir de lexiques de prédiction. Ces partitionnements peuvent être générés de manière à ne contenir pour chaque niveau k qu'une seule partition, ou bien toutes les k-partitions optimales, ou bien encore les classes d'équivalence de ces k-partitions optimales. En outre, par une interface en mode texte et une organisation interne de tous les objets que l'utilisateur voudra utiliser (séquences, partitions, partitionnement,...), ce programme permet de manipuler et d'observer les résultats de ces partitionnements. Cette interface de `pampres` consiste en une série de menus présentés en annexe A.

Dans ce chapitre, nous présentons ce programme selon les deux points de vue de sa réalisation, de sa complexité.

Le terme « classes » est ici utilisé à la fois dans un contexte de classification, comme cela a été le cas dans les deux premiers chapitres, et à la fois dans un contexte de *programmation orientée objet*. Dans ce dernier contexte, une

[1] PArtitionnement Maximalement PRÉdictif de Séquences

classe est un modèle d'organisation d'instances et de méthodes sur ces instances. Le sens adéquat du terme « classe » dépend du contexte de la phrase dans laquelle il sera employé ; il n'y a jamais d'ambiguïté.

Une classe générique[2] correspond, dans un contexte de programmation orientée objet, à un modèle d'organisation d'instances génériques et de méthodes sur ces instances génériques. Une classe générique, soit `Classe`, définie sur une classe quelconque `R` est notée `Classe<R>`.

3.1 Conception

La conception du programme est axée autour de deux priorités : la facilité d'utilisation des différents objets – séquences, partitions, partitionnements – par l'utilisateur et entre ces objets eux mêmes, et la possibilité d'opérer efficacement des partitionnements sur de très grandes séquences, c'est-à-dire de manière rapide et sûre. Ainsi, le programme est composé de deux parties liées. La première concerne la gestion des objets qui doivent pouvoir être accessibles directement par l'utilisateur. La seconde concerne l'implémentation même des algorithmes de partitionnement maximalement prédictif.

L'analyse du problème fait ressortir des listes ayant les mêmes propriétés, mais portant sur des éléments différents. C'est pour ne pas avoir à définir, dans chaque cas, les opérations nécessaires sur ces listes, qu'une classe générique `Tliste<T>` a été conçue, dans laquelle un grand nombre de méthodes de manipulation de listes quelconques a été défini. En outre, pour pouvoir facilement « naviguer » dans ces listes, une classe itérateur générique (voir [STR97]) `Tliste_iter<T>` a été élaborée. En outre, en C++, la gestion de pointeurs est naturelle et l'adresse de la tête d'une liste suffit à la référencer, ce qui est extrêmement économique en temps.

3.1.1 Gestion des classes apparentes

3.1.1.1 Présentation et organisation des classes apparentes

Différentes classes correspondent aux objets que l'on veut pouvoir directement manipuler. Ces classes, appelées **classes apparentes**, sont énumérées dans le tableau suivant et décrites ci-après.

La classe `Clexique` représente les lexiques. La composition de cette classe est décrite en section 3.1.1.2.

La classe `Cseq_prim` représente une séquence d'objets (Ω). Elle comprenant un tableau d'instances d'une classe `Cobjet`. `Cobjet` est la classe qui correspond aux objets (ω_i) de la séquence Ω. L'utilisateur n'est pas appelé à travailler directement avec la classe `Cobjet`.

La classe `Cgenome` représente une séquence génétique. Comme un domaine privilégié d'utilisation du partitionnement maximalement prédictif concerne les séquences génétiques (voir le chapitre 5), on peut vouloir travailler sur des séquences de lettres, dans lesquelles il existe des zones remarquables – tels les gènes

[2] ou classe patron, ou, en anglais, template (cf [STR97, chap. 13]).

objet principal	classe apparente	utilisation décrite en section
Lexique	`Clexique`	A.1
Séquence de `A`, `C`, `G`, `T`	`Cgenome`	A.2
Séquence	`Cseq_prim`	A.3
Partition simple	`Csequence`	A.4
Partitionnement simple	`Cpartitionnement`	A.5
Ensemble de partitionnements	`Cpopulation`	A.6
Partition complexe	`Cseq_equ`	A.7
Partitionnement complexe	`Cparti_comp`	A.8

dans des séquences génétiques. Ainsi, la classe `Cgenome` comprend un tableau de caractères (type `char`), et une liste d'instances de la classe `gene` qui, elle, a pour données membres deux entiers (type `int`) et une chaîne d'au plus trente caractères (type `char[30]`) pour les extrémités et le nom du gène. Le menu de construction d'une instance de `Cseq_prim` propose que cette construction se fasse sur la donnée d'une instance de `Cgenome`. Ceci permet de pouvoir pratiquer le partitionnement sur une séquence d'objets correspondant à la séquence génétique (voir la section A.3).

La classe `Csequence` correspond aux partitions prédictives. Une instance de cette classe est appelée **partition simple**. Son organisation est décrite en section 3.1.1.4.

La classe `Cpartitionnement` correspond à un ensemble de partitions prédictives. Une instance de cette classe est appelée **partitionnement simple**, et est un ensemble de partitions simples. La classe `Cpartitionnement` est composée d'une liste d'instances de la classe `Csequence`, cette liste relevant des différents niveaux qui constituent un partitionnement.

La classe `Cpopulation` correspond aux instances nommées **ensembles de partitionnements**. Une telle instance correspond à un ensemble de partitionnements simples créés à partir d'une population de séquences aléatoires. Une instance de la classe `Cpopulation` contient une liste d'instances de la classe `Cpartitionnement`.

La classe `Cseq_equ` correspond à un ensemble de partitions optimales ex æquo. Une instance de cette classe est appelée **partition complexe**. L'organisation de cette classe est décrite dans la section 3.1.1.4.

La classe `Cparti_comp` correspond aux ensemble de partitions complexes. Une instance de cette classe, instance appelée **partitionnement complexe**, contient une liste d'instances de `Cseq_equ`, pour les niveaux successifs du partitionnement.

Toutes les instances d'une même classe apparente sont contenues dans une liste d'instances de cette classe, et les huit listes résultantes sont des données membres d'une classe `Ctravail`. L'accès à une classe apparente par une autre se fait par le biais de procédures de recherche génériques définies dans `Ctravail`. La généricité de ces procédures exploite le fait que ces classes héritent toutes d'une classe `Csujet`. Cette classe a pour données membres une chaîne d'au plus

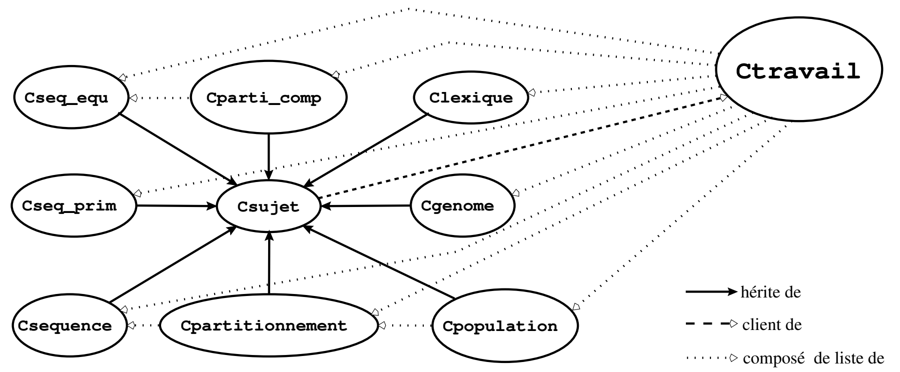


FIG. 3.1 – Relations entre les classes apparentes. Les classes sont figurées par les ellipses, et les flèches entre celles-ci illustrent les relations entre classes. La flèche pleine légendée `hérite de` symbolise une relation d'héritage entre classes. La flèche en tirets légendée `client de` signifie que la classe a pour donnée membre une référence vers une instance de la classe ciblée. La flèche pointillée légendée `composé de liste de` signifie que la classe a pour donnée membre une liste d'instances de la classe ciblée.

trente caractères qui permet à l'utilisateur d'identifier chaque objet à un nom, ainsi qu'un pointeur vers l'instance de `Ctravail` dans laquelle l'instance de la classe apparente considérée est stockée. Par ce pointeur, grâce à son héritage de `Csujet` une instance d'une classe apparente « sait » de quelle instance de `Ctravail` elle dépend, et peut avoir accès à toutes les listes et instances qui dépendent de cette instance de `Ctravail`.

3.1.1.2 Gestion du lexique

Dans le contexte de la classification, le lexique est l'ensemble de tous les prédicteurs de classes admissibles pour le partitionnement. Sa syntaxe a été présentée dans la partie 2.4 et son utilisation l'est dans la section A.1.

Lorsqu'il n'est pas structuré, un prédicteur de classe peut être assimilé à un prédicteur d'objet. Ce n'est pas le cas lorsqu'il est structuré. Nous devons donc différencier les deux notions dans leur représentation en classe C++.

Un prédicteur d'objet correspond à un descripteur et à un facteur pour le poids de ce prédicteur. Le descripteur est stocké dans une classe `Cdescripteur`, qui ne contient qu'un `char`, et le facteur dans une classe `Cvaleur`, qui ne contient qu'une valeur flottante (type `double`). Un prédicteur d'objet doit aussi pouvoir gérer les voisins de cet objet dans la séquence, dans le cas où ce prédicteur est contextuel (par exemple si le mot `"ATC"` est donné dans le lexique, voir la partie 2.4.4). En outre, il peut contenir des ambiguïtés de descripteurs telles que `[ATC]` (voir la partie 2.4.3).

Pour cela, un prédicteur d'objet est une instance de la classe `Cdesc_comp`, héritée de `Cvaleur` et de `Cdescripteur`, et possédant comme données membres deux pointeurs qui, soit pointent sur `NULL`, soit référencent des instances de `Cdesc_comp`. Le premier de ces pointeurs correspond au successeur éventuel du prédicteur dans le contexte, et le second correspond à un descripteur éventuel avec lequel ce prédicteur partage l'ambiguïté. Ces instances pointées peuvent aussi partager une ambiguïté ou bien faire partie d'un contexte plus important.

Dans le cas où il est structuré, un prédicteur de classe est un motif de prédicteurs d'objets. De plus, l'algorithme linéaire issu des calculs de la partie 2.4.5, page 50 veut que la prédiction d'une classe $\omega_{\langle 1,k \rangle}$ par le prédicteur, disons `ABC`, dépende directement de celle de $\omega_{\langle 1,k-1 \rangle}$ par le prédicteur `CAB`. Ainsi, pour l'efficacité du programme, nous avons choisi de gérer toutes les phases d'un prédicteur structuré au sein d'une répétition d'instances de la même classe. Une organisation en liste sans queue ni tête – une boucle – de `Cdesc_comp` est adaptée à ce type de traitement. La relation de succession dans une telle boucle est la représentation de la relation de succession au sein d'un prédicteur structuré.

Un nœud de cette boucle est une instance de la classe `Cmotif`, héritée de `Cdesc_comp`. Il a paru plus adapté de choisir la relation d'héritage entre la classe `Cmotif` et la classe `Cdesc_comp`, parce que la méthode qui calcule la prédiction sur un objet fait partie de la classe `Cdesc_comp` et donc, par héritage, elle est aussi une méthode de `Cmotif`. `Cmotif` a pour donnée membre un pointeur vers une instance de `Cmotif` qui représente son successeur dans le prédicteur structuré. Dans le cas où le motif est un seul descripteur, le pointeur pointe sur l'instance dont il est membre. Par ce moyen, chaque instance de `Cmotif` qui correspond à une phase du prédicteur structuré est directement accessible par l'instance de `Cmotif` qui correspond à sa phase suivante. Ceci permet que le passage d'une phase à une autre ne se fasse que par la transmission de références, ce qui est extrêmement rapide. Bien sûr, l'idée sous-jacente est liée au calcul décrit dans la partie 2.4.5, où les phases successives des variables dépendantes des motifs sont liées. Si le même procédé que celui qui lie les instances de `Cmotif` pouvait être utilisé dans la procédure qui correspond à l'algorithme, il ne serait pas plus coûteux en temps de gérer un prédicteur structuré de longueur n que n prédicteurs d'objets simples.

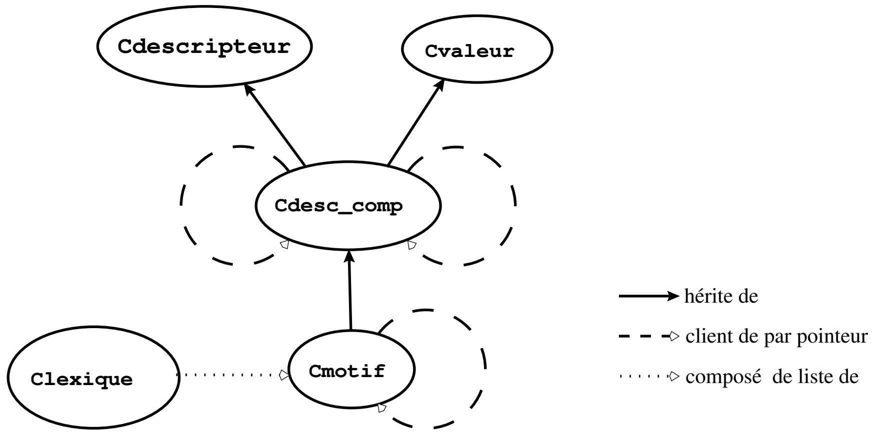


FIG. 3.2 – Organisation des classes correspondant aux prédicteurs et au lexique. La classe `Clexique` contient une liste d'instances de `Cdesc_comp`.

La classe `Clexique` contient la liste de tous les `Cmotif` concernés. Lors de sa construction, deux procédures permettent sa simplification par rapport aux données introduites. L'une d'elles supprime la redondance au sein de motifs (transformant le mot `ATAT` en `AT` par exemple). L'autre supprime les ambiguïtés de prédiction, transformant par exemple le lexique `A(2) A` en `A(2)` simplement.

3.1.1.3 Objets et classes prédictives

Un objet d'une séquence correspond à un descripteur et, par souci de généralité, un objet peut avoir un certain poids. Ainsi, la classe correspondante `Cobjet` hérite de `Cdescripteur` et de `Cvaleur`. Une séquence, représentée par la classe `Cseq_prim`, est un ensemble de tels objets organisé en tableau.

Une classe prédictive – au sens de la classification – est une instance de la classe – au sens C++ – `Csegment`. `Csegment` est cliente d'une liste d'instances de `Cdesc_comp`. Cette liste correspond au prédicteur (éventuellement structuré) de la classe prédictive. Un pointeur sur un élément de cette liste établit la phase du prédicteur structuré de cette classe. De plus, `Csegment` est à deux reprises client d'une classe `Cposition` contenant un `int` ; ces deux relations de clientèle correspondent aux positions des césures de la classe dans la séquence. Une k-partition prédictive (classe C++ `Csequence`) est représentée par un tableau de taille k d'instances de `Csegment`.

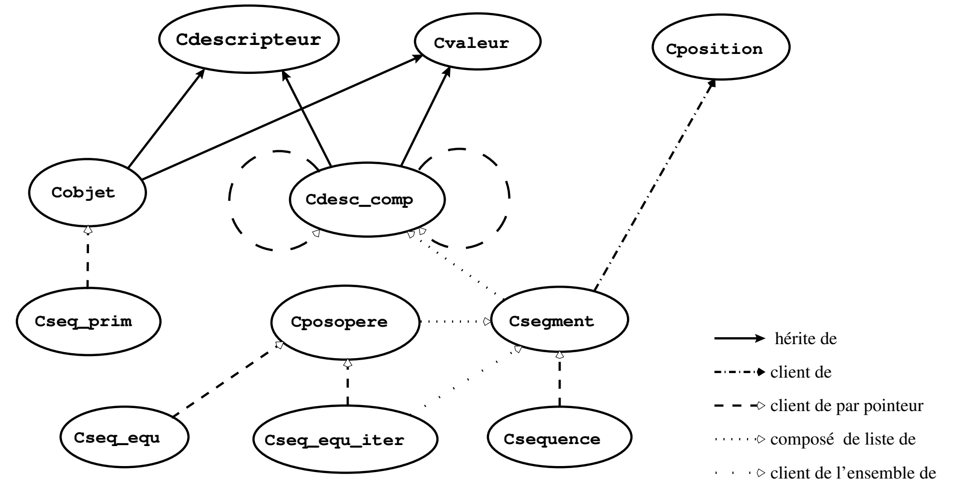


FIG. 3.3 – Organisation des classes C++ associées aux objets, classes prédictives, partitions prédictives et partitionnements. Les flèches pleines représentent les relations d'héritage. La flèche à tête noire en tirets et pointillés correspond à une relation de clientèle. Les flèches à têtes blanches en tirets représentent des relations de clientèle par pointeur, et celles en pointillés des relations de clientèle par liste. La flèche en pointillés très espacés entre `Cseq_equ_iter` et `Csegment` correspond à une relation de clientèle à l'égard d'un ensemble de `Csegment`.

3.1.1.4 Partitions simples et complexes

Représentation

La gestion des partitions simples et des partitions complexes est faite dans deux classes différentes, respectivement `Csequence` et `Cseq_equ`. Cette distinction a été choisie car l'algorithme de partitionnement maximalement prédictif est plus rapide lorsqu'il n'y a qu'une seule représentante des k-partitions optimales à créer. Ainsi, nous avons voulu séparer les classes C++ qui correspondent aux deux préférences : l'efficacité du programme ou bien l'intégralité des partitions optimales. De plus, les méthodes (telles que les calculs, les graphiques, etc) diffèrent que l'on travaille avec une seule partition ou bien avec un ensemble de partitions.

Bien sûr, étant donné le nombre potentiel des partitions optimales ex æquo, il est hors de question de gérer l'ensemble de ces dernières comme un tableau de partitions, et il est bien plus économique de ne gérer que les classes de ces partitions. Par conséquent, nous convenons qu'une instance de `Cseq_equ` qui correspond à l'ensemble des k-partitions prédictives optimales contient un tableau de k instances d'une classe nommée `Cposopere`. La classe `Cposopere` contient une liste d'instances de `Csegment`. L'instance de `Cposopere` qui correspond au premier élément du tableau de taille k de `Cseq_equ` contient les segments des classes les plus à gauche des k-partitions optimales ; le deuxième élément de ce tableau correspond aux classes qui sont les voisines à droite des premières dans les k-partitions optimales, et ainsi de suite jusqu'à remplir le k^{e} élément du tableau, qui correspond aux classes prédictives les plus à droite des k-partitions optimales.

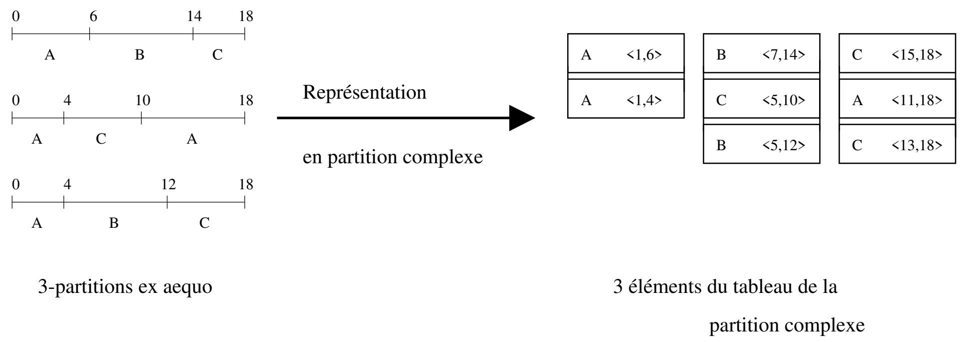


FIG. 3.4 – Représentation de trois 3-partitions ex æquo en une partition complexe. Cette partition complexe comprend un tableau de trois éléments. Chaque élément du tableau comprend des instances de `Csegment` qui représentent les classes des partitions.

Les segments ainsi stockés correspondent exactement aux arêtes du graphe des k-partitions (voir la figure 1.5, page 30). Cette représentation a l'avantage de la compacité mais présente à première vue l'inconvénient qu'il faut reconstruire les partitions prédictives ex æquo lorsque nécessaire.

Reconstruction

À première vue, cette représentation simple en tableau des classes de toutes les k-partitions prédictives optimales n'est pas une représentation fonctionnelle des partitions. Chacune de ces partitions peut être reconstruite à partir de ce tableau : il suffit de choisir, dans chaque case du tableau, une « bonne » instance de `Csegment` de manière à ce que les segments ainsi choisis constituent une partition optimale. Mais s'il fallait vraiment *choisir* ces bonnes instances, reconstruire une partition optimale serait extrêmement coûteux. En fait, comme les k-partitions sur la donnée desquelles ce tableau a été construit sont optimales, on est garanti que quelle que soit la k-partition reconstruite à partir du tableau, elle est optimale. La raison de ceci est liée au lemme de FISHER [FIS58], voir page 36.

En effet, supposons qu'il existe k classes reconstruites à partir du tableau des `Cposopere`, disons successivement $C_1, C_2, \ldots, C_k$, qui forment une k-partition de Ω qui ne soit pas optimale. Notons $(x_i)_{i \in \langle 1,k-1\rangle}$ les positions des césures entre ces classes (voir les notations au début du chapitre deux). Soit $j \in \langle 2, k\rangle$ le plus petit entier tel que $(C_1, \ldots, C_j)$ ne soit pas une j-partition optimale de $\omega_{\langle 1,x_j\rangle}$. Comme la classe C_j a été construite à partir d'une k-partition optimale, soit $(C'_1, \ldots, C'_{i-j}, C_j, C'_{j+1}, \ldots, C'_k)$ une telle k-partition optimale. Par le lemme de FISHER, $(C'_1, \ldots, C'_{j-1}, C_j)$ est une j-partition optimale de $\omega_{\langle 1,x_j\rangle}$, si bien que :

$$\pi(C'_1, \ldots, C'_{j-1}, C_j) > \pi(C_1, \ldots, C_{j-1}, C_j)$$

et donc :

$$\pi(C'_1, \ldots, C'_{j-1}) > \pi(C_1, \ldots, C_{j-1})$$

si bien que $(C_1, \ldots, C_{j-1})$ n'est pas une $(j-1)$-partition optimale de $\omega_{\langle 1,x_{j-1}\rangle}$. Ceci contredit l'hypothèse sur j. Par conséquent, toute k-partition reconstruite à partir du tableau des `Cposopere` est une k-partition optimale, du moment que ce tableau a été correctement construit à partir des classes de k-partitions optimales.

Si l'ensemble des k-partitions ex æquo qui a été représenté sous forme de partition complexe n'est pas l'ensemble de toutes les k-partitions ex optimales mais seulement une k-partition de chaque classe d'équivalence, le mécanisme de représentation de l'ensemble de ces k-partitions est semblable, et garantit la reconstruction d'une seule k-partition par classe d'équivalence.

Une classe C++ itérateur (voir [STR97]), `Cseq_equ_iter`, permet de naviguer dans l'ensemble des k-partitions optimales. Elle est conçue comme un tableau d'itérateurs instances de la classe `Tliste_iter<Csegment>`, et comme un tableau de références sur des instances de `Csegment`. Les opérateurs de la classe `Cseq_equ_iter` sont conçus de telle sorte que les instances de `Csegment` référencées dans deux cases successives du tableau de pointeurs correspondent toujours à des classes prédictives contiguës. Un intérêt d'une telle classe est de permettre de dessiner facilement le graphe des k-partitions optimales qui est une représentation très pratique de l'ensemble des partitions ex æquo (voir la figure 1.5 et la section A.7). Cette classe permet aussi, par un procédé de programmation dynamique, de sélectionner efficacement une k-partition par l'optimisation de critères additifs tels que la densité, l'interaction, etc. Cette classe sert à la reconstruction de l'ensemble des k-partitions optimales à partir de l'ensemble des représentants des classes d'équivalence.

3.1.2 Au cœur de l'algorithme

Nous avons expliqué page 41 qu'une classe prédictive est, au cœur même du programme qui correspond à l'algorithme, référencée par sa seule césure gauche. Cette représentation est duale de la représentation en instance de `Csegment` d'une classe prédictive. Ce choix a été fait pour alléger la gestion de ces classes, et ainsi accélérer l'exécution du programme, car de tels objets sont très nombreux et peuvent être très éphémères pendant l'exécution du programme. Une k-partition est considérée comme une liste de césures.

Par souci de rapidité d'exécution, nous avons programmé deux variantes. La première variante permet de construire toutes les k-partitions prédictives optimales,la seconde variante ne permet de construire qu'une seule k-partition prédictive optimale. La première variante est la plus coûteuse en temps et en espace, même si l'on se restreint aux classes d'équivalence (voir 2.3.1).

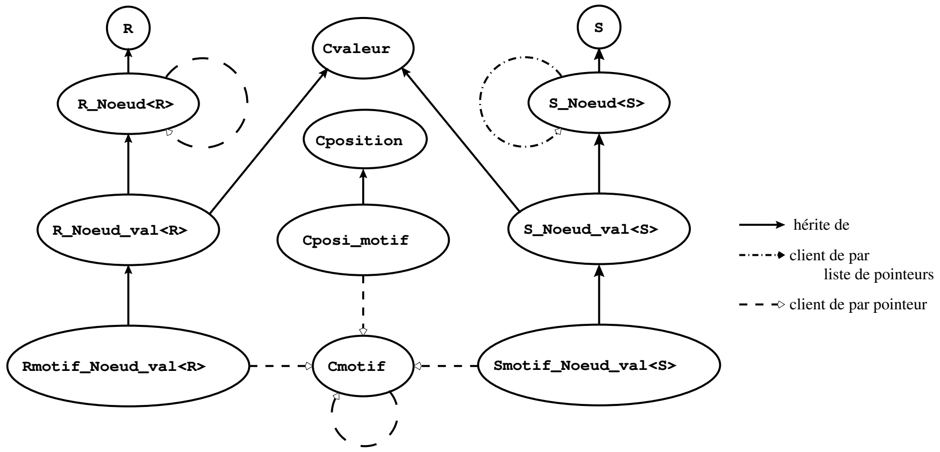


FIG. 3.5 – Organisation des classes au cœur de l'algorithme.

3.1.2.1 Le cas simple

Dans le cas le plus simple, il faut gérer autant de listes de césures qu'il y a de prédicteurs dans le lexique (ce qui correspond aux $\mathbb{M}_k^\delta$, voir page 39) et de positions dans la séquence (ce qui correspond aux partitions des sous-séquences $\mathbb{M}_{k-1}(\omega_{\langle 1,i\rangle})$, voir page 39). Pour rendre indépendant le programme exécutant l'algorithme de partitionnement de la spécification des instances sur lesquelles il s'effectue, toutes les classes qui permettent la gestion des césures sont des classes génériques. L'efficacité de l'exécution n'est en rien pénalisée par ce choix, car lors de la compilation, les méthodes génériques sont incluses directement avec leur spécification dans le code là où elles sont utilisées[3]. Si une césure est représentée par une instance d'une classe R quelconque, comme une partition est une liste de césures, nous gérons une liste d'instances de la classe R. Un chaînon de cette liste est une instance de la classe R_Noeud<R>, héritée de R et qui a pour donnée membre un pointeur vers l'instance de R_Noeud<R> qui le devance dans la liste.

Par économie, dans l'algorithme de la page 41, la construction des listes L[i] est faite par la transmission des adresses des têtes des listes L'[i-1] et $\mathtt{L}^{\delta}$. Par conséquent, une instance de R_Noeud<R> peut être pointée par plusieurs autres instances de R_Noeud<R>. Comme, lorsqu'une liste de césures devient obsolète, il faut la détruire, chaque variable R_Noeud<R> de cette liste, avant d'être détruite, doit savoir si elle est encore pointée. Pour cela, un entier (type longint) permet de compter combien de fois cette instance est pointée[4].

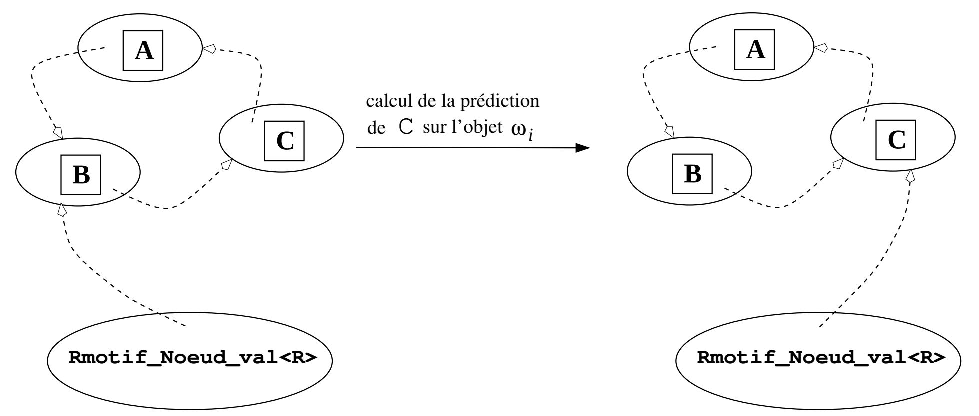


FIG. 3.6 – Le motif concerné est ABC. Une instance de Rmotif_Noeud_val<R> pointe sur l'instance de Cdesc_comp qui possède le caractère B. Après la comparaison d'un objet ω_i avec le prédicteur C, cette instance de Rmotif_Noeud_val<R> voit son pointeur changé dans le sens des relations qui lient les instances Cdesc_comp du Cmotif.

Comme les valeurs des objets correspondant aux $\mathbb{M}_{k-1}(\omega_{\langle 1,i\rangle})$ sont utilisées dans l'algorithme, il faut gérer les têtes des listes de césures auxquelles sont liées ces valeurs, ce qui est fait par la classe R_Noeud_val<R>, héritée de R_Noeud<R> et de Cvaleur. Enfin, pour tous les prédicteurs $\delta \in D$, chaque variable correspondant à $\mathtt{L}^{\delta}$ est représentée par une tête de liste qui doit en plus contenir une référence sur un prédicteur de classe prédictive. Pour cela, on crée la classe C++ Rmotif_Noeud_val<R>, qui est héritée de R_Noeud_val<R> et possède un pointeur vers une instance de Cmotif stockée dans la liste donnée membre de Clexique. Au cours du partitionnement, dans le calcul de chaque instance Rmotif_Noeud_val<R>, ce pointeur est modifié pour pointer sur l'ins-

[3] Elles sont forcément définies en « inline » dans un fichier dont dépend le fichier de déclaration de la classe générique.

[4] Ce comptage permet de libérer une instance qui n'est plus utilisée dans l'arborescence des relations entre césures successives. Lorsqu'une instance de R_Noeud<R> est détruite, l'instance sur laquelle pointait celle-ci voit son compteur être décrémenté. Si celui-ci devient nul, cette instance est aussi détruite, et ainsi de suite.

tance de Cmotif référencée dans l'instance de Cmotif sur laquelle il pointe alors. Ainsi, si on considère l'ensemble des instances des Rmotif_Noeud_val<R> qui correspondent aux variables $\mathbb{M}_k^\delta$ pour les descripteurs δ d'un lexique, c'est le même principe qui organise cet ensemble et l'instance de Clexique corrspondante (voir la figure 3.6).

Dans la partie du programme qui code l'algorithme, la généricité de la classe R est utilisée pour permettre de spécifier cette classe par la classe C++ qui correspond aux positions des césures, puisque ce sont les césures qui constituent les maillons des listes associées aux partitions. La classe associée aux positions des césures est Cposition, déjà utilisée dans la définition de la classe Csegment. Une instance de R_Noeud<Cposition> est construite au début du processus. Ses objectifs sont de définir la césure de la position de début de séquence, et de terminer toutes les listes d'instances de R_Noeud<Cposition>.

3.1.2.2 Le cas complexe

Le principe de construction est semblable dans le cas où on veut construire toutes les partitions optimales ex æquo. Un ensemble de césures doit être alors géré à partir de chaque césure puisque, dans l'algorithme de la page 41, toutes les partitions qui atteignent le maximum doivent être enregistrées. Si S est une classe générique, la classe S_Noeud<S> hérite d'elle et est cliente d'un ensemble de pointeurs vers d'autres S_Noeud<S>, gérés sous forme de liste. De plus, comme ci-dessus, cette classe contient un compteur d'utilisation[5].

Les classes S_Noeud_val<S> et Smotif_Noeud_val<S> sont créées et gérées de la même manière que R_Noeud_val<R> et Rmotif_Noeud_val<R>.

Dans la partie du programme qui correspond à l'algorithme, comme ci-dessus, la classe Cposition est utilisée comme spécification de la classe S. Là aussi, une instance particulière de S_Noeud<Cposition> sert de queue à toutes les listes. Si le choix a été fait de construire les classes d'équivalence des partitions optimales, les césures doivent être différenciées les unes des autres aussi bien par leur position et que par le prédicteur de la classe dont elles sont la borne inférieure. Pour cela, il faut gérer en même temps directement les prédicteurs et les positions des classes pour différencier les césures. La classe Cposi_motif réalise cet objectif. Héritée de Cposition, elle contient un pointeur vers l'instance Cmotif du lexique. Dans ce cas, on construit autant d'instances de S_Noeud<Cposi_motif> que de prédicteurs admissibles qui sont construites pour servir de queues aux listes d'instances de S_Noeud<Cposi_motif>. La classe Cposi_motif s'avère également utile pour d'autres parties du programme où il faut gérer des partitions prédictives.

Lors de la transformation ultime des listes de césures en listes de Csegment au sein de Csequence ou de Cseq_equ, si l'information relative aux césures de classes prédictives ne comprend que leur position (par utilisation de la classe Cposition), les prédicteurs optimaux de chaque classe sont reconstruits à partir

[5]Lorsqu'une instance de S_Noeud<S> est détruite, chaque instance de S_Noeud<S> dont l'adresse est dans la liste de celle-ci voit son compteur être décrémenté, et est détruite si celui-ci devient nul, ce qui entraîne la décrémentation d'autres compteurs d'instances de S_Noeud<S> et ainsi de suite.

du lexique. Pour cela, sur une classe de la séquence connue par ses deux césures, par une itération simple de tous les Rmotif_Noeud_val<Cposition> qui ont été construits sur les motifs du lexique, les prédicteurs optimaux de cette classe sont retrouvés. En revanche, si l'information relative aux césures comprend aussi les prédicteurs des classes prédictives auxquelles elles sont associées (par utilisation de la classe Cposi_motif), cette opération n'est pas nécessaire.

3.1.2.3 Lexique générique

La gestion des instances des classes Rmotif_Noeud_val<R> et des classes Smotif_Noeud_val<S> se fait par le biais d'une classe générique U_lex<U>. Lorsque U est Rmotif_Noeud_val<R>, cette classe est cliente d'une liste d'instances de la classe Rmotif_Noeud_val<R> et d'une liste d'instances de la classe R_Noeud<R>. Lorsque U est Smotif_Noeud_val<S>, cette classe est cliente d'une liste d'instances de la classe Smotif_Noeud_val<S> et une liste d'instances de la classe S_Noeud<S>. La construction d'une instance de U_lex<U> se fait par la donnée d'une instance de la classe Clexique.

3.1.2.4 Ramasse-miettes

Comme les césures des partitions intermédiaires sont susceptibles d'être beaucoup modifiées, les instances des classes qui correspondent aux césures, R_Noeud<R> et S_Noeud<S>, sont fréquemment créées et détruites. Pour accélérer l'exécution et éviter les procédures de réservation et libération de la pile à chaque fois, un système de gestion de mémoire « ramasse-miettes »[6] a été installé de manière que l'allocation et la libération de la mémoire se fassent par paquets de tels objets à chaque fois que nécessaire. La taille de ces paquets a été choisie à 10000, ce qui nous a semblé un bon compromis entre l'inutilité en espace et la difficulté d'allouer de trop gros paquets, et l'inutilité en temps d'allouer de trop petits paquets. Une classe générique Stock<W> gère une liste des adresses des instances de la classe W qui ont été allouées, mais pas encore utilisées. Lorsqu'il y a besoin d'une instance de W, une adresse est enlevée de la liste et envoyée à l'opérateur de construction de W. Lorsqu'une instance de W doit être détruite, son adresse est juste remise dans la liste des adresses. Lorsque cette liste est vide et qu'il faut créer une instance de W, un paquet d'allocations est fait.

Nous venons de présenter les deux aspects de la conception du programme pampres. Nous allons maintenant étudier l'efficacité du partitionnement maximalement prédictif fait par ce programme.

3.2 Performances

Comme un des objectifs du programme pampres est de permettre de réaliser le partitionnement de grandes séquences, il nous semble intéressant d'étudier son efficacité en temps et en mémoire. L'algorithme qui est au cœur du programme

[6] ou garbage-collector

est linéaire en temps ; il s'agit de vérifier si le temps de calcul du programme est aussi linéaire, et si le facteur dont dépend cette linéarité est raisonnable (de l'ordre de la seconde, de l'heure, de la semaine, ou de l'année...). En outre, dans le cas où le partitionnement cherche toutes les partitions ex æquo optimales à tous les niveaux, on peut se demander quelle est la répercussion de cette gestion sur l'efficacité, aussi bien avec l'algorithme qui construit effectivement toutes les partitions qu'avec celui qui ne construit que les classes d'équivalence. Enfin, nous pouvons aussi nous intéresser à l'accroissement prévisible de la complexité avec la taille du lexique de prédicteurs.

Les expérimentations ont été faites sur un Gateway à processeur Pentium III à 500 Mhz, avec 256 Mo de mémoire vive.

3.2.1 Partitionnements simples

La figure 3.7 présente les courbes du temps de création de k-partitions pour k entre 10 et 2000 sur des séquences de longueur entre 1000 et 100 000 lettres. Ces séquences sont sur un alphabet de quatre lettres, et le lexique est `L = *`. Les droites en tirets correspondent à la pente des fonctions linéaires. On peut mesurer que, quelle que soit la longueur de la séquence, le rapport des pentes expérimentales sur la pente de linéarité est $1,075$, si bien qu'en pratique, avec le nombre de classes k, le temps de calcul se comporte comme $k^{1,075}$. L'éloignement de la linéarité est certainement dû à la construction des instances de la classe `Csequence` à partir des instances concernées de la classe `Rmotif_Noeud_val<Cposition>`, qui nécessite d'autant plus de travail et d'allocations en mémoire que k est grand.

En ce qui concerne la complexité en fonction de la taille de la séquence, on observe, pour des partitions simples en 10 et 100 classes, sur la figure 3.8, que les courbes expérimentales sont parallèles à la droite qui correspond aux fonctions linéaires, si bien que le programme est effectivement linéaire avec la taille de la séquence.

La figure 3.9 représente la complexité temporelle du programme, dans le cas de la construction de partitions simples, en fonction de la taille de l'ensemble des descripteurs. Les séquences utilisées sont construites sur un alphabet à quatre lettres, et les lexiques utilisés sont de la forme par exemple `L = ***` pour créer un ensemble de descripteurs de taille $4^3 = 64$. On remarque qu'alors le programme est un peu sous linéaire[7] en fonction de la taille $|D|$ de l'ensemble des descripteurs admissibles – approximativement se comporte comme $|D|^{0,88}$.

Le temps de création d'une partition est, même sur de très grandes séquences et avec beaucoup de classes, très acceptable, puisqu'il faut, avec 4 prédicteurs admissibles, moins de 13 minutes (environ 770 secondes) pour partager en 100 classes une séquence de 2.10^6 lettres, et environ $4,3$ secondes pour partager en 100 classes une séquence de 10 000 lettres, ou alors en 10 classes une séquence de 100 000 lettres.

Pour ce qui est de l'occupation de l'espace mémoire, une séquence de longueur n est stockée dans une instance de la classe `Cseq_prim` sous la forme d'un

[7]On dit qu'une fonction est sous-linéaire si elle est en $o(x)$.

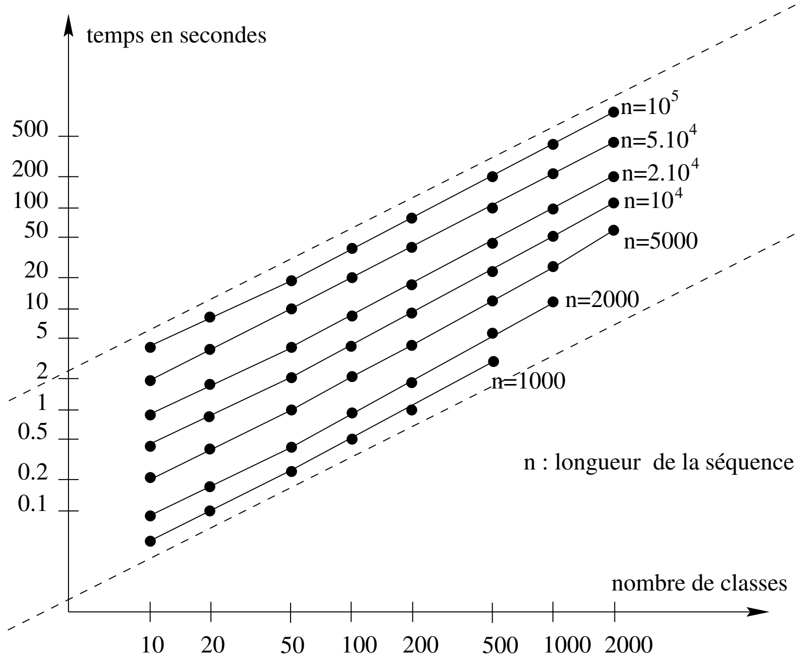


FIG. 3.7 – Courbes du temps moyen de calcul d'une k-partition simple pour k entre 10 et 2000 sur des séquences de longueur entre 1000 et 100 000 lettres. Ces séquences sont sur un alphabet de quatre lettres, et le lexique est `L = *`. Les droites en tirets correspondent aux fonctions linéaires.

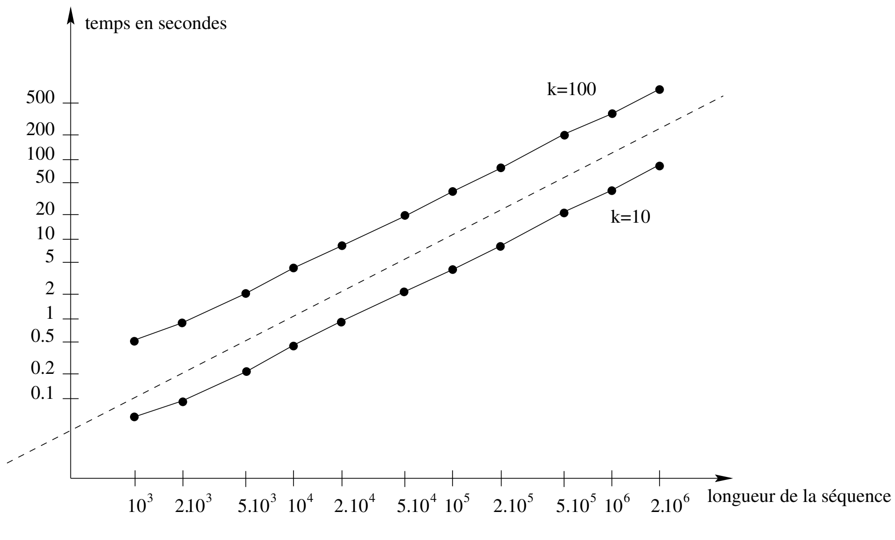


FIG. 3.8 – Courbes du temps moyen de calcul de k-partitions simples pour $k = 10$ et $k = 100$ sur des séquences de longueur entre 1000 et 2.10^6 lettres. Ces séquences sont sur un alphabet de quatre lettres, et le lexique est `L = *`. La droite en tirets correspond aux fonctions linéaires.

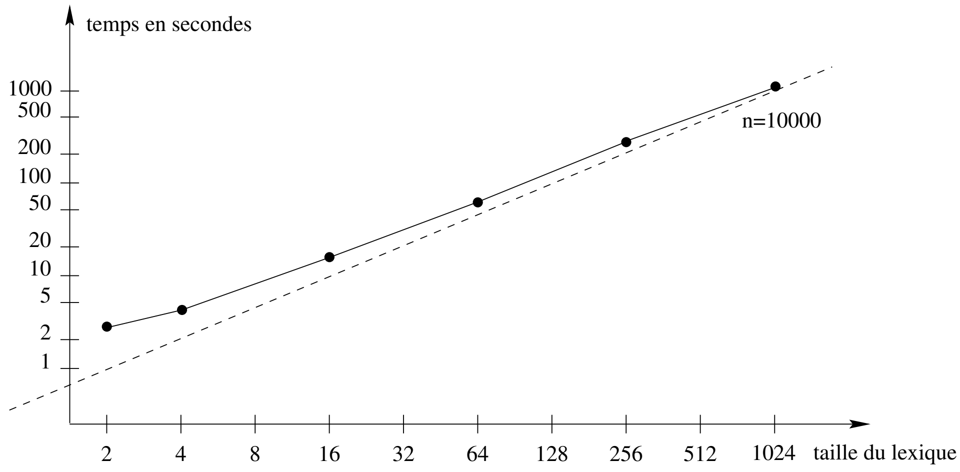


FIG. 3.9 – Courbes du temps moyen de calcul de 100-partitions simples sur des séquences de longueur n, en fonction de la taille du lexique des prédicteurs. La droite en tirets correspond à la pente des fonctions linéaires.

tableau de taille n d'instances de `Cobjet`. Comme une instance de `Cobjet` occupe 16 octets[8], une séquence occupe à peu près $16n$ octets. Lors du partitionnement, les instances de loin les plus nombreuses sont celles qui correspondent aux variables `L[i]` et `L'[i]`, c'est-à-dire celles de la classe `R_Noeud_val<Cposition>`. Elles sont au nombre de $2n$ et occupent chacune 24 octets[9]. Par conséquent, dès le début du processus, la mémoire utilisée est de l'ordre de $64n$ octets. Pour une séquence de 2.10^6 objets, cela fait 128 Mo.

En outre, les autres objets créés en grand nombre sont ceux qui correspondent aux césures, instances de la classe `R_Noeud<Cposition>` qui occupe 12 octets. Leur nombre est variable, selon la séquence partagée, le lexique utilisé et le nombre de classes. Il dépend de l'ordre de grandeur de la taille de la séquence, et est extrêmement lié, comme on pourrait s'y attendre, au nombre de classes. Par exemple, pour une séquence de plus de cent mille lettres sur un alphabet de quatre lettres et le lexique `L = *`, pour créer une k-partition, si k est de l'ordre de la dizaine, ce nombre est de l'ordre de quelques milliers. Bien sûr, plus le nombre de classes est grand, plus le nombre de césures est important. Pour des séquences similaires de 10^5 lettres, le nombre de césures nécessaires pour créer des 1000-partitions est de l'ordre de quelques millions. Ainsi, si le partitionnement en milliers de classes est effectué sur une séquence de plusieurs millions de lettres, la mémoire vive de l'ordinateur risque d'être saturée et la mémoire-cache sera utilisée, ce qui augmente considérablement le temps de calcul.

[8]par héritage de `Cdescripteur` qui prend un octet et par héritage de `Cvaleur` qui prend huit octets, et la gestion MS-DOS qui préfère les multiples de huit...

[9]par héritage de `R_Noeud<Cposition>` qui occupe douze octets (par héritage de `Cposition` qui prend quatre octets, plus une adresse (quatre octets) et un `int` (quatre octets)) et plus un double qui occupe huit octets, et la gestion MS-DOS qui préfère les multiples de huit...

3.2.2 Partitionnements complexes

La complexité des partitionnements qui gèrent toutes les partitions optimales est moins prévisible, car elle dépend de la gestion des listes de césures qui correspondent aux partition ex æquo.

En premier lieu, si on s'intéresse au partitionnement dans lequel toutes les partitions optimales sont directement gérées, on observe sur la figure 3.10 qu'en fonction du nombre de classes, les pentes sont beaucoup moins régulières que dans le partitionnement simple. En général, la complexité temporelle est tout de même très proche de la linéarité avec le nombre de classes. Ceci est très rassurant, car l'éventualité d'avoir à gérer tous les ex æquo pouvait faire craindre une chute brutale de l'efficacité du programme. Cette chute peut être observée dans le cas où la longueur de la séquence est 1000 et le nombre de classes 500, mais ceci s'explique par le fait que dans ce cas, la moyenne des tailles des classes est de deux lettres, et qu'il y a en fait énormément d'ex æquo. Cette remarque est en accord avec les observations du comportement du partitionnement qui seront faites dans le quatrième chapitre « Comportement ».

Ce phénomène se voit aussi dans la figure 3.11. En effet, la courbe du temps de partitionnement en fonction de la longueur de la séquence n'est proche de la linéarité que lorsque la taille moyenne des classes est assez grande.

On peut remarquer que dans ce cas la complexité du programme demeure légèrement moins que linéaire avec le nombre de prédicteurs (figure 3.12).

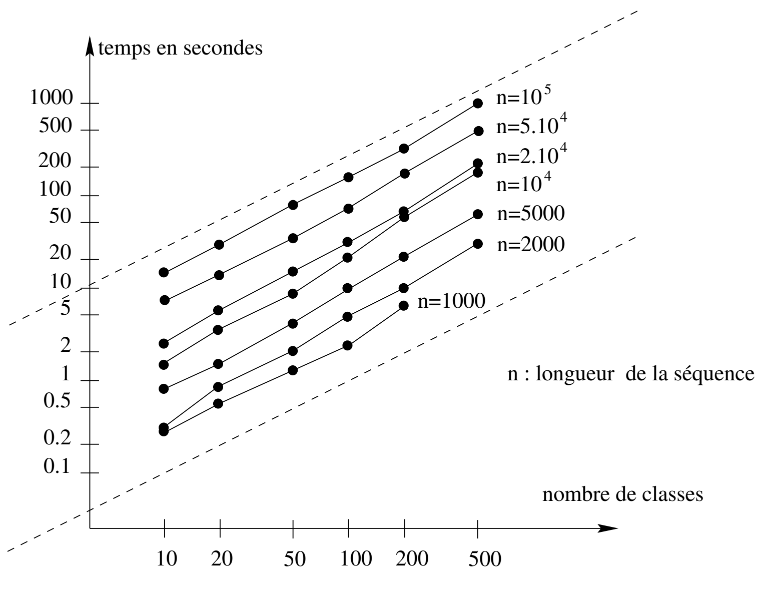


FIG. 3.10 – Courbes du temps moyen de création de k-partitions complexes pour k entre 10 et 500 sur des séquences de longueur entre 1000 et 100 000 lettres. Ces séquences sont sur un alphabet de quatre lettres, et le lexique est `L = *`. Ces partitions complexes ont été construites sans passer par les classes d'équivalence. Les droites en tirets correspondent aux fonctions linéaires.

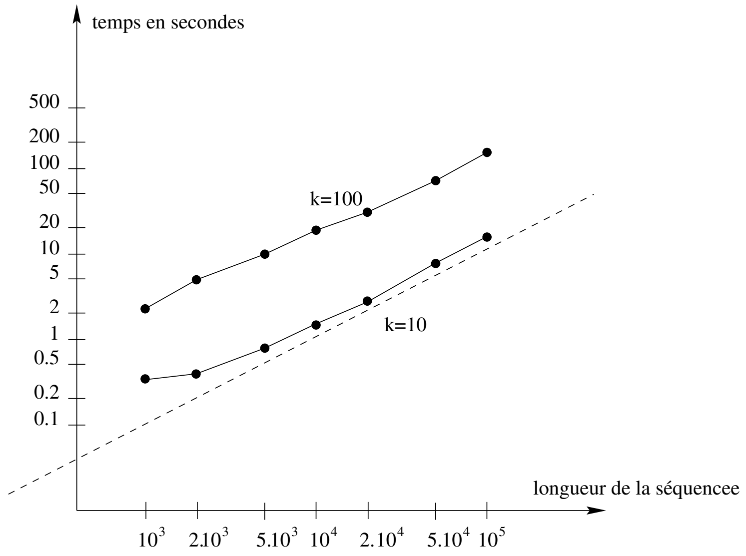


FIG. 3.11 – Courbes du temps moyen de création de k-partitions complexes pour $k = 10$ et $k = 100$ en fonction de la longueur des séquences (entre 1000 et 100 000 lettres). Ces séquences sont sur un alphabet de quatre lettres, et le lexique est `L = *`. La droite en tirets correspond aux fonctions linéaires.

En moyenne, créer une k-partition complexe ne prend que quatre fois plus de temps que créer une k-partition simple, ce qui est très raisonnable. Cette efficacité est sûrement due à la gestion en arborescence de toutes les listes de césures. Cependant, la limitation de la construction des k-partitions optimales concerne la gestion de la mémoire.

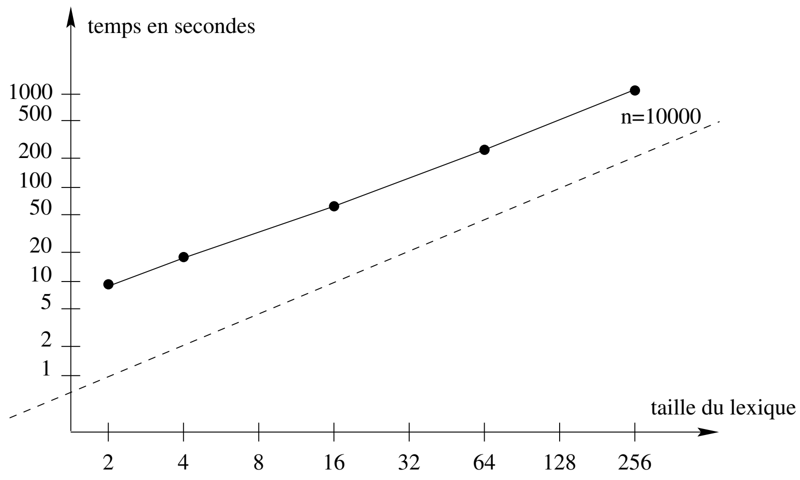


FIG. 3.12 – Courbes du temps moyen de création de 100-partitions simples sur des séquences de longueur $n = 10000$, en fonction de la taille du lexique des prédicteurs. La droite en tirets correspond à la pente des fonctions linéaires.

Comme une instance de la classe `S_Noeud_val<Cposition>` occupe 24 octets, le calcul effectué dans l'étude du partitionnement simple est applicable ici : dès le début du processus, la mémoire occupée pour partager une séquence de taille n est de l'ordre de $64n$ octets. Mais à ceci s'ajoute le fait que pendant le partitionnement, les listes d'adresses de ces instances – adresses des instances de `S_Noeud_val<Cposition>` qui les suivent dans les listes de césures – ne sont pas vides, et que l'espace mémoire occupé peut alors grandir. Ceci est aussi le cas pour les instances de `S_Noeud<Cposition>` qui correspondent aux césures des partitions. Même si la taille d'une instance de `S_Noeud<Cposition>` n'est « à vide » que 12 octets, si ces instances sont nombreuses, le programme peut s'avérer très gourmand en espace mémoire. Le nombre de césures créées est bien sûr extrêmement variable selon les séquences et les lexiques et il dépend principalement du nombre de classes. En général, sur de très grandes séquences construites sur un alphabet de quatre lettres, avec le lexique `L = *`, pour un nombre important de classes (100 à 500), il y a besoin d'allouer 10 à 100 fois plus de mémoire pour les césures dans le partitionnement complexe que dans le partitionnement simple. Par conséquent, il est clair que dans certains cas, avec le partage en centaines de classes de séquences de plusieurs centaines de milliers de lettres, la mémoire vive de l'ordinateur est très vite saturée.

La restriction due à la mémoire est surtout manifeste lorsque l'on veut stocker toutes les partitions complexes qui sont créées lors du partitionnement. En effet, pour une partition, l'instance de `Cseq_equ` concernée doit stocker toutes les informations des classes prédictives de l'ensemble des partitions optimales ex æquo. Rien que gérer une 100-partition complexe peut occuper beaucoup d'espace mémoire, mais alors gérer toutes les partitions complexes du partitionnement qui a permis d'aboutir à celle-ci devient gigantesque. Par exemple, un fichier de sortie (voir la partie A.13) d'une 100-partition complexe a une taille de 50 ko, alors que la 100-partition simple de la même séquence est écrite dans un fichier de taille 3 ko, et les classes d'équivalence de cette partition complexe sont écrites dans un fichier de 4 ko. Comme la gestion des partitions est organisée dans le programme d'une façon similaire à leur écriture en fichiers, l'intérêt pour le gain en mémoire des classes d'équivalence apparaît clairement. S'il s'agit de stocker l'ensemble des partitions complexes d'un partitionnement, cet intérêt est encore plus net. Sur le même exemple, la sortie du partitionnement simple occupe un fichier de 109 ko, celle des classes d'équivalence du partitionnement complexe est écrite dans un fichier de 158 ko, et celle de l'ensemble des partitions ex æquo de tous les niveaux du partitionnement occupe 2.6 Mo.

En ce qui concerne la création des partitions complexes par partitionnement en classes d'équivalence, les figures 3.13 et 3.14 illustrent l'efficacité de cette forme de partitionnement et de la reconstruction complète de l'ensemble des partitions optimales à partir des classes d'équivalence. On observe que les performances en temps sont, à un facteur $0,5$ près, semblables à celles de la construction directe de partitions complexes. La création des classes d'équivalence des partitions complexes sans la récupération de l'intégralité des partitions ex æquo est encore un peu plus rapide, puisqu'il n'y a pas l'important nombre d'instances de `Csegment` qui correspondent à toutes les classes prédictives à

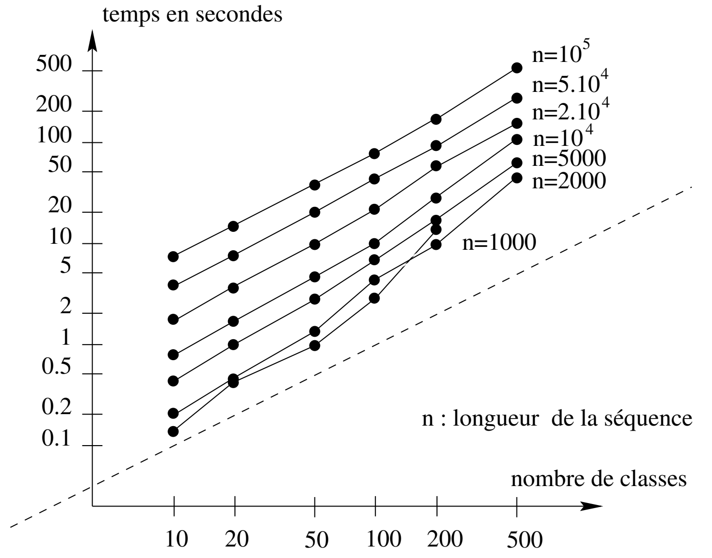


FIG. 3.13 – Courbes du temps de création de k-partitions complexes pour k entre 10 et 2000 sur des séquences de longueur entre 1000 et 100 000 lettres. Ces séquences sont sur un alphabet de quatre lettres, et le lexique est `L = *`. Ces partitions complexes ont été construites sous forme de classes d'équivalence, puis reconstruites sous leur forme intégrale à partir de ces classes d'équivalence. La droite en tirets correspond à la pente des fonctions linéaires.

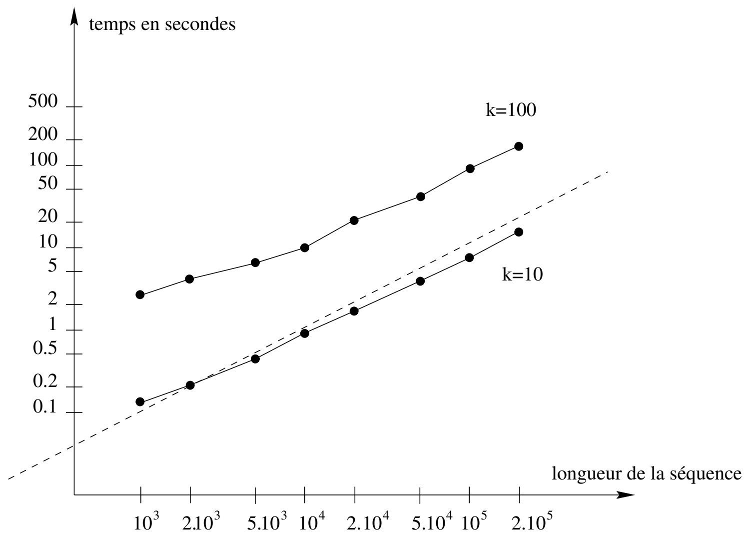


FIG. 3.14 – Courbes du temps de création de k-partitions complexes par l'intermédiaire de la création des classes d'équivalence pour $k = 10$ et $k = 100$ sur des séquences de longueur entre 1000 et 2.10^6 lettres. Ces séquences sont sur un alphabet de quatre lettres, et le lexique est `L = *`. La droite en tirets correspond à la pente des fonctions linéaires.

stocker.

Du point de vue de la complexité en espace mémoire, une instance de la classe `S_Noeud_val<Cposi_motif>` occupe aussi 24 octets, si bien qu'il n'y a rien de perdu par rapport au partitionnement simple. Par contre, la classe `S_Noeud<Cposi_motif>` a besoin de 16 octets au lieu de 12 pour la classe `S_Noeud_val<Cposition>`. Mais si la gestion en classes d'équivalence remplit son rôle qui est de diminuer le nombre de césures, ceci ne doit pas être gênant.

Ces résultats plaident très largement en faveur du partitionnement complexe en classes d'équivalence en comparaison du partitionnement qui construit directement l'intégralité des partitions ex æquo, aussi bien du point de vue du temps que du point de vue de l'espace mémoire.

Comme nous venons de le voir, de par son efficacité en complexité temporelle, il est tout à fait possible de réaliser rapidement des partitionnements maximalement prédictifs sur des séquences qui peuvent être de grande taille et des lexiques qui peuvent être conséquents. Ceci nous a permis d'étudier le comportement du partitionnement, observations qui seront présentées au chapitre quatrième.

Chapitre 4

Comportement

« Two mountain chains traverse the republic roughly from north to south, forming between them a number of valleys and plateaux. »

M. LOWRY
Under the volcano

Grâce au programme `pampres`, le partitionnement maximalement prédictif peut être effectué sur de grandes séquences, et avec une grande variété de lexiques de prédiction. Il s'agit maintenant d'étudier en quoi les niveaux d'un partitionnement diffèrent les uns des autres et s'il est possible de mettre en évidence des propriétés dignes d'intérêt, à la fois pour la compréhension des mécanismes du partitionnement et pour une meilleure gestion de ce type de classification. De plus, on peut espérer que ces propriétés nous permettront d'approfondir, dans ce cas précis, le problème du critère d'arrêt.

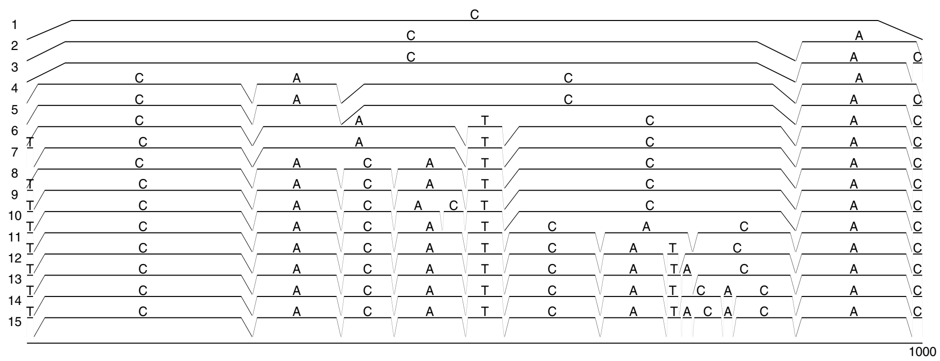
4.1 Observations

4.1.1 Stabilité des classes

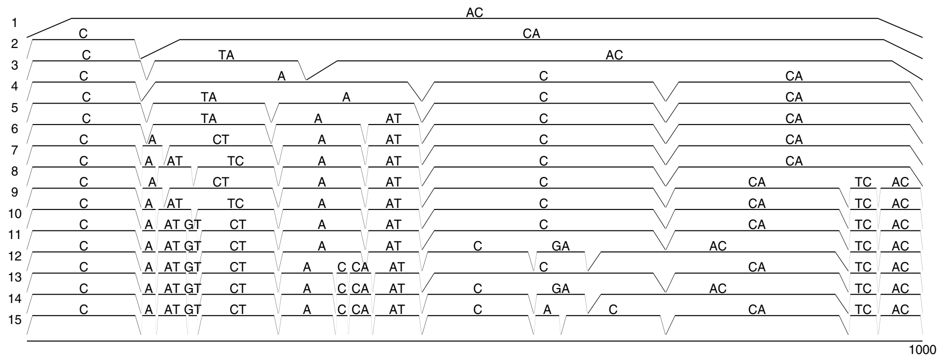
Dans un processus de partitionnement, surtout quand se pose le problème du choix d'une partition, il est nécessaire de s'interroger sur la stabilité des classes des k-partitions du partitionnement en fonction du nombre de classes k. En effet, intuitivement, la pertinence d'une classe est directement liée à sa stabilité dans le partitionnement. Une classe présente sur beaucoup de niveaux[1] du partitionnement a plus d'importance dans la structure de l'ensemble qu'une classe qui n'apparaît que dans une seule k-partition du partitionnement. Par cette approche, la construction d'une partition à partir d'une hiérarchie peut être faite par la sélection dans cette dernière des classes les plus stables (voir 1.4.2).

À la différence d'une hiérarchie, le partitionnement n'est pas seulement un

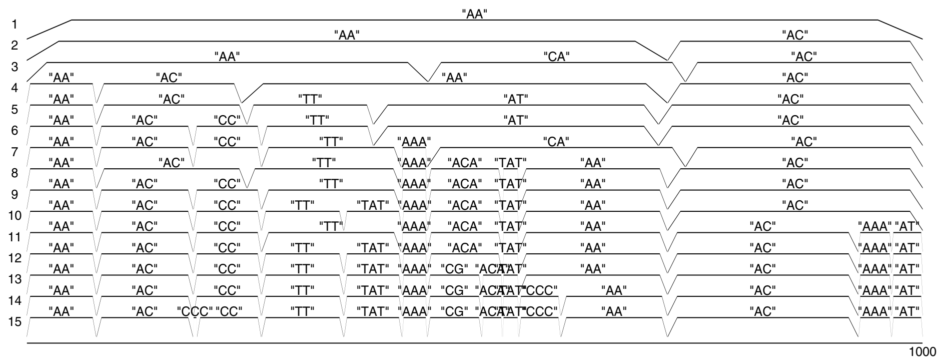
[1] Rappelons qu'on appelle niveau d'un partitionnement un nombre de classes.



(a) `L = *`



(b) `L = **`



(c) `L = "**" "***"(2)`

Fig. 4.1 – Partitions extraites de partitionnements maximalement prédictifs de trois séquences aléatoires, avec trois lexiques différents (syntaxe en 2.4).Chaque séquence est symbolisée par une ligne horizontale, et, pour k allant de 1 à 15, chaque k-partition prédictive est représentée par une suite d'arcs et de mots, représentant les classes prédictives et leurs prédicteurs.

ensemble de classes qui décrit l'organisation de l'ensemble étudié, mais un ensemble de partitions optimales dont chacune décrit, à son propre niveau, l'organisation de l'ensemble. S'il y a beaucoup de variations entre les ensembles de partitions successifs au long du partitionnement, il est, sinon hasardeux, du moins très critique de faire un choix de nombre de classes. Sur une partie de la séquence, que l'ensemble des césures ne présente pas de stabilité témoigne d'une indétermination du partitionnement sur cette partie ; si un choix de moment d'arrêt est fait, une césure existant sur cette partie n'a pas de validité propre et, à l'échelle de cette partie, le niveau choisi est incertain. En revanche, si, parmi plusieurs niveaux consécutifs, une césure de classes apparaît consécutivement dans les partitions, elle est *a priori* plus intéressante. Si plusieurs césures se maintiennent sur plusieurs niveaux, choisir un niveau plutôt qu'un autre parmi ces niveaux n'est pas trop critique en ce qui concerne ces césures.

À l'extrême, si une certaine stabilité des césures est maintenue *tout au long* du partitionnement, les césures valables seront fort probablement présentes dans un quelconque niveau choisi comme niveau d'arrêt. En revanche, il faudra un autre critère que la stabilité pour choisir à quels niveaux le partitionnement est intéressant.

Les figures 4.1 présentent quelques exemples de partitionnements maximalement prédictifs sur différentes séquences aléatoires de mille lettres de l'alphabet `A`, `C`, `G`, `T`, avec leurs lexiques. Le trait horizontal gradué représente la séquence. De haut en bas sont dessinées des k-partitions prédictives, k allant de 1 à 15. Chaque arc représente une classe prédictive. Au-dessus de chaque arc est écrit le prédicteur de la classe prédictive. Pour chaque k, parmi les k-partitions prédictives optimales, celle qui est dessinée est justifiée à gauche.

On remarque que ces trois illustrations ressemblent fortement à des hiérarchies, puisque les césures présentées sont très stables, et que les prédicteurs des classes prédictives persistent au cours du partitionnement. Si la césure gauche d'une classe est déplacée entre deux niveaux successifs, en raison du changement de phase, les mots écrit au-dessus des arcs risquent d'être différents (par exemple `ABC` et `BCA`), mais la prédiction de chaque objet demeure. C'est ce qui se passe sur la figure 4.1.(b), entre les niveaux 1, 2 et 3, sur la classe la plus à droite.

Cette ressemblance avec un processus hiérarchique a toujours été observée, jusqu'aux plus grandes profondeurs. En particulier, il est clair que lorsque deux classes sur un même voisinage gardent les mêmes prédicteurs, il n'y a pas de raison que leur césure commune soit modifiée.

Hors, le partitionnement maximalement prédictif n'est pas conçu comme un processus hiérarchique. En particulier, les $(k+1)$-partitions prédictives optimales ne sont pas construites à partir des k-partitions prédictives optimales. Cependant, à la visualisation des partitions, on peut avoir l'impression de phénomènes de transition entre niveaux. Il s'agit de phénomènes et non de mécanismes, puisque c'est seulement par l'observation qu'ils semblent exister. Ces phénomènes peuvent généralement être décomposés en un suite de phénomènes plus simples. Les deux phénomènes « élémentaires » son présentés ci-dessous.

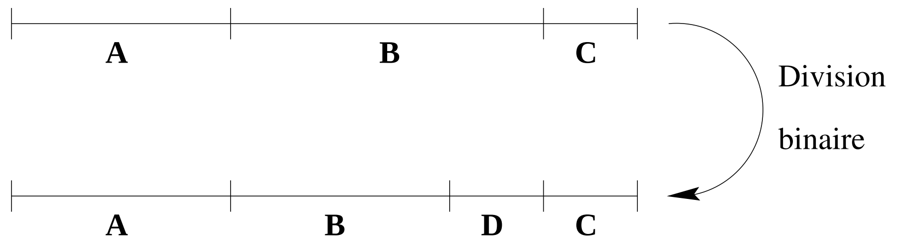
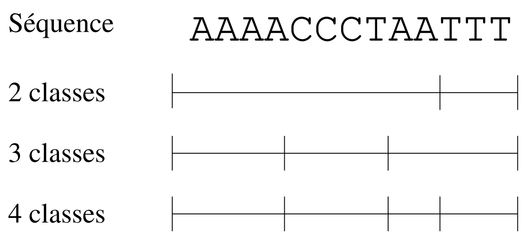


FIG. 4.2 – Division binaire entre deux partitions d'un partitionnement.

Le phénomène le plus courant est celui qui est représenté dans la figure 4.2, à savoir l'apparition d'une classe sur le bord d'une autre. Nous nommerons ce phénomène **division binaire**. L'apparition d'un classe sur le bord d'une autre maintient très souvent les césures, mais pas obligatoirement. Par exemple, le prédicteur de la classe apparue peut venir « chercher » des bonnes prédictions dans une classe voisine, comme dans cet exemple, avec `L=*` :



L'autre phénomène très fréquent, dit **division ternaire**, qui se produit est la « découverte » d'une classe à l'intérieur d'une autre, comme dans la figure 4.3. Comme ce phénomène provoque la création de deux classes d'un coup, il intervient en alternance avec une division binaire, ce qui provoque la disparition d'une classe, comme entre les niveaux 2 et 4 de l'exemple 4.1.a. où la classe la plus à droite prédite par `C` apparue entre les niveaux 2 et 3 disparaît entre les niveaux 3 et 4.

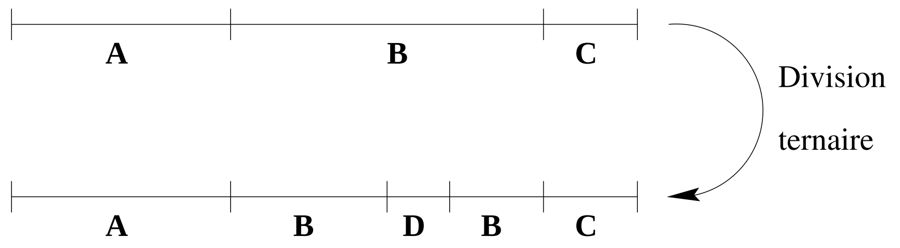
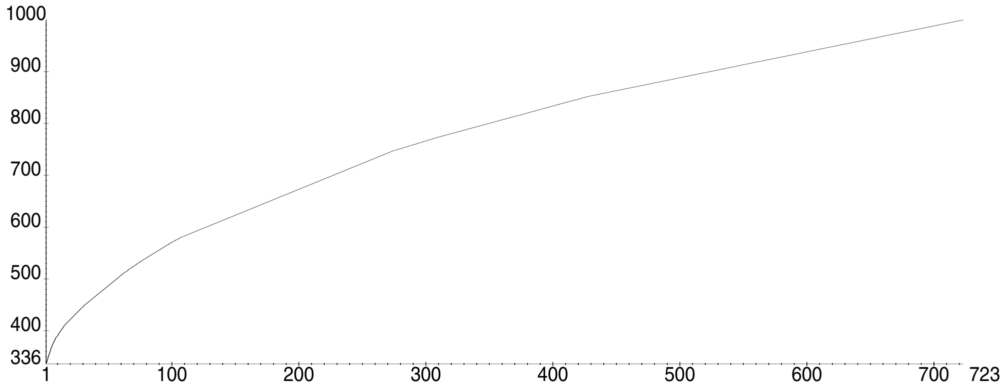
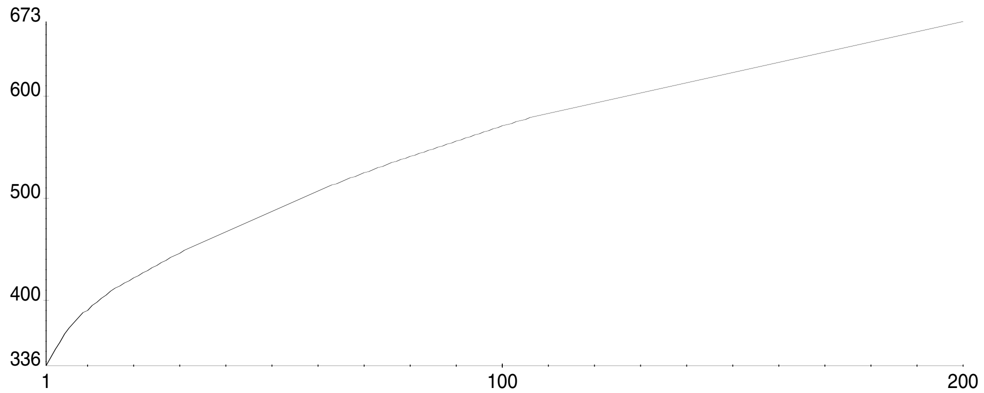


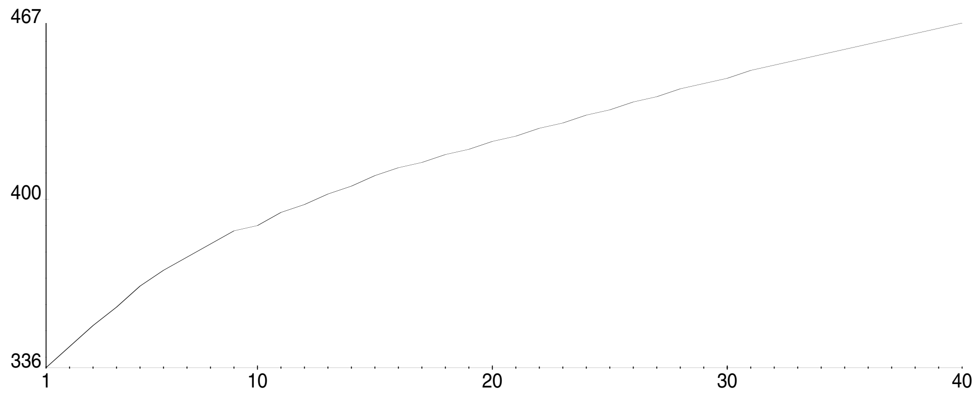
FIG. 4.3 – Division ternaire entre deux partitions d'un partitionnement.



(a) Prédiction en fonction du nombre de classes, jusqu'à la fin du partitionnement

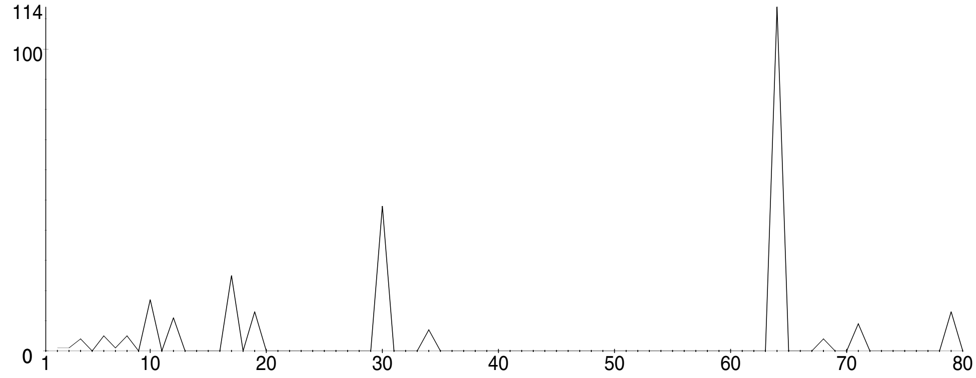


(b) Prédiction en fonction du nombre de classes, jusqu'à 200 classes

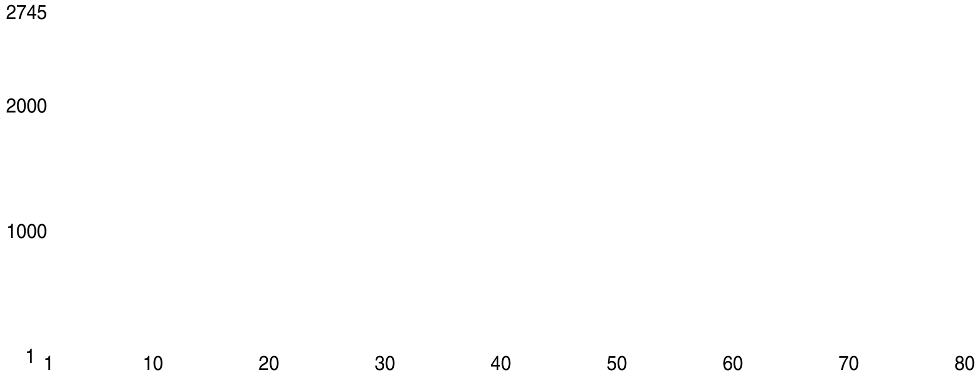


(c) Prédiction en fonction du nombre de classes, jusqu'à 40 classes

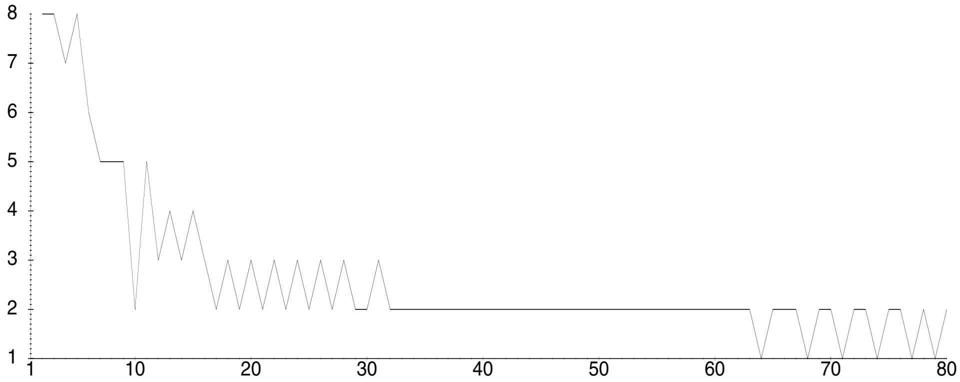
FIG. 4.4 – Prédiction pour l'exemple où `L=*`



(a) Nombre de césures créées en fonction du nombre de classes

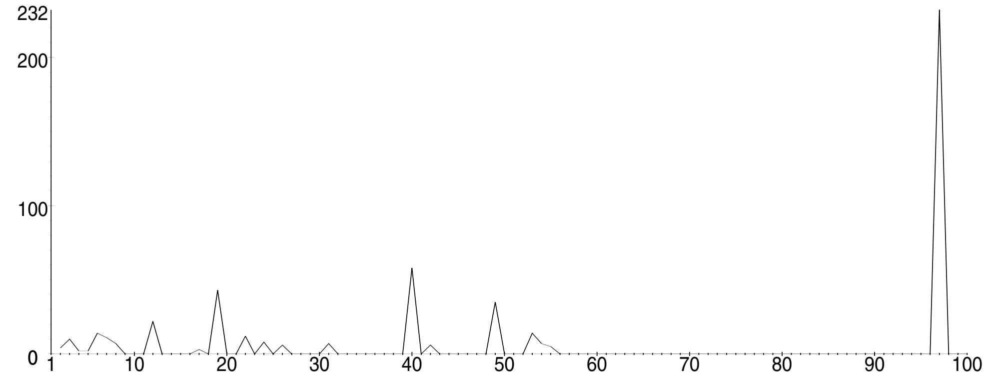


(b) Nombre d'arêtes du graphe des k-partitions en fonction du nombre de classes

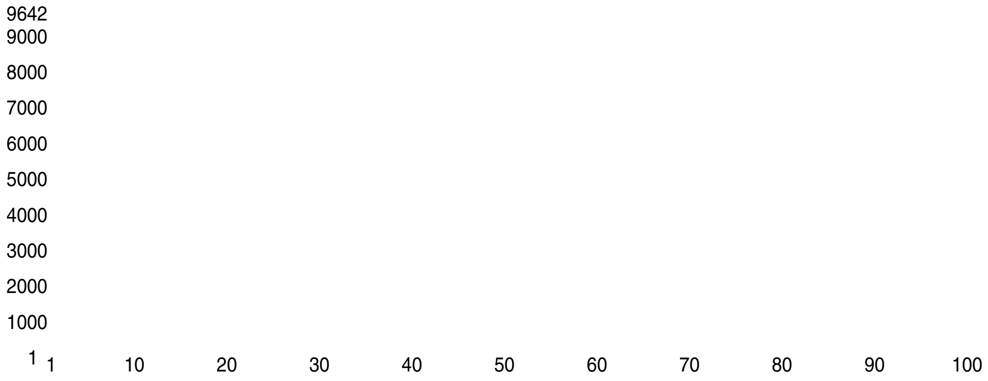


(c) Incrément de prédiction en fonction de nombre de classes

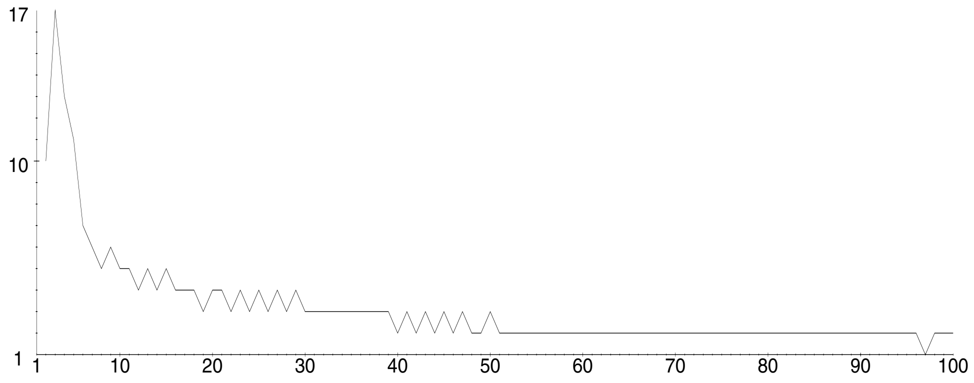
FIG. 4.5 – Courbes pour l'exemple où `L=*`



(a) Nombre de césures créées en fonction du nombre de classes

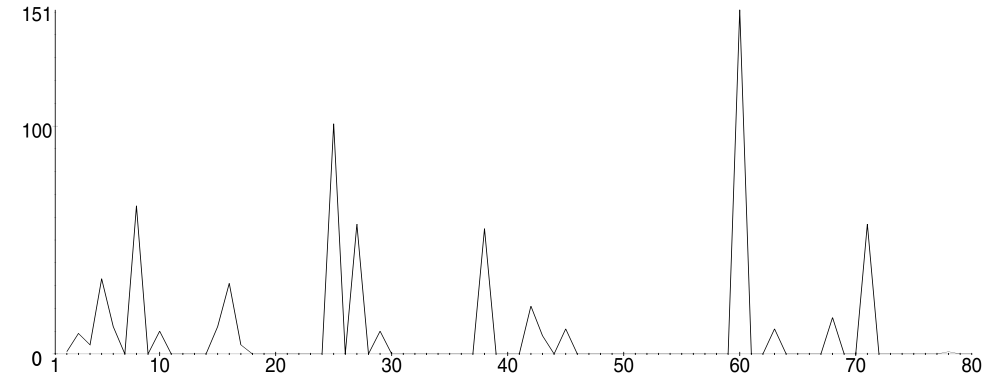


(b) Nombre d'arêtes du graphe des k-partitions en fonction du nombre de classes

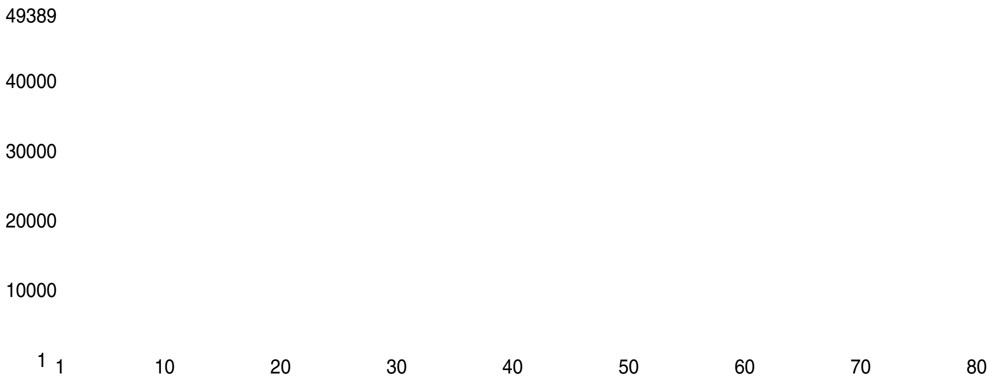


(c) Incrément de prédiction en fonction de nombre de classes

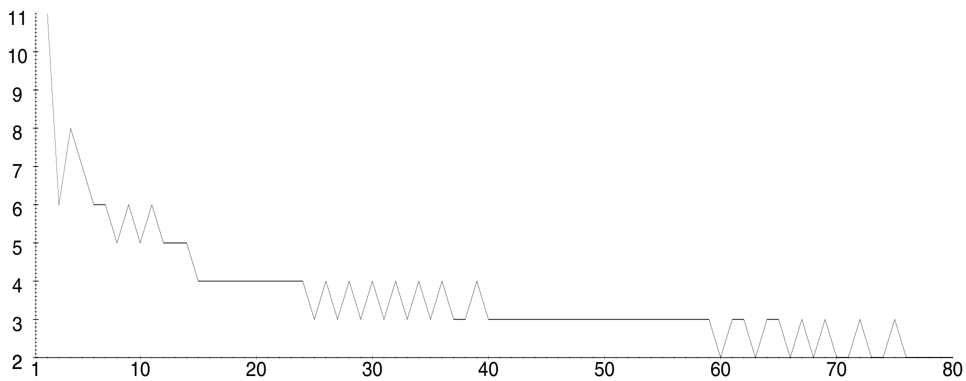
FIG. 4.6 – Courbes pour l'exemple où `L=**`



(a) Nombre de césures créées en fonction du nombre de classes



(b) Nombre d'arêtes du graphe des k-partitions en fonction du nombre de classes



(c) Incrément de prédiction en fonction de nombre de classes

FIG. 4.7 – Courbes pour l'exemple où `L= "**" "***"(2)`

4.1.2 Études de courbes

Comme le principe du partitionnement est de maximiser le gain de prédiction, les phénomènes qui garantissent le plus de gain vont être observés au début du partitionnement, et ceux qui ne permettent que peu de gain vont l'être plus tard. Ceci se voit sur les courbes de la figure 4.4 qui représentent, à différentes échelles, la prédiction du partitionnement 4.1.a en fonction du nombre de classes. Les courbes ont approximativament la même apparence, à savoir une courbe affine par morceaux dont les pentes successives diminuent. Ceci signifie une diminution moyenne du gain de prédiction, et donc des phénomènes (divisions) garantissant en moyenne des gains de plus en plus faibles. Chaque partie de la fonction affine correspond à une phase régulière de la variation de prédiction. Chaque phase régulière est appelée **palier**. Ces paliers sont illustrés dans la figure 4.5.c.

Si, pour chaque partitionnement de la figure 4.1, on trace la courbe du nombre de nouvelles césures créées dans l'ensemble des k-partitions prédictives optimales en fonction du nombre k de classes de ces partitions (courbes 4.5.a, 4.6.a, 4.7.a), on observe que les niveaux où les césures sont créées sont peu nombreux, et surtout regroupés par paquets très nets, hormis lorsqu'il y a encore peu de classes. Ceci signifie que, par moments, le partitionnement voit apparaître de nouvelles césures. Une fois que ces césures sont apparues, elles sont utilisées dans l'ensemble des partitions optimales sur plusieurs niveaux consécutifs du partitionnement sans que de nouvelles césures soient créées.

Nous avons remarqué, page 30, que le nombre d'arêtes du graphe des k-partitions est un critère d'évaluation raisonnable de la validité d'un nombre de classes. Or, on peut noter que c'est exactement juste avant que des césures soient créées qu'il y a le moins d'arêtes dans les graphes des k-partitions (courbes 4.5.b, 4.6.b, 4.7.b). En outre, l'observation des figures 4.5.c, 4.6.c, et 4.7.c montre que les moments de création en grand nombre des césures sont corrélés avec les changements de palier de l'incrément de prédiction.

En effet, hormis au début du partitionnement, la courbe de l'incrément de prédiction est constituée de paliers similaires à ceux de la figure 4.8.

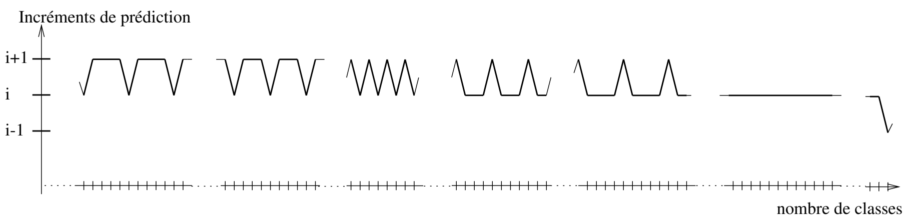


FIG. 4.8 – Exemple d'enchaînement de paliers dans la courbe des incréments de prédiction en fonction du nombre de classes.

Dans les faits, généralement, au sein de chaque palier, les gains de prédiction sont corrélés avec l'observation de mélanges de divisions binaires et ternaires.

Le cas le plus simple est celui où le gain d'incrément g est constant. Il traduit le plus souvent une suite de divisions binaires dont tous les gains sont égaux à g.

Si, sur une partition optimale, il y a plusieurs positions où diviser une classe en deux permet le même gain de prédiction, et aucune division ternaire ne permet de gagner $2g$ ou plus, ce sont ces divisions binaires qui vont se produire successivement. Si les positions des divisions sont assez éloignées les unes des autres pour être indépendantes, c'est-à-dire qu'une division peut se produire en une position quelconque sans qu'il soit nécessaire qu'une autre se soit déjà produite, et qu'une division en une position quelconque peut se produire alors que n'importe quelles autres divisions se sont déjà produites, le partitionnement a un comportement très simple : à partir d'une k-partition optimale, s'il y a x divisions binaires, il y a x $(k+1)$-partitions optimales, puis, par combinaison, C_x^2 $(k+2)$-partitions optimales, et ainsi de suite, C_x^i $(k+i)$-partitions optimales. Lorsque $i = x$, les partitions doivent posséder les x classes créées lors de ces x divisions, si bien qu'il n'y a qu'une seule $(k+x)$-partition. Il faudra créer de nouvelles césures pour continuer le partitionnement. Ce type de phénomène se produit pour l'ensemble des partitions optimales. Ceci explique la régularité de la courbe du nombre d'arêtes des graphes, ainsi que le petit nombre de partitions optimales à la fin de ce genre de paliers.

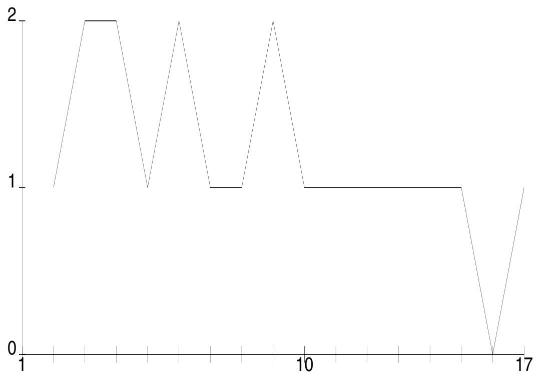
Les paliers où le gain d'incrément est de période 2 dans la courbe des incréments correspond aux divisions ternaires. Comme une telle division crée deux classes, il y a alternance entre une division binaire, qui disparaît lorsque la division ternaire devient effective. Si la division binaire provoque un gain g, et la division ternaire un gain h, la prédiction croît successivement de g, $h-g$, puis g. Les classes qui sont créées lors des divisions ternaires doivent donc garantir un gain suffisant. De la même manière que ci-dessus, lorsqu'une suite de telles divisions est entamée à partir d'une partition optimale, il y a plusieurs candidats possibles de même gain parmi ces divisions, et la longueur du palier correspond au temps nécessaire à leur intégration à toutes dans une même partition optimale. Le même calcul que ci-dessus peut être fait en ne considérant qu'un nombre de classes sur deux, et explique que la courbe du nombre d'arêtes des graphes des k-partitions suive de façon alternée deux courbes parallèles à tendance parabolique.

Les paliers correspondent le plus souvent à des phénomènes similaires, ou à des mélanges de divisions binaires et ternaires, ce qui provoque que le gain de prédiction est localement de périodes différentes, comme par exemple dans la figure 4.9.

Bien sûr, comme il ne s'agit pas d'un processus hiérarchique, il se peut que les césures soient modifiées, si bien que les phénomènes peuvent être beaucoup plus compliqués que le simple mélange de divisions binaires et ternaires, mais celles-ci sont très largement les plus courantes.

La longueur d'un palier est proportionnelle au produit du nombre de phénomènes de comportements similaires qui s'y produisent et de la durée de ces phénomènes. On peut s'attendre à ce que si les phénomènes d'un même type sont nombreux, ils agissent sur de petites parties de la séquence, et ainsi ne permettent chacun que peu de gain. Ainsi, plus la prédiction croît en moyenne lentement sur un palier, plus cette phase de croissance va couvrir de nombreux niveaux du partitionnement. C'est ce que l'on voit sur la courbe affine par morceaux de la figure 4.4 : les morceaux de cette courbe sont en moyenne de plus

(a) partitionnement avec `L = *`



(b) Courbe des incréments de prédiction

FIG. 4.9 – Partitionnement et courbe de l'incrément de prédiction de ce partitionnement avec `L = *`. Les phénomènes composés de divisions binaires et ternaires engendrent des motifs de périodes variables dans la courbe des incréments de prédiction.

en plus longs alors que leur pente est de plus en plus faible.

Si on observe le début du partitionnement, il n'apparaît pas de réels paliers. Ceci est dû au fait qu'au début du partitionnement l'incrément de prédiction est très grand, si bien que les parties de la séquence sur lesquelles sont visibles les phénomènes de division sont très grandes. Par conséquent, ces phénomènes sont trop peu nombreux pour qu'il y ait de réels paliers de régularités.

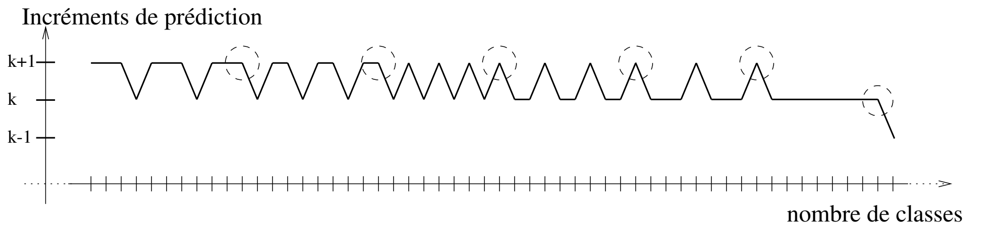


FIG. 4.10 – Les moments stables sur la courbe des incréments de prédiction sont entourés d'un cercle en pointillés.

D'un point de vue pratique, le plus important dans ces observations est la corrélation entre, d'une part, les fins de paliers de la courbe des incréments de prédiction et, d'autre part, les niveaux du partitionnement où la plupart des ambiguïtés entre partitions optimales sont résolues. En dehors de ces niveaux, l'irrésolution quant au choix des partitions optimales est d'autant plus grande qu'il y a d'endroits sur la séquence où il est possible d'avoir un même gain de prédiction. Ceci condamne le nombre de classes en question comme éventuel choix du bon nombre de classes. Ainsi, la donnée simple de la fonction de prédiction permet de connaître les niveaux du partitionnement où il y a peu d'ambiguïté, sans avoir besoin de construire l'ensemble des partitions optimales. Ces niveaux sont appelés **moments stables** (figure 4.10).

Dans cette étude empirique des partitions successives d'un partitionnement, nous avons mis en évidence le comportement presque hiérarchique du partitionnement, essentiellement dû à la combinaison de divisions binaires et ternaires. De plus, nous avons remarqué une forte corrélation entre la variation de l'incrément de prédiction et les moments stables.

Comme nous allons le montrer maintenant, c'est rigoureusement ce qui se passe dans le cas où il n'y a que deux prédicteurs. Même s'il s'agit d'un cas très particulier, on peut espérer que la détermination précise des phénomènes qui se produisent dans le cas où il n'y a que deux prédicteurs, et les différentes propriétés que l'on peut en déduire, permettront de mieux appréhender le comportement du partitionnement dans le cas général.

4.2 Étude théorique dans le cas de deux prédicteurs

Si l'ensemble des prédicteurs admissibles D est de cardinal 2, et a et b des prédicteurs d'objets, on peut avoir $D = \{a, b\}$ ou bien $D = \{ab, ba\}$ ou encore $D = \{a, ?(0.5)\}$. Nous allons voir que dans ce cas, le partitionnement peut être assimilé à une succession de divisions binaires et ternaires. De plus, on vérifie qu'aux moments stables, il n'y a au plus que deux classes d'équivalences et que ces moments sont corrélés avec la diminution stricte de la variation entre niveaux de même parité de la prédiction.

Soit $X \in \mathbb{M}_k^D(\Omega)$ une partition prédictive optimale définie sur l'ensemble de prédicteurs D, et soit $\omega_i \in \Omega$, rappelons que $\hat{\pi}_X(\omega_i)$ est le prédicteur d'objet affecté à ω_i par X. X est dite propre si, quelles que soient les classes prédictives de X, $(C, \underline{\delta})$ et $(C', \underline{\delta'})$ contiguës, $\underline{\delta} \neq \underline{\delta'}$.

Soient $t, u \in \langle 0, n \rangle$ ($n = |\Omega|$) avec $t < u$. On note $X/\langle t, u \rangle$ la partition prédictive X restreinte à la sous-séquence de Ω, $\omega_{\langle t+1, u \rangle}$.

Soit $X \in \mathbb{M}_k^D(\Omega)$ et $Y \in \mathbb{M}_l^D(\Omega)$, on définit :

$$(X|Y) = \begin{cases} n+1 & \text{si } \forall j \in \langle 1, n \rangle : \hat{\pi}_X(\omega_j) = \hat{\pi}_Y(\omega_j) \\ \min\{j \in \langle 1, n \rangle : \hat{\pi}_X(\omega_j) \neq \hat{\pi}_Y(\omega_j)\} & \text{sinon} \end{cases}$$

La première proposition signifie que de toute k-partition prédictive optimale, on peut construire une $(k+2)$-partition prédictive optimale par une division ternaire ou deux divisions binaires sur les extrémités de la séquence.

FIG. 4.11 – Illustration de la proposition 4.2.1 avec $D = \{a, b\}$. De $X \in \mathbb{M}_k^D(\Omega)$, il existe $Y \in \mathbb{M}_{k+2}^D(\Omega)$ tel que l'un des deux passages de X à Y existe. Sur les parties pointillées, X et Y sont identiques.

Proposition 4.2.1. *Si $|D| = 2$, soit k tel que $M_{k+2}^D(\Omega) > M_{k+1}^D(\Omega)$.*
Soit $X \in \mathbb{M}_k^D(\Omega)$ une partition prédictive propre, il existe $Y \in \mathbb{M}_{k+2}^D(\Omega)$ tel que l'on puisse passer de X à Y, soit par une division ternaire, soit par deux divisions binaires qui interviennent sur les extrémités de Ω.

Tout d'abord un lemme (illustré dans la figure 4.12), qui découle de l'optimalité des sous-partitions d'une partition, et qui est valable avec un ensemble de prédicteurs D quelconque :

Lemme 4.2.1. *Soient $X = \{(x_i, \underline{\delta_i})\}_{i \in \langle 0, k-1 \rangle} \in \mathbb{M}_k^D(\Omega)$ et*
$Y = \{(y_i, \underline{\gamma_i})\}_{i \in \langle 0, l-1 \rangle} \in \mathbb{M}_l^D(\Omega)$.
S'il existe $0 \leqslant p < q < k$ et $0 \leqslant r < s < l$ tels que :
- *$\underline{\delta_p} = \underline{\gamma_r}$, $x_p < y_{r+1}$ et $y_r < x_{p+1}$*
- *$\underline{\delta_q} = \underline{\gamma_s}$, $x_q < y_{s+1}$ et $y_s < x_{q+1}$*
- *$q - p = s - r$*

alors la partition prédictive :
$\{(x_0, \underline{\delta_0}), \dots, (x_p \underline{\delta_p}), (y_{r+1}, \underline{\gamma_{r+1}}), \dots, (y_s, \underline{\gamma_s}), (x_{q+1}, \underline{\delta_{q+1}}), \dots, (x_{k-1}, \underline{\delta_{k-1}})\}$
appartient à $\mathbb{M}_k^D(\Omega)$

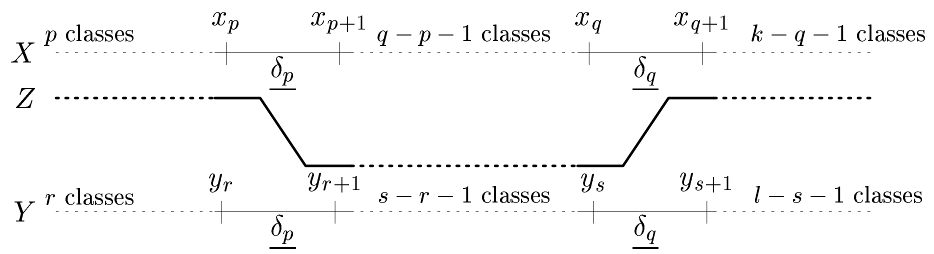


FIG. 4.12 – Illustration du lemme 4.2.1 : $X \in \mathbb{M}_k^D(\Omega)$, $Y \in \mathbb{M}_l^D(\Omega)$. La partition Z est semblable à X lorsque le trait qui la représente est proche de de celui qui représente X, et de même pour Y. Les segments – obliques – qui sont entre ces parties de similarité correspondent à des parties où X et Y sont semblables. $Z \in \mathbb{M}_k^D(\Omega)$.

■ **Démonstration du lemme 4.2.1** :

Soit $Z = \{(x_0, \underline{\delta_0}), \ldots, (x_p, \underline{\delta_p}), (y_{r+1}, \underline{\gamma_{r+1}}), \ldots$
$\ldots, (y_s, \underline{\gamma_s}), (x_{q+1}, \underline{\delta_{q+1}}), \ldots, (x_{k-1}, \underline{\delta_{k-1}})\}$.

Z possède $p + 1 + (s - r - 1) + 1 + k - (q + 1) = k$ classes.

Soit t tel que $x_p \leqslant t < x_{p+1}$ et $y_r \leqslant t < y_{r+1}$ (t existe grâce aux hypothèses). De même, soit u tel que $x_q \leqslant u < x_{q+1}$ et $y_s \leqslant u < y_{s+1}$.

- Comme $t \in \langle x_p, x_{p+1}\rangle$, $Z/\langle x_0, t\rangle = X/\langle x_0, t\rangle$.
- Comme $t \in \langle y_r, y_{r+1}\rangle$ et $u \in \langle y_s, y_{s+1}\rangle$, $Z/\langle t, u\rangle = Y/\langle t, u\rangle$.
- Comme $u \in \langle x_q, x_{q+1}\rangle$, $Z/\langle u, n\rangle = X/\langle u, n\rangle$.

<u>Supposons $\pi(X) > \pi(Z)$</u> :

$\pi(X/\langle t, u\rangle) > \pi(Z/\langle t, u\rangle)$ car X et Z sont semblables ailleurs.

Or, $Z/\langle t, u\rangle = Y/\langle t, u\rangle$ donc

$$\pi(X/\langle t, u\rangle) > \pi(Y/\langle t, u\rangle)$$

La partition prédictive
$W = \{(y_0, \underline{\gamma_0}), \ldots, (y_r, \underline{\gamma_r}), (x_{p+1}, \underline{\delta_{p+1}}), \ldots$
$\ldots, (x_q, \underline{\delta_q}), (y_{s+1}, \underline{\gamma_{s+1}}), \ldots, (y_{l-1}, \underline{\gamma_{l-1}})\}$
possède $r + 1 + (q - p - 1) + 1 + l - (s + 1) = l$ classes.

De la même manière que pour Z :

- Comme $t \in \langle y_r, y_{r+1}\rangle$, $W/\langle y_0, t\rangle = Y/\langle y_0, t\rangle$.
- Comme $t \in \langle x_p, x_{p+1}\rangle$ et $u \in \langle x_r, x_{r+1}\rangle$, $W/\langle t, u\rangle = X/\langle t, u\rangle$.
- Comme $u \in \langle y_s, y_{s+1}\rangle$, $W/\langle u, n\rangle = Y/\langle u, n\rangle$.

Donc :

$$\begin{aligned} \pi(W) - \pi(Y) &= \pi(W/\langle t, u\rangle) - \pi(Y/\langle t, u\rangle) \\ &= \pi(X/\langle t, u\rangle) - \pi(Y/\langle t, u\rangle) \\ &> 0 \end{aligned}$$

et $\pi(W) > \pi(Y)$, ce qui est faux puisque $Y \in \mathbb{M}_l^D(\Omega)$.

Par conséquent, $\pi(X) \leqslant \pi(Z)$ et, comme $X \in \mathbb{M}_k^D(\Omega)$, $Z \in \mathbb{M}_k^D(\Omega)$.

□

Un autre lemme général, illustré dans la figure 4.13 :

Lemme 4.2.2. *Soient* $X = \{(x_i, \underline{\delta_i})\}_{i \in \langle 0,k-1\rangle} \in \mathbb{M}_k^D(\Omega)$ *et*
$Y = \{(y_i, \underline{\gamma_i})\}_{i \in \langle 0,l-1\rangle} \in \mathbb{M}_l^D(\Omega)$.
S'il existe $i \in \langle 1, k-1\rangle$ *et* $j \in \langle 1, l-1\rangle$ *tels que*
$x_i = y_j$, $\underline{\delta_i} = \underline{\gamma_{j-1}}$ *et* $\underline{\gamma_j} = \underline{\delta_{i-1}}$, *alors :*

$$\forall x \in \langle \max\{x_{i-1}, y_{j-1}\}, \min\{x_{i+1}, y_{j+1}\}\rangle, \pi_{\underline{\delta_i}}(x_i, x) = \pi_{\underline{\gamma_j}}(x_i, x)$$

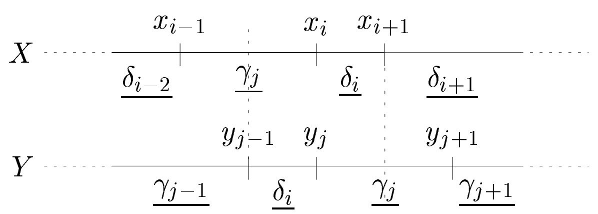


FIG. 4.13 – Illustration du lemme 4.2.2 : Si $X \in \mathbb{M}_k^D(\Omega)$ et $Y \in \mathbb{M}_l^D(\Omega)$, toute partie $\langle x_i, x\rangle$ pour $x \in \langle y_{j-1}, x_{i+1}\rangle$ a la même prédiction sur X et sur Y.

■ **Démonstration du lemme 4.2.2** :

Par exemple, $y_{j+1} \geqslant x_{i+1}$.

S'il existe $x \in \langle x_i, x_{i+1}\rangle$ tel que $\pi_{\underline{\delta_i}}(x_i, x) < \pi_{\underline{\gamma_j}}(x_i, x)$:

Comme $\underline{\gamma_j} = \underline{\delta_{i-1}}$,

$$\begin{aligned}
\pi_{\underline{\delta_{i-1}}}(x_{i-1}, x) + \pi_{\underline{\delta_i}}(x, x_{i+1}) &= \pi_{\underline{\delta_{i-1}}}(x_{i-1}, x_i) + \pi_{\underline{\gamma_j}}(x_i, x) + \pi_{\underline{\delta_i}}(x, x_{i+1}) \\
&> \pi_{\underline{\delta_{i-1}}}(x_{i-1}, x_i) + \pi_{\underline{\delta_i}}(x_i, x) + \pi_{\underline{\delta_i}}(x, x_{i+1}) \\
&> \pi_{\underline{\delta_{i-1}}}(x_{i-1}, x_i) + \pi_{\underline{\delta_i}}(x_i, x_{i+1})
\end{aligned}$$

et la k-partition

$$\{(x_0, \underline{\delta_0}), \ldots, (x_{i-1}, \underline{\delta_{i-1}}), (x, \underline{\delta_i}), (x_{i+1}, \underline{\delta_{i+1}}), \ldots, (x_{k-1}, \underline{\delta_{k-1}})\}$$

a une prédiction strictement supérieure à celle de X, ce qui contredit l'optimalité de X.

De la même manière, s'il existe $x \in \langle x_i, x_{i+1}\rangle$ tel que $\pi_{\underline{\delta_i}}(x_i, x) > \pi_{\underline{\gamma_j}}(x_i, x)$, il y a contradiction avec l'optimalité de Y.

Le même type de raisonnement est valable à gauche de la césure étudiée.

□

Voici un corollaire du lemme lorsqu'il n'y a que deux prédicteurs admissibles. Si les deux partitions prédictives X et Y sont propres, avec $x_i = y_j$. Si, en outre, $\underline{\gamma_j} \neq \underline{\delta_i}$ alors on a, en raison de l'alternance des prédicteurs, $\underline{\delta_i} = \underline{\gamma_{j-1}}$ et $\underline{\delta_{i-1}} = \underline{\gamma_j} = \underline{\delta_{i+1}}$

On suppose $i < k-1$. Si, par exemple, $x_{i+1} \leqslant y_{j+1}$,

$$\begin{aligned}
&\pi_{\underline{\delta_{i-1}}}(x_{i-1}, x_i) + \pi_{\underline{\delta_i}}(x_i, x_{i+1}) + \pi_{\underline{\delta_{i+1}}}(x_{i+1}, x_{i+2}) \\
&\qquad = \pi_{\underline{\delta_{i-1}}}(x_{i-1}, x_i) + \pi_{\underline{\gamma_j}}(x_i, x_{i+1}) + \pi_{\underline{\delta_{i+1}}}(x_{i+1}, x_{i+2}) \\
&\qquad = \pi_{\underline{\delta_{i-1}}}(x_{i-1}, x_{i+1})
\end{aligned}$$

et par conséquent $M_k^D(\Omega) = M_{k-2}^D(\Omega)$.

Si $i = k-1$, le résultat du lemme appliqué de la même façon entre x_{i-2} et x_i permet d'aboutir à la même conclusion.

Par conséquent :

Corrolaire. *Si $|D| = 2$, soient* $X = \{(x_i, \underline{\delta_i})\}_{i\in\langle 0,k-1\rangle} \in \mathbb{M}_k^D(\Omega)$ *et*
$Y = \{(y_i, \underline{\gamma_i})\}_{i\in\langle 0,l-1\rangle} \in \mathbb{M}_l^D(\Omega)$.

On suppose que $M_k^D(\Omega) \neq M_{k-2}^D(\Omega)$. Pour tous $i, j \in \langle 1, k-1\rangle$ tels que $\underline{\delta_i} \neq \underline{\gamma_j}$, on a :

$$x_i \neq y_j$$

■ **Démonstration de la proposition 4.2.1** :

On note $X = \{(x_i, \underline{\delta_i})\}_{i\in\langle 0,k-1\rangle} \in \mathbb{M}_k^D(\Omega)$.

Comme $M_{k+2}^D(\Omega) > M_{k+1}^D(\Omega)$, il existe $Y = \{(y_i, \underline{\gamma_i})\}_{i\in\langle 0,k+1\rangle} \in \mathbb{M}_{k+2}^D(\Omega)$ une partition prédictive propre qui maximise $(X|Y)$ dans $\mathbb{M}_{k+2}^D(\Omega)$.

Il y a deux possibilités :

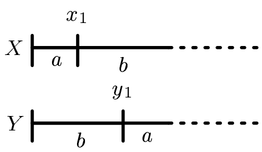
– Si $\delta_0 \neq \gamma_0$:

Par propreté de X et de Y, leurs prédicteurs alternent, si bien que

$$\forall j, \underline{\delta_{j-1}} = \underline{\gamma_j} \text{ et } \underline{\delta_{k-1}} \neq \underline{\gamma_{k+1}}$$

Le corollaire du lemme 4.2.2 interdit $x_1 = y_1$. Par conséquent, il y a deux possibilités :

– Si $x_1 < y_1$:



$$\begin{aligned} X/\langle x_1, n\rangle &= \{(x_1, \underline{\delta_1}), \ldots, (x_{k-1}, \underline{\delta_{k-1}})\} \\ &\in \mathbb{M}_{k-1}^D(\omega_{\langle x_1+1,n\rangle}) \end{aligned}$$

et

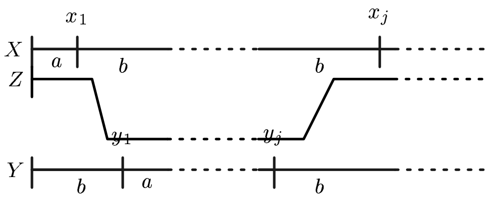
$$\begin{aligned} Y/\langle x_1, n\rangle &= \{(x_1, \underline{\gamma_0}), ((y_1), \underline{\gamma_1}), \ldots, (y_{k+1}, \underline{\gamma_{k+1}})\} \\ &\in \mathbb{M}_{k+2}^D(\omega_{\langle x_1+1,n\rangle}) \end{aligned}$$

Y a $(k+2)$ classes et X a $(k-1)$ classes sur $\omega_{\langle x_1+1,n\rangle}$, si bien qu'il existe $j > 0$ tel que :
$\omega_{\langle x_{j-1}+1,x_j\rangle} \cap \omega_{\langle y_j+1,y_{j+1}\rangle} \neq \emptyset$.

On considère la partition prédictive :

$$Z = \{(x_0, \underline{\delta_0}), (x_1, \underline{\gamma_0}), (y_1, \underline{\gamma_1}), \ldots, (y_j, \underline{\gamma_j}), (x_j, \underline{\delta_j}), \ldots, (x_{k-1}, \underline{\delta_{k-1}})\}$$

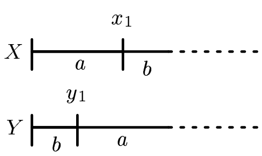
Grâce au lemme 4.2.1, $Z \in \mathbb{M}_{k+2}^D(\Omega)$. Il est clair que $(X|Y) < (X|Z)$, si bien qu'il y a contradiction avec l'hypothèse sur Y.



– Si $x_1 > y_1$:

Par le corollaire ci-dessus, comme $\underline{\delta_{k-1}} \neq \underline{\gamma_{k+1}}$, on a

$$x_{k-1} \neq y_{k+1}$$

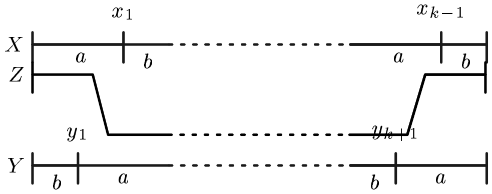


Deux cas se présentent :

- Si $x_{k-1} > y_{k+1}$, en considérant :

$$Z = \{(x_0, \underline{\delta_0}), (y_2, \underline{\gamma_2}), \ldots, (y_{k+1}, \underline{\gamma_{k+1}}), (x_{k-1}, \underline{\delta_{k-1}})\}$$

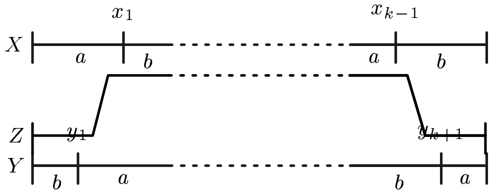
Grâce au lemme 4.2.1, $Z \in \mathbb{M}^D_{k+2}(\Omega)$. Il est clair que $(X|Y) < (X|Z)$, si bien qu'il y a contradiction avec l'hypothèse sur Y.



- Si $x_{k-1} < y_{k+1}$, en considérant :

$$Z = \{(y_0, \underline{\gamma_0}), (y_1, \underline{\gamma_1}), (x_1, \underline{\delta_1}), \ldots, (x_{k-1}, \underline{\delta_{k-1}}), (y_{k+1}, \underline{\gamma_{k+1}})\}$$

$Z \in \mathbb{M}^D_{k+2}(\Omega)$ et on passe de X à Z par deux divisions binaires qui se font aux extrémités de Ω (sur les parties $\omega_{\langle 1, y_1\rangle}$ et $\omega_{\langle y_{k+1}+1, n\rangle}$).



- Si $\delta_0 = \gamma_0$:
Les prédicteurs de X alternent, de même que ceux de Y, si bien que :

$$\forall j, \underline{\delta_j} = \underline{\gamma_j} \text{ et } \underline{\delta_{k-1}} = \underline{\gamma_{k+1}}$$

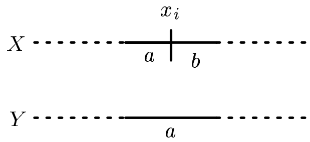
Comme X et Y sont propres et n'ont pas le même nombre de classes, il y a forcément des positions où leurs fonctions de prédiction sont différentes. On regarde la plus petite telle position. La césure correspondante peut être sur X ou sur Y :

- Si elle est sur X, en $x_i < y_i$:

$$X/\langle x_i, n\rangle = \{(x_i, \underline{\delta_i}), \ldots, (x_{k-1}, \underline{\delta_{k-1}})\} \in \mathbb{M}^D_{k-i}(\omega_{\langle x_1+1, n\rangle})$$

et

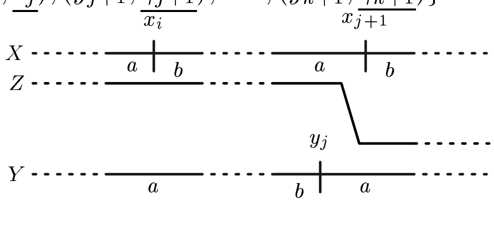
$$Y/\langle x_i, n\rangle = \{(x_i, \underline{\gamma_{i-1}}), (y_i, \underline{\gamma_i}), \ldots, (y_{k+1}, \underline{\gamma_{k+1}})\} \in \mathbb{M}^D_{k-i+3}(\omega_{\langle x_1+1, n\rangle})$$



Y a $(k-i+3)$ classes et X a $(k-i)$ classes sur $\omega_{\langle x_i+1, n\rangle}$, si bien qu'il existe $j > 0$ tel que
$\omega_{\langle x_j+1, x_{j+1}\rangle} \cap \omega_{\langle y_j+1, y_{j+1}\rangle} \neq \emptyset$
On considère la partition prédictive :

$$Z = \{(x_0, \underline{\delta_0}), \ldots, (x_j, \underline{\delta_j}), (y_{j+1}, \underline{\gamma_{j+1}}), \ldots, (y_{k+1}, \underline{\gamma_{k+1}})\}$$

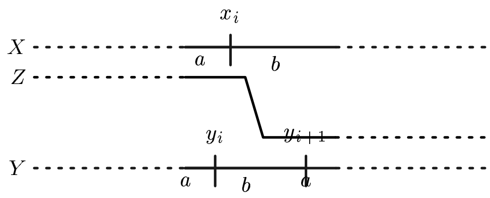
Grâce au lemme 4.2.1, $Z \in \mathbb{M}^D_{k+2}(\Omega)$. Il est clair que $(X|Y) < (X|Z)$, si bien qu'il y a contradiction avec l'hypothèse sur Y.



- Si la césure est sur Y, en y_1 avec $x_{i-1} < y_i < x_i$
  Encore deux éventualités
  - Si $x_i < y_{i+1}$:
    Soit :

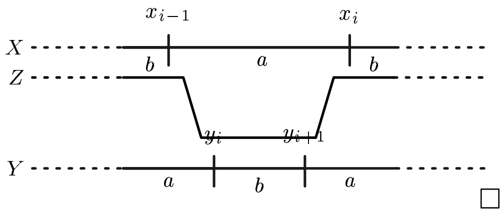
$$Z = \{(x_0, \underline{\delta_0}), \ldots, (x_i, \underline{\delta_i}), (y_{i+1}, \underline{\gamma_{i+1}}), \ldots, (y_{k+1}, \underline{\delta_{k+1}})\}$$

    Grâce au lemme 4.2.1, $Z \in \mathbb{M}^D_{k+2}(\Omega)$. Il est clair que $(X|Y) < (X|Z)$, si bien qu'il y a contradiction avec l'hypothèse sur Y.



  - Si $y_{i+1} < x_i$, en prenant :
    $Z = \{(x_0, \underline{\delta_0}), \ldots, (x_{i-1}, \underline{\delta_{i-1}}), (y_i, \underline{\gamma_i}),$
    $(y_{i+1}, \underline{\gamma_{i+1}}), (x_i, \underline{\delta_i}), \ldots, (x_{k-1}, \underline{\delta_{k-1}})\}$

    $Z \in \mathbb{M}^D_{k+2}(\Omega)$ et on passe de X à Z par une division ternaire, sur la partie $\omega_{\langle y_i+1, y_{i+1}\rangle}$.



$\square$

Ainsi, une partie des $(k+2)$-partitions prédictives optimales de $\mathbb{M}^D_{k+2}(\Omega)$ peut être considérée comme un ensemble de partitions déduites de *toutes* les k-partitions prédictives propres optimales $\mathbb{M}^D_k(\Omega)$ par des divisions ternaires ou binaires sur les extrémités de la séquence. Il s'agit maintenant, en retour, de vérifier que toute $(k+2)$-partition prédictive optimale est le résultat d'une transformation similaire à partir d'une k-partition prédictive optimale, comme cela est illustré dans la figure 4.14.

FIG. 4.14 – $D = \{a, b\}$. Les deux types de construction de $X \in \mathbb{M}^D_{k+2}(\Omega)$ à partir de $Y \in \mathbb{M}^D_k(\Omega)$. Sur les parties pointillées, X et Y sont identiques.

Proposition 4.2.2. *Si $|D| = 2$, soit $X \in \mathbb{M}^D_{k+2}(\Omega)$ une partition prédictive propre, il existe $Y \in \mathbb{M}^D_k(\Omega)$ tel que l'on puisse passer de Y à X par une division ternaire ou deux divisions binaires qui interviennent alors sur les extrémités de Ω.*

■ **Démonstration de la proposition 4.2.2** :

On note $X = \{(x_i, \underline{\delta_i})\}_{i\in\langle 0,k+1\rangle} \in \mathbb{M}^D_{k+2}(\Omega)$.

Soit $Y = \{(y_i, \underline{\gamma_i})\}_{i\in\langle 0,k-1\rangle} \in \mathbb{M}^D_k(\Omega)$ une k-partition prédictive propre qui maximise $(X|Y)$ dans $\mathbb{M}^D_k(\Omega)$.

Il y a deux possibilités :

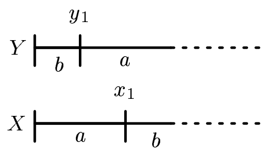
– si $\underline{\delta_0} \neq \underline{\gamma_0}$:

Les prédicteurs de X alternent, de même que ceux de Y, si bien que :

$$\forall j, \underline{\delta_{j+1}} = \underline{\gamma_j} \text{ et } \underline{\delta_{k+1}} \neq \underline{\gamma_{k-1}}$$

Le corollaire du lemme 4.2.2 interdit $x_1 = y_1$. Par conséquent, il y a deux possibilités :

– Si $y_1 < x_1$:



$$\begin{aligned} Y/\langle y_1,n\rangle &= \{(y_1,\underline{\gamma_1}),\ldots,(y_{k-1},\underline{\gamma_{k-1}})\} \\ &\in \mathbb{M}^D_{k-1}(\omega_{\langle y_1+1,n\rangle}) \end{aligned}$$

et

$$\begin{aligned} X/\langle y_1,n\rangle &= \{(y_1,\underline{\delta_0}),(x_1,\underline{\delta_1}),\ldots,(x_{k+1},\underline{\delta_{k+1}})\} \\ &\in \mathbb{M}^D_{k+2}(\omega_{\langle y_1+1,n\rangle}) \end{aligned}$$

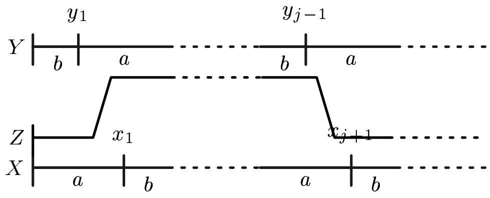
X a $k+2$ classes et Y $k-1$ classes sur $\omega_{\langle y_1+1,n\rangle}$, si bien qu'il existe $j > 0$ tel que

$\omega_{\langle x_j+1,x_{j+1}\rangle} \cap \omega_{\langle y_{j-1}+1,y_j\rangle} \neq \emptyset$.

On considère la partition prédictive, comme $\underline{\gamma_{j-1}} = \underline{\delta_j}$

$$Z = \{(x_0,\underline{\delta_0}),(y_2,\underline{\gamma_2}),\ldots,(y_{j-1},\underline{\gamma_{j-1}}),(x_{j+1},\underline{\delta_{j+1}}),\ldots,(x_{k+1},\underline{\delta_{k+1}})\}$$

Grâce au lemme 4.2.1, $Z \in \mathbb{M}^D_k(\Omega)$. Il est clair que $(X|Y) < (X|Z)$, si bien qu'il y a contradiction avec l'hypothèse sur Y.



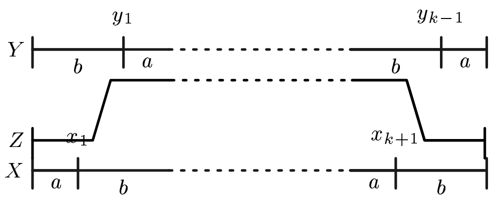
– Si $y_1 > x_1$:

Comme $\underline{\gamma_{k+1}} \neq \underline{\delta_{k-1}}$, par le corollaire ci-dessus, $x_{k+1} \neq y_{k-1}$. Deux cas :

– Si $x_{k+1} < y_{k-1}$, en considérant, comme $\underline{\gamma_{k-2}} = \underline{\delta_{k+1}}$:

$$Z = \{(x_0,\underline{\delta_0}),(x_1,\underline{\delta_1}),(y_1,\underline{\gamma_1}),\ldots,(y_{k-2},\underline{\gamma_{k-2}})\}$$

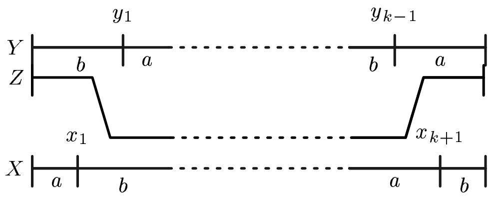
Grâce au lemme 4.2.1, $Z \in \mathbb{M}^D_k(\Omega)$. Il est clair que $(X|Y) < (X|Z)$, si bien qu'il y a contradiction avec l'hypothèse sur Y.



– Si $x_{k+1} > y_{k-1}$, en considérant :

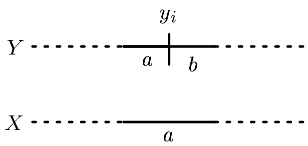
$$Z = \{(y_0,\underline{\gamma_0}),(x_2,\underline{\delta_2}),\ldots,(x_k,\underline{\delta_k})\}$$

$Z \in \mathbb{M}_k^D(\Omega)$ et on passe de Z à X par deux partages de période 1 qui se font aux extrémités de Ω (sur les parties $\omega_{\langle 1,x_1\rangle}$ et $\omega_{\langle x_{k+1}+1,n\rangle}$).

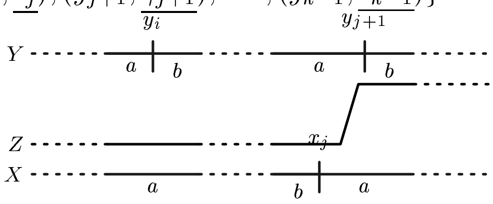


- Si $\delta_0 = \gamma_0$:
  Les prédicteurs de X alternent, de même que ceux de Y, si bien que :
  $$\forall j, \underline{\delta_j} = \underline{\gamma_j}$$
  Comme X et Y sont propres et n'ont pas le même nombre de classes, il y a forcément des positions où leurs fonctions de prédiction sont différentes. On regarde la plus petite telle position. La césure correspondante peut être sur X ou sur Y :

  - Si elle est sur Y, en $y_i < x_i$:
    $Y/\langle y_i, n\rangle \ = \{(y_i, \underline{\gamma_i}), \dots, (y_{k-1}, \underline{\gamma_{k-1}})\} \in \mathbb{M}_{k-i}^D(\omega_{\langle y_1+1,n\rangle})$
    et
    $X/\langle y_i, n\rangle \ = \{(y_i, \underline{\delta_{i-1}}), (x_i, \underline{\delta_i}), \dots, (x_{k+1}, \underline{\delta_{k+1}})\} \in \mathbb{M}_{k+3-i}^D(\omega_{\langle y_1+1,n\rangle})$

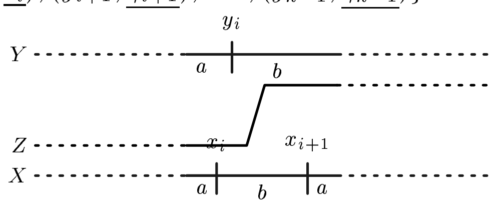
    

    X a $k-i+3$ classes et Y a $k-i$ classes sur $\omega_{\langle y_i+1,n\rangle}$, si bien qu'il existe $j > 0$ tel que :
    $\omega_{\langle x_j+1,x_{j+1}\rangle} \cap \omega_{\langle y_j+1,y_{j+1}\rangle} \neq \emptyset$.
    On considère la partition prédictive, comme $\underline{\delta_j} = \underline{\gamma_j}$,
    $$Z = \{(x_0, \underline{\delta_0}), \dots, (x_j, \underline{\delta_j}), (y_{j+1}, \underline{\gamma_{j+1}}), \dots, (y_{k-1}, \underline{\delta_{k-1}})\}$$
    Grâce au lemme 4.2.1, $Z \in \mathbb{M}_k^D(\Omega)$. Il est clair que $(X|Y) < (X|Z)$, si bien qu'il y a contradiction avec l'hypothèse sur Y.

  - Si la césure est sur X, en x_i avec $y_{i-1} < x_i < y_i$.
    Encore deux éventualités :

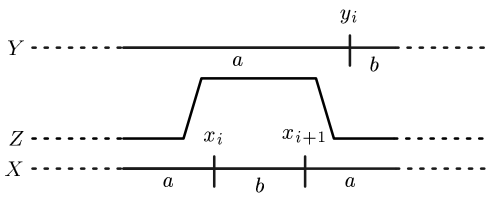
    - Si $y_i < x_{i+1}$:
      Soit :
      $$Z = \{(x_0, \underline{\delta_0}), \dots, (x_i, \underline{\delta_i}), (y_{i+1}, \underline{\gamma_{i+1}}), \dots, (y_{k-1}, \underline{\gamma_{k-1}})\}$$
      Grâce au lemme 4.2.1, $Z \in \mathbb{M}_k^D(\Omega)$. Il est clair que $(X|Y) < (X|Z)$, si bien qu'il y a contradiction avec l'hypothèse sur Y.

– Si $x_{i+1} < y_i$ et, en prenant :

$$Z = \{(x_0, \underline{\delta_0}), \ldots, (x_{i-1}, \underline{\delta_{i-1}}), (x_{i+2}, \underline{\delta_{i+2}}), \ldots, (x_{k+1}, \underline{\delta_{k+1}})\}$$

$Z \in \mathbb{M}_k^D(\Omega)$ et on passe de Z à X par une division ternaire, sur la partie $\omega_{\langle x_i+1, x_{i+1}\rangle}$.



□

Ainsi, les ensembles $\mathbb{M}_{2k}(\Omega)$ des $2k$-partitions pour les k successifs aussi bien croissants que décroissants, peuvent être considérés comme se déduisant simplement les uns des autres par division ternaire ou par couples de divisions binaires sur les extrémités de la séquence. Chaque $2k$-partition se déduit par de telles divisions « bien choisies » d'au moins une $2k-2$-partition, et chaque $2k-2$-partition permet de construire par ce moyen au moins une $2k$-partition. Les ensembles $\mathbb{M}_{2k+1}(\Omega)$ peuvent être liés de la même manière.

Pour préciser quelles divisions permettent de lier ces différents ensembles, il s'agit maintenant de savoir sur quelles parties de la séquence elles s'effectuent. Les divisions ternaires qui lient une k-partition aux $(k+2)$-partitions prédictives optimales doivent, par optimalité de celles-ci, se faire sur les parties des classes de la partition qui garantissent le plus de gain de prédiction par changement de prédicteur. Ceci correspond à la proposition suivante :

Proposition 4.2.3. *Si $|D| = 2$, soit $u_k = M^D_{2(k+1)}(\Omega) - M^D_{2k}(\Omega)$, la suite $(u_k)_k$ est décroissante (de même la suite $(M^D_{2k+1}(\Omega) - M^D_{2k-1}(\Omega))_k$ est décroissante).*

■ **Démonstration de la proposition 4.2.3** :

Soit $X = \{(x_i, \underline{\delta_i})\}_{i\in\langle 0,2k-1\rangle} \in \mathbb{M}^D_{2k}(\Omega)$.

Supposons qu'il existe $Y \in \mathbb{M}^D_{2k+2}(\Omega)$ qui soit issue de X par une division ternaire. On nomme les positions des deux césures créées y et y'.

Soit i tel que $x_i < y < y' < x_{i+1}$.

$$u_k = \pi(Y) - \pi(X) = \pi_{\underline{\delta_{i+1}}}(y, y') - \pi_{\underline{\delta_i}}(y, y')$$

Considérons $Z \in \mathbb{M}^D_{2k+4}(\Omega)$, qui soit issue de Y, par exemple par une division ternaire. Soient les positions des deux césures créées z et z' :

$$u_{k+1} = \pi(Z) - \pi(Y)$$

Deux cas peuvent se présenter :

– Si $y < z < z' < y'$:

$$u_{k+1} = \pi_{\underline{\delta_i}}(z, z') - \pi_{\underline{\delta_{i+1}}}(z, z')$$

Comme

$$u_k = \pi_{\underline{\delta_{i+1}}}(y, z') - \pi_{\underline{\delta_i}}(y, z') + \pi_{\underline{\delta_{i+1}}}(z', y') - \pi_{\underline{\delta_i}}(z', y')$$

on a

$$\begin{aligned} u_k - u_{k+1} &= \pi_{\underline{\delta_{i+1}}}(y,z') - \pi_{\underline{\delta_i}}(y,z') + \pi_{\underline{\delta_{i+1}}}(z',y') - \pi_{\underline{\delta_i}}(z',y') \\ &\qquad + \pi_{\underline{\delta_{i+1}}}(z,z') - \pi_{\underline{\delta_i}}(z,z') \\ &= \pi_{\underline{\delta_{i+1}}}(y,z') - \pi_{\underline{\delta_i}}(y,z') + \pi_{\underline{\delta_{i+1}}}(z,y') - \pi_{\underline{\delta_i}}(z,y') \end{aligned}$$

En considérant la partition

$W = \{(x_0,\underline{\delta_0}),\dots,(x_i,\underline{\delta_i}),(z',\underline{\delta_{i+1}}),$
$(y',\underline{\delta_i}),(x_{i+1},\underline{\delta_{i+1}}),\dots,(x_{2k-1},\underline{\delta_{2k-1}})\}$

qui a $2k+2$ classes,

$$\pi(Y) - \pi(W) = \pi_{\underline{\delta_{i+1}}}(y,z') - \pi_{\underline{\delta_i}}(y,z')$$

Comme $Y \in \mathbb{M}^D_{2k+2}(\Omega)$, on a forcément : $\pi_{\underline{\delta_{i+1}}}(y,z') - \pi_{\underline{\delta_i}}(y,z') \geqslant 0$

De la même manière, en considérant la $(2k+2)$-partition

$$\{(x_0,\underline{\delta_0}),\dots,(x_i,\underline{\delta_i}),(y,\underline{\delta_{i+1}}),(z,\underline{\delta_i}),(x_{i+1},\underline{\delta_{i+1}}),\dots,(x_{2k-1},\underline{\delta_{2k-1}})\}$$

on obtient : $\pi_{\underline{\delta_{i+1}}}(z,y') - \pi_{\underline{\delta_i}}(z,y') \geqslant 0$

si bien que

$$u_k - u_{k+1} \geqslant 0$$

– Si, par exemple $z < z' < y < y'$:

En considérant la $(2k+2)$-partition W créée par la division ternaire sur X par les césures z et z', on a $\pi(W) - \pi(X) = u_{k+1}$.
Comme $\pi(Y) - \pi(X) = u_k$, $\pi(Y) - \pi(W) = u_k - u_{k+1}$.
Or, $Y \in \mathbb{M}^D_{2k+2}(\Omega)$, si bien que :

$$u_k - u_{k+1} \geqslant 0$$

Le raisonnement est identique dans le cas où le passage de Y à Z se fait par deux divisions binaires aux extrémités de la séquence, ainsi que dans le cas où le passage de X à Y se fait par deux divisions binaires aux extrémités de la séquence.

□

En conséquence de cela, comme le gain de prédiction ne se fait, grâce à la proposition 4.2.1, que sur les classes créées au sein de classes déjà existantes, ce sont forcément les classes qui garantissent les gains les plus forts qui sont d'abord construites. Ainsi, il est immédiat de trouver toutes les $(k+2)$-partitions optimales qui peuvent être déduites d'une k-partition optimale. Mais pour obtenir toutes les $(k+2)$-partitions optimales, il faudrait considérer les transformations réalisables à partir de l'ensemble des k-partitions optimales. La proposition suivante permet d'éviter cela, en majorant, aux moments stables, le nombre de classes d'équivalence des partitions optimales (voir la section 2.3.1).

Proposition 4.2.4. *Si* $|D| = 2$, *soit* $u_k = M^D_{2(k+1)}(\Omega) - M^D_{2k}(\Omega)$.
Si $u_{k+1} < u_k$, *alors* $|\mathbb{M}^D_{2k}(\Omega)_{/\sim}| \leqslant 2$. *Le résultat est similaire pour la suite* $(M^D_{2k+1}(\Omega) - M^D_{2k-1}(\Omega))_k$.

Tout d'abord, un petit résultat, dans lequel on minimise la différence entre les deux prédictions sur une classe d'une partition optimale :

Lemme 4.2.3. *Si $|D| = 2$, soit $X = \{(x_i, \underline{\delta_i})\}_{i\in\langle 0,k+1\rangle} \in \mathbb{M}^D_{k+2}(\Omega)$ une partition prédictive propre.*
Soit $u = M^D_{k+2}(\Omega) - M^D_k(\Omega)$. On a
$\forall i \in \langle 1, k\rangle, \pi_{\underline{\delta_i}}(x_i, x_{i+1}) - \pi_{\underline{\delta_{i+1}}}(x_i, x_{i+1}) \geqslant u$
$\pi_{\underline{\delta_0}}(x_0, x_1) - \pi_{\underline{\delta_1}}(x_0, x_1) + \pi_{\underline{\delta_{k+1}}}(x_{k+1}, n) - \pi_{\underline{\delta_k}}(x_{k+1}, n) \geqslant u.$

■ **Démonstration du lemme 4.2.3**

Soit $i \in \langle 1, k\rangle$.

Si on enlève la classe prédictive $(\omega_{\langle x_i+1,x_{i+1}\rangle}, \underline{\delta_i})$ à X, on obtient une k-partition Z. De plus, puisque le segment $\omega_{\langle x_i+1,x_{i+1}\rangle}$ est prédit par $\underline{\delta_{i+1}}$ dans Z :

$\pi(X) - \pi(Z) = \pi_{\underline{\delta_i}}(x_i, x_{i+1}) - \pi_{\underline{\delta_{i+1}}}(x_i, x_{i+1})$

Or, $\pi(Z) \leqslant M^D_k(\Omega)$, si bien que : $\pi(X) - \pi(Z) \geqslant u$, et la conclusion suit.

On peut faire le même raisonnement avec l'union des deux classes extrêmes de X. □

■ **Démonstration de la proposition 4.2.4** :

Soit $X = \{(x_i, \underline{\delta_i})\}_{i\in\langle 0,2k+1\rangle} \in \mathbb{M}^D_{2(k+1)}$ justifiée à gauche dans sa classe.

Soit $Y = \{(y_i, \underline{\gamma_i})\}_{i\in\langle 0,2k+1\rangle} \in \mathbb{M}^D_{2(k+1)}$ tel que $\gamma_0 = \delta_0$, également justifiée à gauche dans sa classe.

Supposons, par exemple, que $x_1 < y_1$.

Comme, grâce au lemme 4.2.1, $x_2 \neq y_1$, deux cas se présentent :

- Si $x_2 < y_1$:
  Comme $\underline{\delta_1} \neq \underline{\gamma_0}$, en opérant une division ternaire dans Y par l'insertion de la classe $\omega_{\langle x_1+1,x_2\rangle}$, on obtient une $2(k+2)$-partition.
  Comme, grâce au lemme 4.2.3,
  $\pi_{\underline{\delta_1}}(x_1, x_2) - \pi_{\underline{\gamma_0}}(x_1, x_2) \geqslant u_k$
  on a
  $$M^D_{2(k+2)} - M^D_{2(k+1)} \geqslant u_k$$
  ce qui contredit $u_{k+1} < u_k$.
- Si $x_2 > y_1$:
  Par optimalité de X et Y, on a forcément :
  $\pi_{\underline{\delta_1}}(x_1, y_1) = \pi_{\underline{\gamma_0}}(x_1, y_1)$.
  Comme $\underline{\delta_1} = \underline{\gamma_1}$, la partition
  $$\{(y_0, \underline{\gamma_0}), (x_1, \underline{\gamma_1}), (y_2, \underline{\gamma_2}), \dots , (y_{2k+1}, \underline{\gamma_{2k+1}})\}$$
  est équivalente à Y, mais plus justifiée à gauche, ce qui contredit la définition de Y.

Par conséquent, $x_1 = y_1$.

En faisant successivement le même raisonnement successivement pour toutes les césures de Y, on obtient $X = Y$.

Par conséquent, il n'y a qu'une classe d'équivalence dans $\mathbb{M}^D_{2(k+1)}(\Omega)$ dont le prédicteur de la classe de ω_1 est $\underline{\delta_0}$.

Comme $|D| = 2$, on a le résultat voulu. □

Ce résultat confirme, en association avec la proposition 4.2.3, dans le cas très particulier où il n'y a que deux prédicteurs admissibles, l'existence de paliers de régularités (ici de période deux) dans les gains de prédiction, et la corrélation entre les moments stables et les ruptures dans ces paliers.

D'un point de vue pratique, on sait alors que, dans le cas où il n'y a que deux prédicteurs admissibles, il suffit, lorsque

$$M_k^D(\Omega) - M_{k-2}^D(\Omega) > M_{k+2}^D(\Omega) - M_k^D(\Omega)$$

de connaître deux k-partitions optimales dont les premiers prédicteurs sont différents pour, au moyen de l'algorithme de reconstruction des individus d'une classe d'équivalence, retrouver toutes les k-partitions optimales. Si, pour un k donné, toutes les partitions prédictives optimales commencent par le même prédicteur, il n'y a qu'une seule classe d'équivalence. Dans ce cas, avec l'algorithme de partitionnement qui ne génère qu'une seule k-partition optimale et par la reconstruction des classes d'équivalence, il est possible de trouver toutes les k-partitions optimales. En pratique, cette situation est extrêmement courante.

En fait, ces résultats permettent de générer directement les classes d'équivalence qui sont présentes aux moments stables, sans avoir à construire les partitions qui sont entre ces moments. Pour cela, il suffit, à partir d'une partition, de chercher en un seul parcours de la séquence les segments entre les césures déjà créées où le gain de changement de prédicteur est maximal, et de stocker toutes ces nouvelles césures avec les anciennes pour obtenir une partition optimale qui corresponde directement à un moment stable.

Dans un cas particulier, nous avons justifié les comportements observés dans les exemples. Cela nous conforte dans notre étude expérimentale faite au début de ce chapitre. En particulier, en ce qui concerne la définition des moments stables, et la corrélation de ces moments avec des variations de palier de l'incrément de prédiction.

4.3 Retour sur le critère d'arrêt

À la fin du premier chapitre (page 28 et suivantes), nous avons présenté différents critères qui peuvent servir à déterminer le nombre de classes dans le cadre du partitionnement maximalement prédictif sous contrainte d'ordre total. Maintenant que nous avons quelques informations sur le comportement du partitionnement, nous revenons sur plusieurs de ces critères.

4.3.1 Critères d'ambiguïté

Certains critères d'arrêt prennent en compte l'ambiguïté, à chaque niveau, entre les différentes partitions ex æquo. Ces critères considèrent soit le nombre d'arêtes des graphes des k-partitions, soit l'incertitude de la prédiction des objets. Les observations de ce chapitre ont révélé que ces critères présentent des minima locaux très nets en plusieurs niveaux du partitionnement que nous avons appelés moments stables. Mais ces moments se présentent tout au long du partitionnement, à chaque fois qu'un palier de l'incrément de prédiction est terminé.

De plus, l'ensemble des phénomènes qui se produisent au sein de chaque palier cherche justement à résoudre ces ambiguïtés. Ainsi, si ces critères d'ambiguïté ont permis de révéler ces moments stables, ce qui est très positif, ils ne semblent pas pouvoir permettre de choisir parmi ceux-ci.

En fait, on peut considérer, par l'existence des moments stables, qu'en général une séquence ne se partage pas seulement en un nombre fixé de classes, mais plutôt en un ensemble de nombres de classes, tous candidats à déterminer l'arrêt du partitionnement, et à « bien » partager la séquence. Si une précision de prédiction particulière est souhaitée, le plus petit moment stable pour lequel ce seuil est satisfait est un bon candidat comme moment d'arrêt. Cependant, si on ne veut pas avoir à fixer ce seuil, il s'agit d'évaluer si certains moments stables ne sont tout de même pas plus pertinents que d'autres, pour être à même de choisir parmi ceux-ci celui (ou ceux) qui résume au mieux la structure éventuelle de la séquence.

4.3.2 Coude de prédiction

La présence d'un coude dans la courbe de la prédiction devrait pouvoir révéler un bon critère d'arrêt (voir page 33), puisque ce coude correspond à un moment précis où le gain moyen de prédiction entre niveaux consécutifs devient brutalement faible. D'après les observations de ce chapitre, ce moment devrait être celui où les phénomènes qui apparaissent permettent subitement des gains moins importants. Sur chaque palier, les phénomènes permettent un incrément de prédiction uniforme en moyenne, et il n'y a pas à considérer de coude sur ces parties. Ceci revient à considérer que la courbe de prédiction est affine sur ces paliers. La recherche du coude se fera entre ces parties affines de la fonction de prédiction. Pour cela, on peut donc considérer que sur chaque palier la pente de la fonction de prédiction est la moyenne du gain d'incrément. C'est la comparaison des angles entre ces pentes successives qui permettra d'observer un éventuel coude (figure 4.15).

Comme ces angles dépendent de l'échelle qui a été utilisée pour représenter la prédiction en fonction du nombre de classes, il faut normaliser ces échelles entre les différents partitionnements, pour disposer de valeurs comparables entre partitionnements.

Imaginons que l'on opère deux partitionnements, un avec un lexique `L`, et un autre avec un lexique `L'` qui a les mêmes prédicteurs que `L`, mais tous avec un poids k fois plus important que pour `L`. Les deux partitionnements sont identiques en tous points, à ceci près que toutes les prédictions du second seront k fois plus fortes que celles du premier. Ainsi, les valeurs des angles pour les coudes doivent être semblables entre ces deux partitionnements. Pour neutraliser un quelconque facteur de prédiction uniformément réparti sur les prédicteurs, il convient d'établir un rapport d'échelle entre la prédiction et le nombre de classes. Par exemple, on peut fixer que, sur le dessin, un niveau de partitionnement a une taille proportionnelle à la prédiction en une classe de la séquence, M_1^D.

Les pentes successives de la courbe de prédiction diminuent vers zéro, si bien que pour exploiter la plus grande gamme d'angles possible entre 0 et $\frac{\pi}{2}$ radian, et que les angles ne soient pas trop faibles en raison de « l'écrasement » de

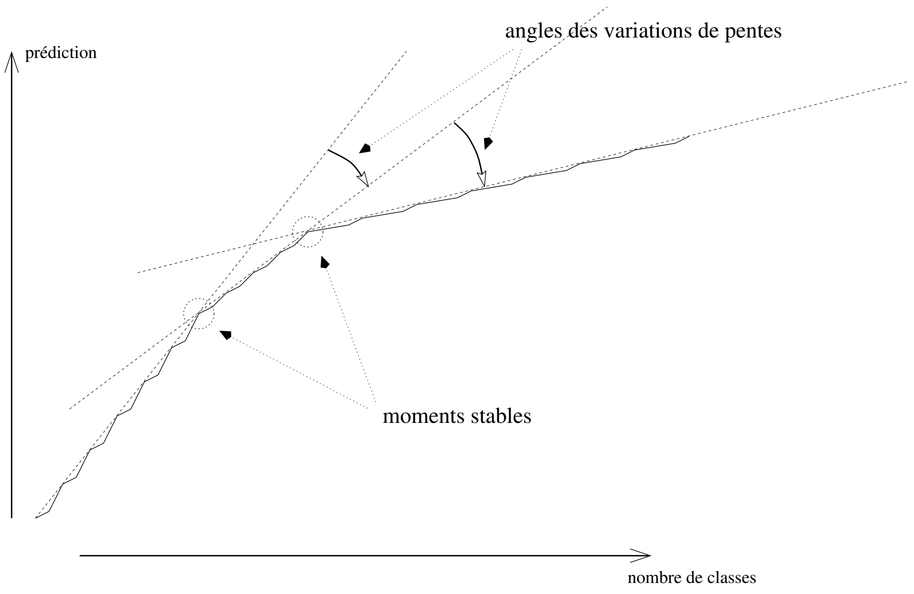


FIG. 4.15 – Angles des variations de pentes aux moments stables, à partir de la fonction de prédiction.

la courbe sur l'axe des abscisses on prend comme facteur de proportion 1/100. Dans ce cas, l'angle formé entre la verticale et le segment qui relie l'origine et le point qui correspond à la prédiction en une classe (coordonnées $(1, M_1^D)$) est égal à $\arctan(1/100)$, soit à peu près 10^{-2} radian (0.57 degré).

On considère un moment stable donné. L'accroissement moyen de prédiction par niveau de partitionnement sur le palier qui est avant ce moment est x. y est l'accroissement moyen de prédiction par niveau de partitionnement sur le palier qui est après ce moment. L'angle $\alpha(x, y)$ entre les pentes en ce moment est :

$$\alpha(x,y) = \arcsin\left(\frac{(x-y).\frac{M_1^D}{100}}{\sqrt{x^2+\left(\frac{M_1^D}{100}\right)^2}.\sqrt{y^2+\left(\frac{M_1^D}{100}\right)^2}}\right)$$

Considérer la prédiction en une classe de la séquence comme valeur étalon permet aussi que l'angle puisse être calculé au fur et à mesure du partitionnement dès que celui-ci commence. Si un seuil de critère d'arrêt sur les angles est fixé pour déterminer un coude, le partitionnement peut alors être arrêté automatiquement juste quand il faut, sans avoir à le poursuivre.

En outre, après le dernier moment stable, le partitionnement est abouti, donc la pente de l'incrément est nulle. Comme on considère des valeurs d'angles positives et éventuellement négatives, on est sûr que la somme de tous les angles entre le premier et le dernier moment stable est toujours égale à $(\frac{\pi}{2} - 10^{-2})$.

Si on considère un niveau du partitionnement, la somme des angles aux moments stables entre ce niveau et le partitionnement abouti (pente nulle) est l'angle entre la pente de l'incrément de prédiction en ce niveau et la pente nulle. Si le gain moyen en ce niveau est m, la somme des angles entre le premier moment stable et celui qui est juste avant le niveau où la pente est m est

$$\frac{\pi}{2} - 10^2 - \arcsin\left(\frac{1}{\sqrt{\left(\frac{M_1^D}{100.m}\right)^2 + 1}}\right)$$

Évaluer cette pente m est facile lorsque les classes sont plus petites, car les phénomènes qui apparaissent dans le partitionnement sont prévisibles. Par exemple, la pente minimale non nulle (donc juste avant que le partitionnement soit terminé) correspond à une division ternaire qui ne permet d'améliorer la prédiction que de la valeur de la prédiction d'un objet. Le gain de prédiction de cette division est seulement la prédiction d'un seul objet, et ce gain se fait sur deux niveaux de partitionnement. Si la plus petite prédiction d'un objet dans la séquence est p, l'angle entre cette pente et la pente nulle du partitionnement abouti est alors au moins :

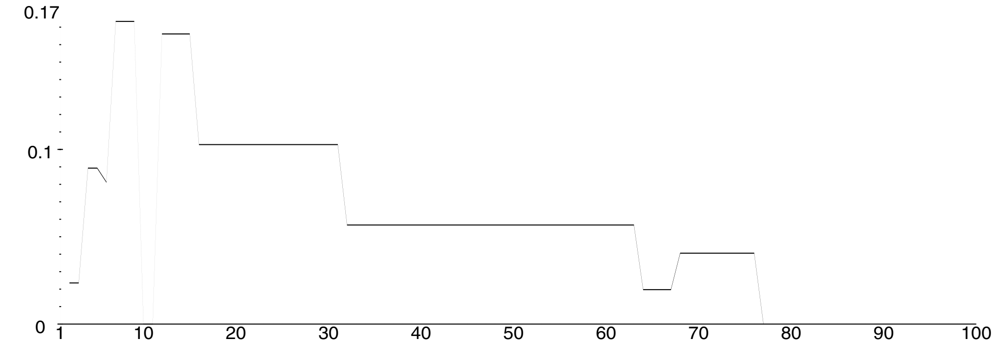
$$\alpha(p/2, 0) = \arcsin\left(\frac{1}{\sqrt{\left(\frac{M_1^D}{50.p}\right)^2 + 1}}\right)$$

On voit parfaitement bien, pour estimer un moment d'arrêt, l'intérêt de la considération des angles des variations de pentes par rapport à la variation de la prédiction. La seule connaissance des pentes de gain de prédiction – et donc la seule connaissance du mécanisme du partitionnement – permet d'estimer la situation d'un niveau du partitionnement par rapport à la partition complète. En considérant l'incrément de prédiction comme critère, une telle estimation nécessite, en plus de la connaissance des mécanismes et donc des gains moyens du partitionnement, celle la longueur des paliers où se manifestent ces mécanismes. Cette longueur peut être très difficile à évaluer.

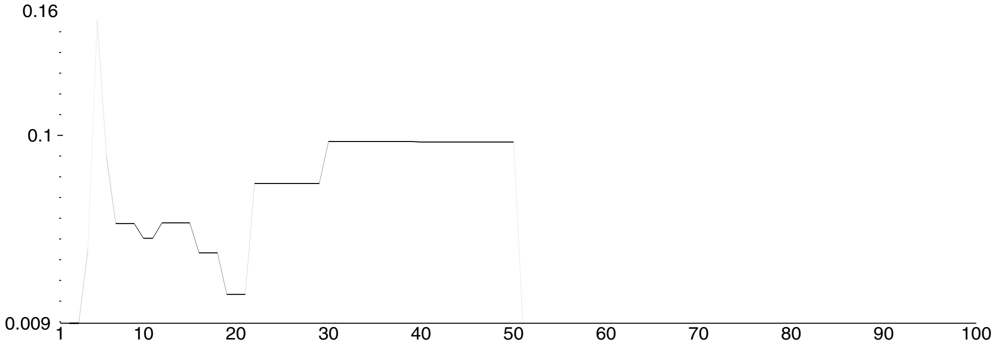
On appellera courbe de la fonction de coude la fonction qui aux moments stables affiche la valeur de la différence entre les angles des pentes d'incrément avant et après ces moments, et qui est fixée constante entre moments stables pour faciliter la visualisation. La valeur constante sur un palier est la valeur de la variation d'angle au moment stable qui *termine* ce palier.

Pour les trois partitionnements sur des séquences aléatoires donnés en exemple au début de ce chapitre (voir figures 4.5, 4.6, 4.7), les courbes de fonctions de coudes sont tracées en figure 4.16.

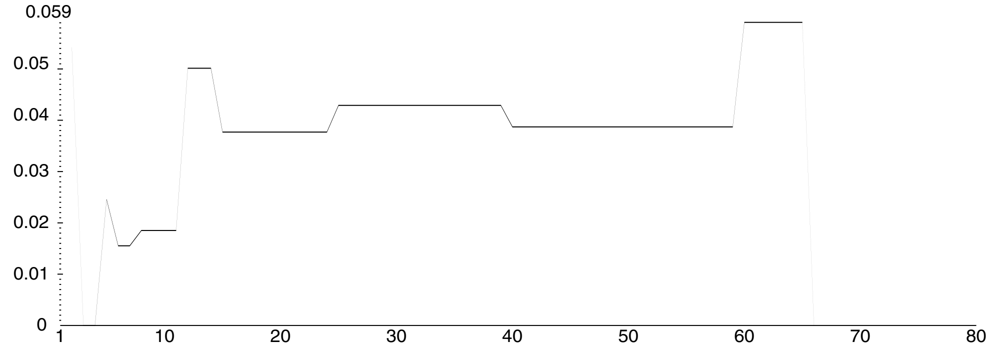
On peut remarquer sur ces figures que les angles sont tous très faibles (au maximum 0.17 radian, c'est-à-dire moins de 10 degrés), et que le maximum n'est pas remarquablement plus important que les autres valeurs. Pour les partitionnements des courbes 4.16(a) et 4.16(b), comme la prédiction en une classe est à peu près 300, l'angle entre la pente du gain d'une prédiction par niveau et la pente nulle est environ 0.28 radian. Par conséquent, la somme restante des



(a) `L = *`



(b) `L = **`



(c) `L = "**" "***"(2)`

Fig. 4.16 – Courbes de la fonction de coudes pour trois partitionnements maximalement prédictifs (voir figures 4.5, 4.6, 4.7). L'axe des abscisses représente les niveaux des partitionnements, et en ordonnée sont indiquées les différences d'angles entre moments stables successifs. Par indétermination de l'angle aux moments non représentés dans le graphique, cette valeur est arbitrairement fixée à zéro.

angles entre ces moments est à peu près 1.26 radian. Au maximum, la part que prennent les angles observés de cette proportion est 14 %. Dans le cas de la courbe 4.16(c), la prédiction en une classe est 106, si bien que l'angle calculé ci-dessus qui correspond au gain d'une prédiction par niveau est 0.8 radian, et la somme des angles restants est 0.76 radian. Le maximum d'angle observé est 0.06 radian, soit 8 % de cette somme. On peut décider que dans ces trois cas, cela est trop faible pour décréter la présence d'un coude.

Pour un coude remarquable, on pourrait s'attendre à ce que ce l'angle soit remarquablement plus fort. Par exemple, si on reprend les exemples de la figure 1.6, page 34, les fonctions liées aux angles dans la figure 4.17 font apparaître des maxima très nets. Sur la première courbe, il y a en fait deux angles importants et comparables, qui totalisent plus de 1.1 radian (63 degrés). Sur la seconde courbe, un seul maximum atteint 0.9 radian (52 degrés). Ces angles représentent plus de la moitié de la somme de tous les angles, ce qui est considérable.

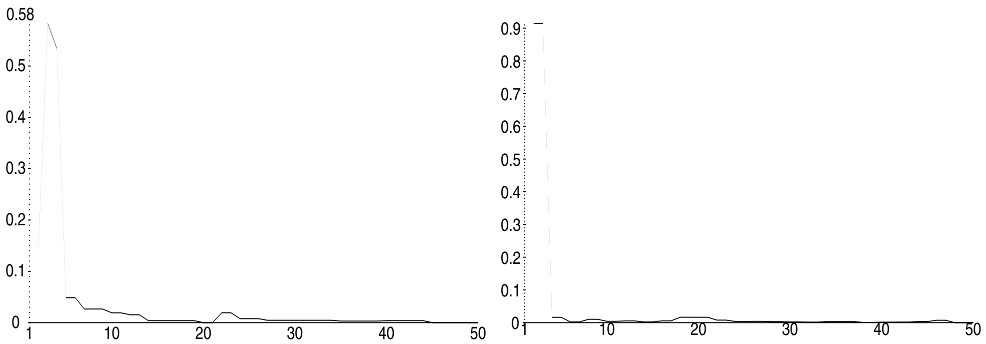


FIG. 4.17 – Deux fonctions de coude pour les fonctions de prédiction de la figure 1.6.

Considérer la valeur par rapport à $\frac{\pi}{2}$ d'un angle pour déterminer la présence d'un coude est certes un critère tout-à-fait convenable dans le sens où, si cette valeur est grande, on peut dire qu'alors il y a un coude. En revanche, dans certains cas, qu'un maximum soit de valeur moyenne ne suffit pas à déterminer qu'il n'y a pas de coude significatif. C'est un peu le cas sur la courbe de gauche de la figure 4.17. Si les deux angles importants étaient séparés, il aurait fallu les considérer tous deux comme coudes significatifs. Or, comme la somme de leurs valeurs est inférieure à $\pi/2$, elles ne pouvaient pas être aussi importantes que celle de la courbe de droite. Un seuil peut être utilisé pour déterminer l'existence ou non d'*un* seul nombre de classes. Mais s'il y a plusieurs nombres de classes qui se démarquent, il ne pourra pas les détecter.

Si deux valeurs d'angles aux moments stables sont proches, mais nettement supérieures aux autres valeurs d'angles, étant donnée que les angles ont tendance à diminuer à fur et à mesure du partitionnement, on peut estimer que celui de ces deux angles qui correspond au niveau du partitionnement le plus élevé a plus d'intérêt que l'autre. En outre, le fait qu'une partie de l'intervalle $[0, \frac{\pi}{2}]$ ait déjà été « utilisée » par l'angle du plus petit niveau des deux valorise en comparaison l'angle du plus grand niveau.

Considérer ce calcul d'angle pour déterminer les éventuels coudes d'une courbe de partitionnement, et donc d'éventuels moments d'arrêt valables, présente de nombreux avantages. Sa facilité d'interprétation est tout de même satisfaisante dans un contexte expérimental. Le fait qu'il varie dans un intervalle déterminé $[0, \pi/2]$ et que la somme des angles sur tout un partitionnement soit égale à $\pi/2 - 10^{-2}$ permet de comparer les angles obtenus entre plusieurs partitionnements. La simple observation (ou prévision) des phénomènes liés à la fin du partitionnement permet de réduire cet intervalle, et à l'occasion de se fixer un critère de validation de coude en fonction de ces calculs. En outre, ce critère peut être évalué presque en même temps que le partitionnement est effectué, et il est extrêmement rapide à calculer.

Ainsi, on a conçu un moyen d'évaluation des moments stables qui permet de les comparer, et d'évaluer si certains sont particulièrement intéressants, auxquel cas ils seront considérés comme candidats valables au statut de moment d'arrêt. Si aucun angle ne se démarque particulièrement, aucun niveau n'est particulièrement valable pour être moment d'arrêt, et on peut donc dire que, selon ce partitionnement, la séquence étudiée ne peut pas être résumée de façon satisafisante en plusieurs classes.

Ce chapitre a été très instructif en ce qui concerne le partitionnement maximalement prédictif. En effet, tout d'abord, nous avons remarqué la grande stabilité des césures des partitions aux niveaux successifs du partitionnement. Nous avons expliqué cette stabilité par des phénomènes apparents – combinaisons de divisions binaires et ternaires – qui donnent l'impression d'un comportement presque hiérarchique du partitionnement. Dans le cas particulier où il n'y a que deux prédicteurs admissibles, nous avons montré que ce comportement existe sur l'ensemble des partitions ex æquo des partitionnements, entre niveaux de même parité. Nous avons corroboré dans ce cas les observations qui révèlent des niveaux très particuliers du partitionnement, niveaux où le nombre de partitions ex æquo est très faible. Nous avons appelé ces niveaux particuliers moments stables. Entre ces moments, la fonction d'incrément de prédiction est régulière (si elle en a le temps), et les moments stables sont en corrélation avec des variations de régularité dans la courbe des incréments de prédictions. Ainsi, la seule considération de la variation des prédictions permet de sélectionner les niveaux où il est convenable d'extraire des partitions du partitionnement. Enfin, grâce à ces observation, nous avons approfondi un critère d'arrêt qui, lui aussi, se calcule sur la seule donnée de la fonction de prédiction.

Après les travaux de ce chapitre et des chapitres précédents, nous pouvons nous considérer en mesure de pratiquer des expérimentations sur les données qui nous intéressent plus particulièrement, les séquences génétiques.

Chapitre 5

Expérimentations

« Je suis aujourd'hui convaincu, comme toi, qu'il n'y a pour nous qu'un seul objet d'étude : la Vie. »
G.A. d'ANNUNZIO
L'enfant de volupté

5.1 Rappels biologiques

Les mécanismes biologiques qui assurent la maintenance et la pérennité de tout organisme vivant sont en grande partie déterminés par une information génétique codée par la composition d'une molécule. Cette molécule est l'acide désoxyribonucléique – ou ADN. Elle est constituée d'une longue chaîne – parfois un anneau – de bases. Il y a quatre types de bases : deux pyrimidines, thymine et cytosine, et deux purines, adénine et guanine. Ces bases sont de façon standard notés par les lettres T, C, A, G respectivement. Une base est la partie fonctionnelle d'une molécule plus grande, le nucléotide. Une séquence d'ADN a une orientation, le sens direct étant nommé $5' - 3'$, et le sens inverse $3' - 5'$. L'ADN est une double hélice [WC53] formée de deux brins tête-bêche, symétriques puisque liés base à base par des liaisons A–T [1] et C–G, si bien que sur l'ensemble de la double hélice, il y a autant de A que de T, et autant de C que de G (première loi de CHARGAFF).

Les processus biologiques qui concernent l'ADN suivent des mécanismes dont beaucoup encore inconnus, mais d'autres sont déjà raisonnablement bien déterminés. Ainsi, la **transcription**, par laquelle l'information génétique est extraite de parties de l'ADN pour être copiée en un acide ribonucléique – ou ARN. La transcription est faite selon un mécanisme qui suit le sens direct du brin transcrit. Cette étape fait partie du processus qui concerne la création des protéines à partir de l'information génétique. Certains ARN ainsi formés restent tels quels dans la cellule, d'autres servent à coder les protéines. Une fois transcrit, l'ARN peut être modifié, ce qui donne un ARN mature. Les processus qui transforment

[1] Une adénine sur un brin est liée à une thymine sur l'autre brin, et inversement.

l'ARN à ce moment peuvent être extrêmement variables. En particulier, dans les organismes eucaryotes[2] le pre-ARN est excisé, c'est-à-dire que des parties lui sont enlevées. Ces parties excisées sont appelées introns. Les autres parties – les exons – sont préservées[3] et serviront effectivement à construire les protéines. Le processus de création d'une protéine à partir d'un ARN est appelé **traduction**. La traduction se fait aussi dans le sens qui correspondait au sens direct du brin d'ADN. Une protéine est créée sur une partie contiguë de l'ARN mature, et il peut y avoir plusieurs protéines formées à partir d'un seul ARN. Les parties de l'ADN qui après la transcription et après les transformations de l'ARN sont utilisées pour construire une protéine déterminent, sur l'ADN, un **gène**. Dans le cas des eucaryotes, un gène peut être séparé en plusieurs parties sur le brin d'ADN qui a servi à construire la protéine.

Une protéine est une molécule constituée d'un chapelet d'acides aminés. Les mécanismes de la traduction peuvent être résumés au fait que les bases de l'ARN sont « lues » par paquets de trois, et que chaque triplet est traduit en un acide aminé. Il existe en tout 20 acides aminés (table 5.1), et à chacun correspondent pour le coder plusieurs triplets de nucléotides, que l'on appelle codons (table 5.2). Il y a 61 codons, et chaque codon est traduit en exactement un acide aminé ; les trois triplets restant (sur les 64 possibles) ne codent rien et stoppent le processus de traduction (codons « stop »). Le code de passage des codons aux acides aminés est, à de très rares exceptions, universel dans le monde du vivant. Comme il existe plusieurs codons par acide aminé, ce code est redondant, mais on remarque que l'utilisation de cette redondance n'est pas du tout aléatoire au sein d'un ADN [GGG+81, BH82], ni entre différentes espèces [GGG80, Wri90]. L'utilisation particulière de cette redondance par une espèce est appelée **usage du code**. Par exemple, dans un génome, on peut observer que pour coder la proline (P), la proportion du codon CCA est 6 %, celle de CCC 18 %, celle de CCG 35 % et celle de CCT 41 %.

Alanine	A	Cystéine	C	Acide aspartique	D
Acide glutamique	E	Phénylalanine	F	Glycine	G
Histidine	H	Isoleucine	I	Lysine	K
Leucine	L	Méthionine	M	Asparagine	N
Proline	P	Glutamine	Q	Arginine	R
Sérine	S	Thréonine	T	Valine	V
Tryptophane	W	Tyrosine	Y		

Tab. 5.1 – Acides aminés et leurs symboles à une lettre

L'autre mécanisme important qui concerne l'ADN est la **réplication**, c'est-à-dire le processus qui permet à un ADN d'être recopié en un autre ADN identique lors de la division cellulaire (figure 5.1).

[2]Un eucaryote est un organisme dont les – ou la – cellules ont un noyau délimité par une enveloppe. L'ADN est dans ce noyau. Un organisme procaryote est constitué d'une cellule qui ne possède pas de noyau, et dans laquelle l'ADN « flotte ».

[3]Elles sont dites épissées

1re lettre	2^{e} lettre : T		C		A		G		3^{e} lettre
T	TTT	F	TCT	S	TAT	Y	TGT	C	T
T	TTC	F	TCC	S	TAC	Y	TGC	C	C
T	TTA	F	TCA	S	TAA	Stop	TGA	Stop	A
T	TTG	F	TCG	S	TAG	Stop	TGG	W	G
C	CTT	L	CCT	P	CAT	H	CGT	R	T
C	CTC	L	CCC	P	CAC	H	CGC	R	C
C	CTA	L	CCA	P	CAA	Q	CGA	R	A
C	CTG	L	CCG	P	CAG	Q	CGG	R	G
A	ATT	I	ACT	T	AAT	N	AGT	S	T
A	ATC	I	ACC	T	AAC	N	AGC	S	C
A	ATA	I	ACA	T	AAA	K	AGA	R	A
A	ATG	M	ACG	T	AAG	K	AGG	R	G
G	GTT	V	GCT	A	GAT	D	GGT	G	T
G	GTC	V	GCC	A	GAC	D	GGC	G	C
G	GTA	V	GCA	A	GAA	E	GGA	G	A
G	GTG	V	GCG	A	GAG	E	GGG	G	G

TAB. 5.2 – Code génétique universel

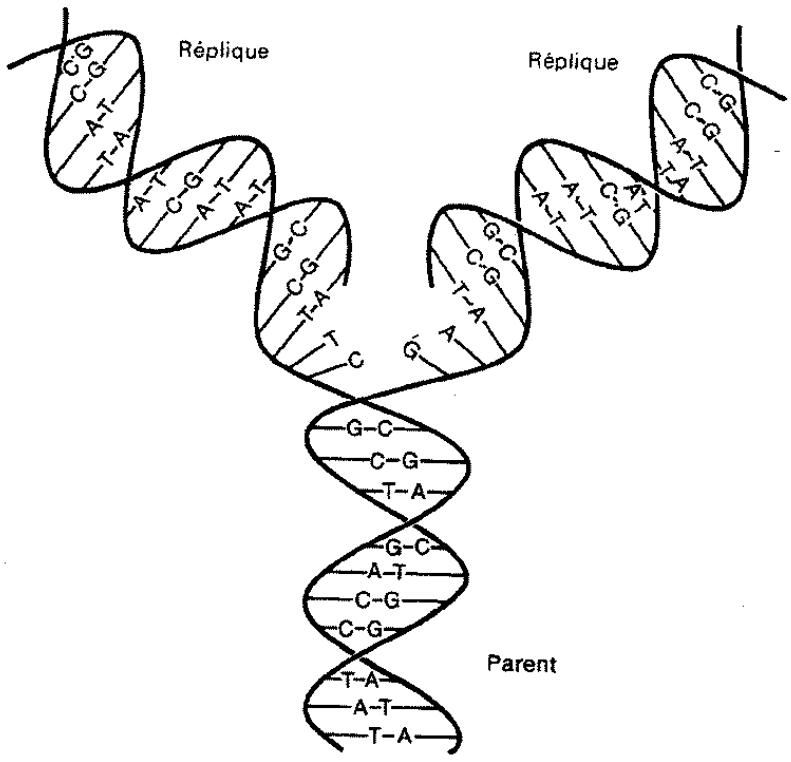


FIG. 5.1 – Mécanisme local de la réplication (de [SAG96])

La réplication consiste en une séparation progressive des deux hélices de la séquence génétique à partir d'une ou de plusieurs positions de la chaîne génomique. On nomme **origine de réplication** une position à laquelle la réplication peut être initiée. Comme la réplication passe par une séparation progressive des deux brins de la séquence, elle peut se faire dans un seul sens – réplication en rolling circle – ou, si l'origine ne se situe pas au bout de la séquence (ou si celle-ci est circulaire), en même temps dans les deux sens – réplication en théta. Comme les deux brins de l'ADN sont en sens inverse, sur toute partie de celui-ci, la progression de la séparation se fait dans le sens direct d'un brin et dans le sens indirect de l'autre brin. Les parties de brin sur lesquelles la réplication est faite dans le sens direct sont appelées parties leading et celles sur lesquelles elle se fait dans le sens inverse parties lagging (voir la figure 5.2).

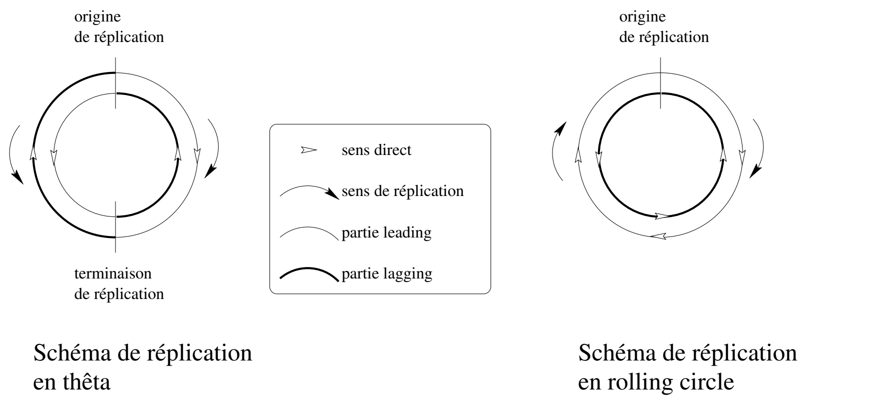


FIG. 5.2 – Zones leading et lagging sur des génomes circulaires. La réplication débute à une origine de réplication et peut suivre deux mécanismes de réplication. Les deux brins l'ADN sont représentés par les deux cercles concentriques.

Les mécanismes en action sur chaque partie, leading ou lagging, sont différents car localement, la molécule[4] qui assure la réplication ne peut lire un brin que dans le sens direct. Par conséquent, si la réplication peut être faite en continu sur la partie leading, elle ne peut être faite que de façon discontinue sur la partie lagging : localement, sur ce brin, la réplication est faite dans le sens opposé au sens de progression de la séparation des deux brins. La terminaison de réplication est l'endroit où les deux fourches de réplication se rejoignent à la fin de ce processus. À l'opposé de l'origine de réplication, qui occupe une position bien précise, la localisation de la terminaison dépend de la vitesse des mécanismes de réplication sur chaque coté, et varie alors plus ou moins.

L'ADN est un des éléments des organismes vivants qui se transmet de façon plus ou moins stable au cours du temps, si bien que la constitution d'un ADN est liée à l'histoire phylogénétique de son organisme. Ainsi, à partir de correspondances qui peuvent être établies entre différents organismes (par exemple sous la forme de classifications en eucaryotes, procaryotes, bactéries, archebactéries, etc), on peut prévoir – ou au moins espérer – des correspondances entre

[4] l'ARN polymérase

différentes parties d'ADN, ce qui peut aussi être utilisable dans une approche en systématique. Cette connaissance pourrait aussi permettre d'évaluer l'universalité ou la particularité de caractéristiques de composition. Plusieurs études sur une relation entre la phylogénie d'une espèce et les différentes proportions de mots de son génome sont présentées dans [KC94].

Devant l'accroissement gigantesque du nombre de séquences génétiques, et la taille de ces séquences, il apparaît important de pouvoir mettre à la disposition des biologistes des outils informatiques à même de leur faciliter et accélérer l'analyse de ces séquences, pour pouvoir gérer le fonctionnement, le rôle, et l'histoire de tout ou partie d'un ADN. L'approche informatique de l'analyse de séquences génétiques peut aussi être utile à mieux appréhender les mécanismes qui régissent le fonctionnement de la cellule. En effet, on est loin de connaître parfaitement des processus si complexes, et des méthodes informatiques et statistiques permettraient de mettre en relief des phénomènes, régularités ou anomalies, qui serviraient de catalyseurs de découvertes biologiques.

Il existe bien sûr de nombreuses voies de recherche en ce sens, de multiples manières d'exploiter les caractéristiques – génétiques, physico-chimiques, statistiques – qui sont progressivement mises en évidence dans les séquences génétiques, en particulier par des anomalies dans la composition en nucléotides de celles-ci [SWS83, Nus84, KM91, BCK92].

Ainsi, il existe de nombreux biais de proportions d'oligonucléotides – mots de nucléotides – entre les différentes régions des génomes. Dans [RVD98], une analyse du biais par rapport à un modèle markovien au sein des mots d'au plus huit nucléotides est effectuée sur quatre génomes bactériens, et en particulier sur celui de *Bacillus subtilis*. Un des buts de cette étude est d'observer les différences de proportion de différents mots sur plusieurs régions caractéristiques – déjà connues – du génome, comme les gènes, les zones intergéniques, les prophages, ou encore les zones de codage des ARN. Il apparaît que les proportions de lettres et de mots sont très variables entre ces zones, et que les mots sur-représentés et sous-représentés y sont différents. Il s'agit ensuite d'étudier s'il est possible, à l'inverse, d'utiliser ces différents biais pour l'analyse de génomes.

Dans beaucoup de méthodes d'analyse, un ADN est simplement vu comme un mot sur l'alphabet $\{A, C, G, T\}$. Cependant, il faut toujours garder en mémoire que ces lettres correspondent à des molécules entre lesquelles, par de multiples propriétés physico-chimiques, peuvent être établies des relations qui peuvent influencer une quelconque approche informatique du traitement de l'ADN [Sag96]. De plus, la connaissance des mécanismes biologiques peut servir à mieux concevoir ou appliquer des techniques informatiques.

5.2 Recherches de zones de séquences génétiques

Parmi les méthodes qui cherchent des zones « intéressantes » dans les séquences génétiques, il y a celles dont la recherche se base sur l'apparition de mots caractéristiques qui délimitent ces zones, et celles dont la recherche se base sur le contenu même de ces classes [KC94]. En général, les outils mis en œuvre pour cela utilisent ces deux approches de concert. Nous allons présenter quelques méthodes qui visent à partager les séquences génétiques en classes, ou au moins à en extraire des zones intéressantes, par l'analyse du contenu de ces zones, puisque le partitionnement maximalement prédictif, en tant que méthode classificatoire, cherche justement à opérer des partitions en fonction du contenu des classes formées. Ensuite, nous verrons quelques expérimentations du partitionnement maximalement prédictif.

5.2.1 Comparaison avec des séquences connues

Les banques de données de séquences génétiques et protéiques regorgent de plus en plus de parties de séquences annotées, c'est-à-dire d'informations sur la composition de nombreuses protéines et des régions géniques qui les codent.

L'approche la plus utilisée pour, sur une séquence génétique inconnue, trouver les parties codantes est d'utiliser l'information disponible sur ces banques et de comparer cette séquence aux portions de séquences déjà reconnues. Pour cela, des méthodes d'alignement telles que celles présentées dans la section 1.3.1 sont utilisées. Si une partie d'un génome a une forte ressemblance avec une partie identifiée comme codant une protéine, on espère que ces deux parties codent des protéines dont les fonctions sont apparentées [Sta90a].

Dans la logique de cette idée, on peut vouloir, à partir de plusieurs parties similaires dont la fonction est voisine, construire un consensus de ces parties. Ce consensus peut être apparenté à une séquence avec des trous et des ambiguïtés sur ses positions, mais de façon plus riche et plus complexe il peut prendre la forme d'expressions régulières. Une partie de séquence étudiée est déclarée faisant partie de la même famille si elle respecte l'expression régulière construite. Dans [BJEG98], différentes approches de la construction de consensus de séquences sont présentées et intégrées dans un programme d'apprentissage faisant appel à des méthodes d'intelligence artificielle. Cette approche permet non seulement de faciliter la reconnaissance des régions d'intérêt de génomes inconnus, mais aussi d'extraire, par la synthèse que fournit chaque consensus une connaissance qui puisse permettre d'expliquer la variation des séquences qui ont formé ce consensus.

Une autre utilisation de l'information disponible dans les banques de données génétiques est d'observer différentes proportions d'oligonucléotides sur plusieurs parties comparables de plusieurs séquences, et d'essayer d'identifier les parties de la séquence étudiée qui présentent les mêmes proportions. L'hypothèse de cette approche est que ces propriétés sont caractéristiques des parties où elles ont été observées. Par conséquent, il est important de prendre en compte l'hétérogénéité inter espèces [GGG80, BCK92, KC94] liée à la phylogénie.

Par exemple, Fickett [Fic82] observe la valeur d'un certain critère sur une base de données génétiques pour discriminer des régions codantes de régions non-codantes. Les critères qu'il utilise sont liés au fait que sur des régions codantes la probabilité que deux bases en phase de trois positions (base en position 1, en position 4, en position 7, etc) portent la même lettre est plus forte que sur des parties non-codantes. À partir de cette observation, des critères de discrimination sont établis. Ils sont ensuite testés sur une base de données afin d'être liés à une loi de la probabilité qu'une région est codante ou non. Enfin, sur une séquence inconnue, ces critères sont mesurés et, à l'aide des lois de probabilité construites, cette séquence est déclarée codante ou non-codante.

5.2.2 Modèles de composition

Une autre approche consiste à rechercher sur différentes séquences des variations similaires de comportements entre régions à discriminer, de tirer des lois de ces observations, et d'appliquer ces lois pour discriminer des régions dans de nouvelles séquences. La différence par rapport aux approches précédentes est que ces lois devraient permettre d'analyser une séquence par la *seule* donnée de sa composition, et non par des facteurs obtenus à l'aide d'autres séquences, ou par la comparaison directe avec des séquences. D'un certain point de vue, la recherche de séquences consensus peut se rapprocher de ce type d'étude, mais dans le cas présent, les lois sont liées à une composition répartie sur l'ensemble de chaque partie, et non seulement à des « points d'ancrage » qui permettent d'aligner une séquence et un modèle.

Par exemple, pour identifier les régions de séquences génomiques codant les protéines, Staden [Sta90b] observe des dépendances entre bases dans des zones codantes alors que ces dépendances sont absentes dans des zones non-codantes. Ainsi, il remarque qu'indépendamment de la phase de lecture, il y a de fortes variations de compositions entre les différentes bases : par une mesure des variations entre bases, il conçoit une méthode de discrimination de zones codantes et non-codantes. Un autre point de vue lui fait considérer les proportions attendues en bases dans les trois phases de codage en fonction des proportions en bases de la première position des codons et de la redondance du code de traduction. Ces valeurs attendues sont utilisées dans le but de déterminer la phase de lecture des zones codantes dans les séquences. Un troisième critère part de l'hypothèse que tous les gènes dans une séquence ont un usage du code similaire, et que cet usage est assez fort pour être utilisé comme moyen de discrimination. Dans ces trois cas, les différentes valeurs qui permettent de discriminer les positions de la séquence sont calculées sur une fenêtre glissante d'une taille fixe.

Même si elles ne visent pas précisément à un modèle de redescription de zones, ces méthodes utilisent neanmoins des lois sensées être caractéristiques des régions qu'elles veulent connaître. En poussant plus loin cette approche, on peut concevoir des méthodes qui cherchent à partager des séquences en zones décrites par des modèles plus ou moins approchés, déterministes ou stochastiques.

5.2.3 Chaînes de Markov à états cachés

Cette approche correspond à une volonté de modélisation stochastique de la séquence d'ADN selon le modèle présenté en section 1.3.2.3, dans lequel chaque état correpond à une chaîne de Markov. Tout le travail réside, une fois certains paramètres fixés, comme les caractéristiques sur lesquelles le travail s'opère, le nombre d'états cachés, la taille des processus markoviens,... à trouver les coefficients des matrices qui permettent de modéliser au mieux la séquence d'étude. Cette modélisation permet ensuite de partager la séquence en autant de parties que le modèle produit de segments sur lesquels la probabilité d'appartenance

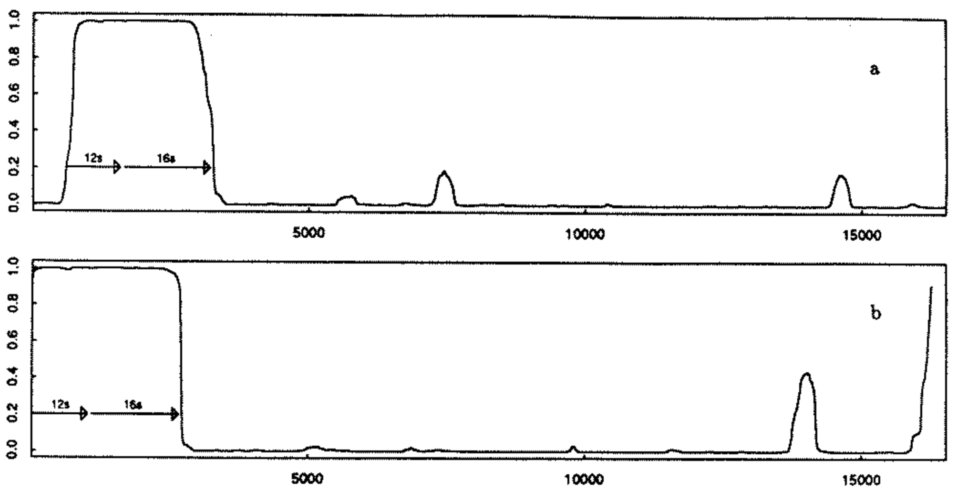


FIG. 5.3 – Courbes lissées du maximum de vraisemblance pour les deux états purine et pyrimidine, sur deux séquences d'ADN mitochondriaux : humain (a) et de souris (b). Sur l'axe vertical est indiquée la probabilité d'appartenance à l'état correspondant à un excès de purine. L'axe horizontal représente la séquence, et les flèches indiquent les zones codant l'ARN ribosomique(de [CHU89]).

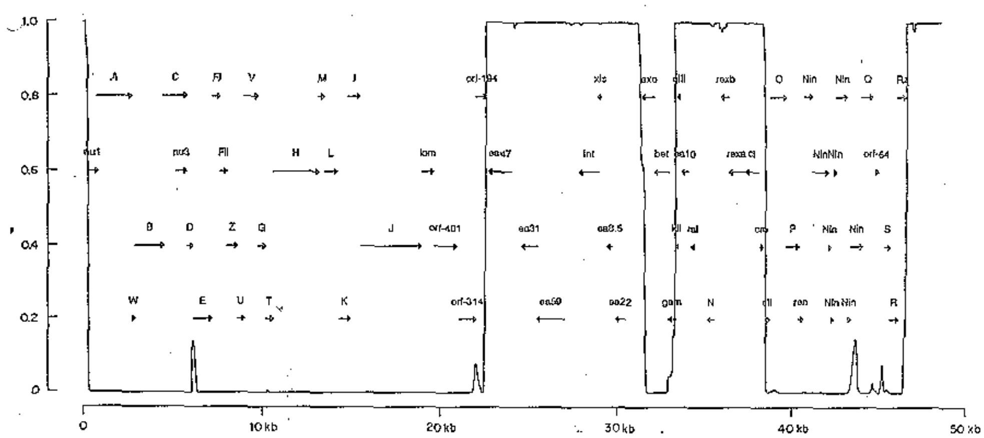


FIG. 5.4 – Courbes lissées du maximum de vraisemblance pour une chaîne de MARKOV à deux états cachés, avec chacun quatre états – un par base – sur la séquence du génome complet du bactériophage lambda. Sur l'axe vertical est indiquée la probabilité d'appartenance à l'état un. L'axe horizontal représente la séquence et les flèches indiquent les zones codant les protéines, dans leur sens de transcription (de [CHU92]).

à un même état caché est la plus forte. Par exemple, Churchill présente des décompositions de différentes séquences, tels des génomes mitochondriaux animaux (figure 5.3) selon deux états relevant de l'opposition purine-pyrimidine ({A, G } versus {T, C }), ou le génome du bactériophage lambda (figure 5.4) selon la meilleure modélisation de la séquence par deux chaînes de Markov de profondeur 1. En pour la détection du sens de lecture, on sait [AK90] que l'ARN messager préfère les codons qui commencent par une purine. Par conséquent, un biais existe entre les zones en sens direct et indirect, selon une plus forte proportion de purine (sens direct) ou de pyrimidine (sens indirect) en première position des codons.

Les chaînes de Markov à états cachés peuvent aussi être utilisées pour reconnaître les gènes dans une séquence. Par exemple, dans [KMD94], un modèle markovien est développé en ce sens pour reconnaître les gènes de *E.coli*. L'état caché qui correspond aux zones codantes modélise les fréquences d'occurences des codons dans les gènes, fréquences déterminées à partir d'une base de données de gènes déjà connus. L'état caché qui correspond aux zones non codantes modélise les extrémités des zones codantes (codons stop et codons initiateurs), ainsi que les fréquences d'occurences des bases sur ces zones.

Une autre utilisation des chaînes de Markov est celle qui est au cœur du programme GENMARK [BM93], pour reconnaître les parties codantes d'une séquence. Les chaînes de Markov relatives aux parties codantes sont dites non-homogènes ; c'est-à-dire qu'en fait six chaînes de Markov sont utilisées en même temps. Selon chacune des trois phases de codage, et selon chacun des deux sens de lecture (ce qui donne six), une chaîne de Markov de degré un est construite. La construction de chacune des six chaînes est faite à l'aide d'un ensemble d'apprentissage. Les zones non-codantes, quant à elles, sont modélisées par une chaîne de Markov de premier ordre homogène. Ensuite, sur la séquence étudiée, des parties successives (de taille 16, 32 ou 48 bases) sont extraites par une fenêtre glissante. La probabilité conditionnelle de chaque partie selon chaque chaîne de Markov du modèle codant est calculée, puis, grâce à des formules de Bayes, on calcule la probabilité que cette partie est codante. Cette méthode est extrêmement utilisée à l'heure actuelle pour détecter les zones codantes d'une séquence.

5.2.4 Minimisation d'énergie de dénaturation

Cette approche s'appuie sur des propriétés physico-chimiques de l'ADN. Des observations concernant la dénaturation et la renaturation de l'ADN (c'est-à-dire la séparation des deux brins de la double hélice, et le processus inverse) ont révélé que ces processus ne s'effectuent pas de façon continue, mais section par section. Une séquence d'ADN peut être découpée de manière unique et complète en segments de longueurs diverses emboîtés les uns dans les autres et possédant une stabilité chimique propre. Gabarro-Arpa et Michel [GAM82] ont cherché à opérer un partage hiérarchique d'une séquence pour retrouver ces sections, en fonction de leur facilité à subir une dénaturation, en utilisant les différences d'énergie libre des liaisons A–T et C–G qui sont cassées lors de la dénaturation. Si la différence d'occurences entre A ou T et C ou G sur un segment est assez grande, on peut dire qu'il est possible qu'une dénaturation s'y déclenche, ou si elle y est déjà engagée, qu'elle s'y propage. Le terme « assez grande » signifie que l'énergie libre d'une section sur laquelle la dénaturation s'effectue doit être suffisante pour vaincre un seuil d'équilibre chimique, et ceci dépend évidemment des différents facteurs qui interviennent dans les réactions chimiques, comme la température.

Dans une situation chimique donnée, en connaissant les facteurs qui permettent de décider si une section de la séquence correspond à la création potentielle d'une boucle (dénaturation), un algorithme simple permet d'opérer une partition en parties où la dénaturation peut se déclencher, où elle peut se propager, et où elle est impossible. Par exemple, dans [NM89], ce modèle de coût le plus simple pour évaluer si un morceau d'hélice qui contient x paires A–T et y paires G–C peut être transformé en une boucle est : $C = W - x\alpha + y\beta$ avec $W = 2$, $\alpha = 0.41$ et $\beta = 0.59$. Si $C < 0$, le domaine considéré est une boucle potentielle. En outre, si $-x\alpha + y\beta < 0$, une boucle déjà commencée au bord de ce morceau peut s'étendre sur tout le morceau[5]. À partir de telles équations, il est extrêmement facile de déterminer quelles zones correspondent à la création et à la propagation de boucles.

Dans la réalité, les facteurs qui permettent de reconnaître les zones de dénaturation (comme dans le modèle ci-dessus α et β) sont variables. En fonction de ces facteurs, les bornes des sections sont modifiées et une arborescence de bornes de dénaturations peut être construite [GAM82].

5.2.5 Variations de proportions en bases

Dans [Lob96a, Lob96b], Lobry considère les biais de proportion entre A et T, et C et G sur un même brin. En effet, une symétrie dans les substitutions entre les deux brins d'ADN, entraîne que sur un brin[6] $\pi_A = \pi_T$, et que de même $\pi_C = \pi_G$ (seconde loi de Chargaff). Un écart de cette situation signifie au contraire une assymétrie des mécanismes de substitution. Ainsi, Lobry a regardé sur trois génomes bactériens trois facteurs de variation à partir de ce modèle ; occurences en C et G : $\frac{\pi_C+\pi_G}{\pi_A+\pi_T+\pi_C+\pi_G}$, déviation de $\pi_A = \pi_T$: $\frac{\pi_A-\pi_T}{\pi_A+\pi_T}$, et déviation de $\pi_C = \pi_G$: $\frac{\pi_C-\pi_G}{\pi_C+\pi_G}$, en comptant le nombre de chaque base dans des fenêtres successives de grande taille (mille ou dix mille bases). Cette étude, à partir de l'indice $\frac{\pi_G-\pi_C}{\pi_G+\pi_C}$, révèle une forte tendance à ce que les G soient plus nombreux que les C sur le brin de leading de ces génomes, et inversement sur le brin de lagging. Le passage en ces deux zones se fait dans les voisinages de l'origine et de la terminaison de réplication (comme par exemple dans la figure 5.5). Une étude comparable effectuée avec des mots de huit nucléotides est présentée dans [SSKT98].

Ainsi, l'étude de Lobry témoigne d'un biais important avec une signification biologique, et il serait extrêmement intéressant de pouvoir grâce à ce biais déterminer assez précisément les origines de réplication. Dans [Gri98], la cri-

[5] Le terme $W = 2$ correspond à l'énergie nécessaire à la création de deux bornes lors de la création d'une boucle, alors qu'une boucle qui se propage ne provoque pas de création de bords.

[6] π_A est le nombre de A.

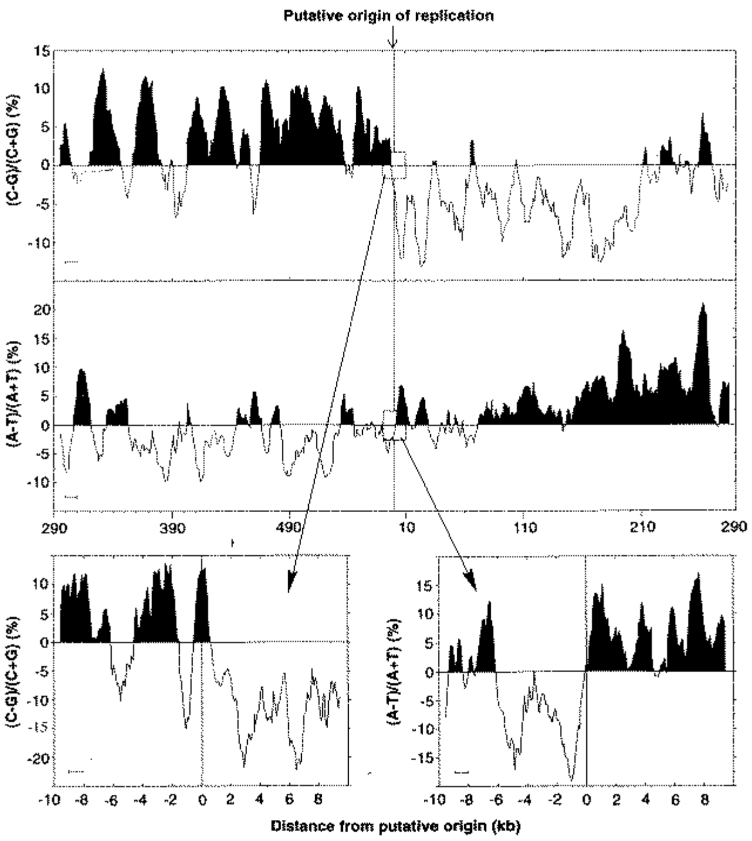


FIG. 5.5 – Corrélation entre les biais de proportion $\frac{\pi_A-\pi_T}{\pi_A+\pi_T}$, $\frac{\pi_C-\pi_G}{\pi_C+\pi_G}$ et l'origine de réplication pour *Mycoplasma Genitalium.* Les calculs de biais ont été faits avec une fenêtre glissante de 10^4 bases de long par pas de 1000 bases en haut, et avec une fenêtre glissante de 1000 bases de long par pas de 100 bases en bas (de [LOB96B]).

tique faite à cette étude concerne l'utilisation d'une fenêtre de taille fixe pour le comptage ; une trop petite fenêtre laisse passer trop de bruit et ne permet pas d'observer les variations de grande échelle, et une trop grande fenêtre est au contraire trop lissante pour espérer obtenir une position assez précise. Pour observer plus précisément les variations de proportions de nucléotides, GRIGORIEV propose de tracer la courbe des sommes successives des biais (pour G et C, calculer la somme successive de $\frac{\pi_G-\pi_C}{\pi_G+\pi_C}$) au long de la séquence, en utilisant de plus petites fenêtres (de taille uniforme) pour chaque calcul. Ainsi, sur plusieurs bactéries et archæa bactéries, cette méthode permet d'approcher les origines et terminaisons de réplication, en fonction des extrema principaux de cette courbe (figure 5.6).

Sur beaucoup d'études par biais de G - C, les variations de courbes ne sont pas aussi monotones que si cette variation était simplement due à la différence de proportion entre les zones de lagging et de leading. Par exemple, sur la courbe 5.6, on peut remarquer des forts changements de pentes sur certaines parties, comme entre 10% et 20%, ou encore 55% et 65% approximativement. Une expli-

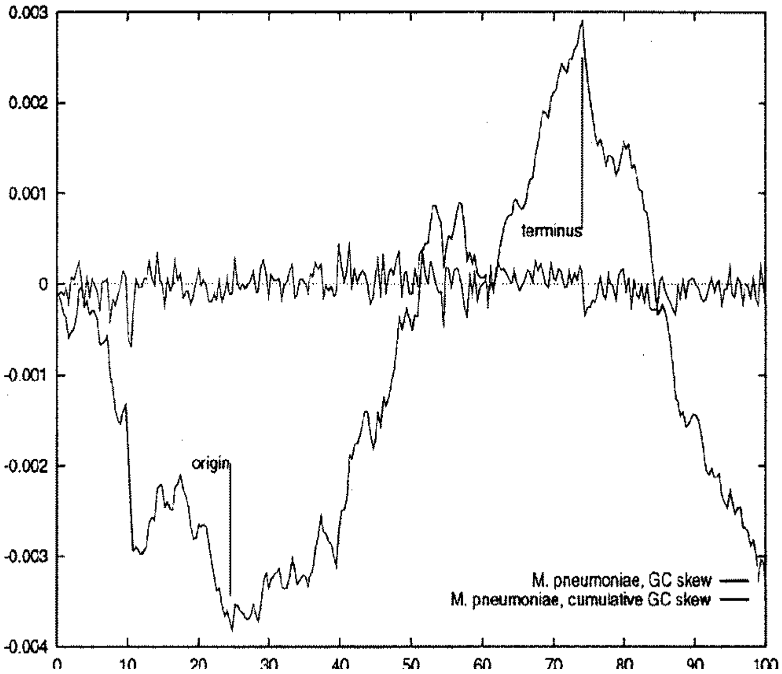


FIG. 5.6 – Courbes $\frac{\pi_G-\pi_C}{\pi_G+\pi_C}$ et $\sum\frac{\pi_G-\pi_C}{\pi_G+\pi_C}$ sur *M. pneumoniæ*. L'axe horizontal est gradué selon le pourcentage de la séquence totale. Sont indiquées l'origine et la terminaison de réplication déduites de la courbe (de [GRI98]).

cation de ces variations serait qu'une portion d'un brin qui correspondait à une partie en lagging (riche en C) se serait, par un processus de translocation, mise dans une partie de brin en leading (riche en G). Le processus inverse expliquerait les variations inverses. La pression de sélection agissant sur cette zone n'aurait pas (encore) suffit à rétablir un biais en accord avec les mécanismes usuels de réplication. Il est très intéressant de pouvoir déterminer assez précisément ces parties, mais deux problèmes se posent : comment les délimiter, et quel facteur de taille permet de choisir qu'une partie est intéressante, alors que telle autre n'est pas assez déterminante.

Une conséquence importante de ce résultat vis-à-vis d'approches de modélisation comme les chaînes de MARKOV à états cachés est que ces approches doivent prendre en compte ce type de biais à grande échelle.

5.2.6 Analyse discriminante

Dans [RDV99], une méthode d'analyse discriminante linéaire est utilisée pour évaluer si des critères (au même titre que les proportions relatives en C et G) permettent de discriminer un génome circulaire en deux parties, et dans ce cas si ces deux parties correspondent aux parties leading-lagging.

Les fonctions manipulées sont des fonctions linéaires de la position relative sur la séquence à une origine putative qui cherchent, selon un certain critère, à discriminer au mieux le génome en deux parties ; pour une origine putative, la précision d'une fonction est évaluée selon son aptitude à prendre des valeurs négatives sur la moitié du génome qui est d'un coté du diamètre qui passe par cette origine putative, et à retourner des valeurs positives sur la moitié du génome qui est de l'autre coté de ce diamètre.

Si un critère de discrimination est fixé, une courbe peut être tracée, en fonction des positions de l'origine putative, de la précision de la fonction linéaire de discrimination qui est établie sur ce critère (figure 5.7). Les critères utilisés sont les fréquences relatives en nucléotides sur des fenêtres de bases, les proportions en nucléotides sur les différentes positions des codons sur les gènes, les fréquences relatives des codons des gènes, ou encore les fréquences relatives en acides aminés traduits de ces gènes.

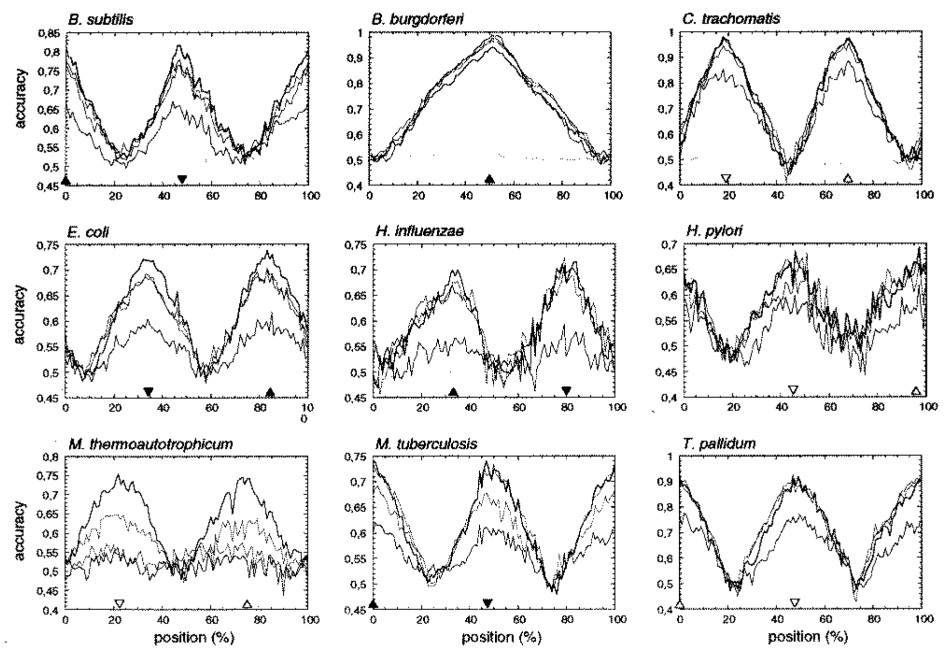


Fig. 5.7 – Précision de quatre fonctions linéaires discriminantes en fonction de la position putative de l'origine de réplication, pour quatre différents critères sur plusieurs génomes bactériens. Voir la légende dans le texte (extrait de [RDV99]).

Dans ces diagrammes, la position sur l'axe des abscisses est exprimée comme le pourcentage de la taille du chromosome. Pour chaque espèce, les quatre courbes représentent différents critères de discrimination (plus de détails et de couleurs dans l'article [RDV99]). Les triangles noirs représentent les origines (pointe en bas) et les terminaisons (pointe en haut) de réplication connues. Les triangles clairs représentent les origines et terminaisons proposées dans cet article, mais pas déterminées biologiquement.

On remarque que pour ces différents critères, la précision des fonctions discriminantes dépend fortement de l'origine putative, et les deux maxima de précision de ces fonctions sont souvent nettement détectés aux voisinages des origines et terminaisons de réplication (par symétrie de la précision des fonctions de discrimination). Il est rassurant, en regard des études précédentes, que le critère qui concerne les proportions de nucléotides permette de discriminer le génome en parties leading-lagging. Mais cette méthode détecte aussi l'existence d'un biais en proportions des positions des nucléotides dans les codons des gènes de part et d'autre de l'origine de réplication, ainsi qu'un biais en proportions en codons de ces gènes. Ceci peut dépendre directement du biais en C - G. Mais le biais observé en acides aminés est plus étonnant ; selon qu'un gène est situé d'un coté ou de l'autre de l'origine de réplication, il aura tendance à favoriser ou défavoriser la production en tel ou tel acide aminé.

Pour chaque ensemble de critères, les critères les plus influents dans la discrimination ont été recherchés, sur différentes espèces de bactéries. [RDV99] remarquent que sur la plupart des génomes bactériens étudiés, les facteurs les plus discriminants sont fortement semblables. Ainsi, sur tous les génomes, parmi l'ensemble des critères de discrimination selon la proportion relative en nucléotides sur les positions des codons des gènes, le facteur le plus discriminant en faveur de la partie leading est l'occurence de G en troisième position, et celui pour la partie lagging est l'occurence de C en troisième position. De même, en ce qui concerne les acides aminés produits, la thréonine (T) est universellement la plus discriminante en faveur de la partie leading, et la valine (V) est universellement la plus discriminante pour la partie lagging. Des observations similaires ont été faites sur les codons des gènes.

L'avantage de ce genre d'approche est de pouvoir automatiquement déterminer si un ensemble de critères permet de relever des biais de proportions importants dans la séquence, et quels sont les plus discriminants parmi ceux-ci.

Le partitionnement maximalement prédictif a pour objectif de partager au mieux des séquences, aussi à partir d'un ensemble de critères, non pour discriminer au mieux selon ces critères, mais pour extraire des zones d'homogénéité des biais qui peuvent être révélés par cet ensemble de critères.

5.3 Applications du partitionnement maximalement prédictif

Une approche telle que celle que nous venons de voir, parce qu'elle exprime des biais sur plusieurs critères à la fois différents les uns des autres mais pas totalement indépendants entre eux, nous amène à utiliser les multiples possibilités offertes par les lexiques de prédicteurs. Si on adapte à chaque biais un lexique ad hoc, le partitionnement maximalement prédictif extrait un partage en zones où le biais est le plus fortement exprimé. Il serait intéressant de comparer, pour les différents biais proposés ici, quel serait le comportement de notre procédé.

Pour l'instant, il s'agit d'étudier, dans plusieurs cas, quel comportement a le partitionnement maximalement prédictif sur des séquences génétiques, et s'il est à même de faire apparaître des biais de constitution dans ces séquences. Nous essaierons aussi d'exploiter les connaissances biologiques au sujet des biais de composition, comme ceux présentés ci-dessus, pour générer des partitions biologiquement intéressantes.

5.3.1 Occurrences de nucléotides sur le bactériophage λ

À partir de la modélisation par chaînes de MARKOV à deux états cachés sur le génome circulaire du bactériophage λ, CHURCHILL parvient à retrouver les différentes zones qui correspondent au sens de transcription des gènes (figure 5.4). Ce modèle gère des chaînes de MARKOV à quatre états, un par nucléotide. À notre tour, nous avons voulu étudier ce que produisait le partitionnement maximalement prédictif avec un lexique constitué des quatre nucléotides : `L = A C G T`. Si on regarde la courbe des coudes de prédictions en fonction du nombre de classes (figure 5.8) on remarque une nette tendance pour un critère d'arrêt à quatre classes.

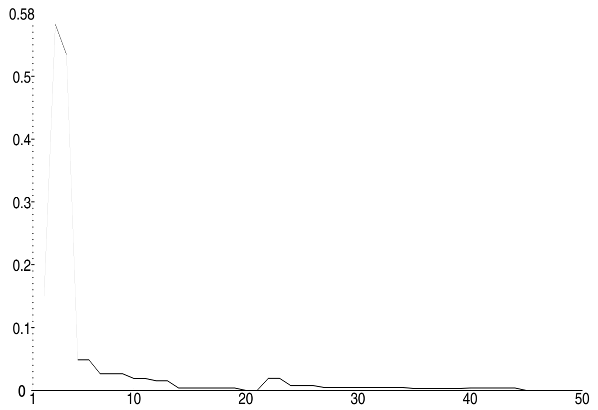


FIG. 5.8 – Angles des coudes de prédiction relatifs au partitionnement du génome du bactériophage λ avec le lexique `L = *`

En regardant les k-partitions produites pour k allant de 1 à 10, on observe que les bornes de la 4-partition correspondent aux limites des sens de lecture des gènes, et que la classe la plus à droite correspond à une zone non transcrite (figure 5.9).

Plus en détail, il apparaît que la 5-partition révèle en outre sur la gauche une petite classe prédite par `T`, et sur laquelle, comme à l'autre extrêmité, il n'y a pas de gènes codés. Dans la réalité, le génome est circulaire et ces deux parties sont réunies. Cette classe était révélée dans la modélisation de CHURCHILL. Que l'on ait choisi de partager en 4 classes, et non en 5, tient à la petitesse de cette classe et au fait que le gain de prédiction procuré à sa découverte est très faible en comparaison des grandes classes trouvées auparavant. Cette petitesse provoque aussi sa disparition au sein de la 6-partition (par la découverte d'un gène riche en A qui a perturbé la méthode de CHURCHILL), mais sa persistence par la suite est tout de même un gage de sa qualité, si on la compare aux classes beaucoup plus grandes qui sont construites par la suite. Le tableau suivant permet de comparer les limites des zones de même sens de lecture aux césures obtenues dans les 5-partitions optimales. Pour chaque césure sont indiquées les valeurs extrêmes obtenues dans l'ensemble des 5-partitions optimales.

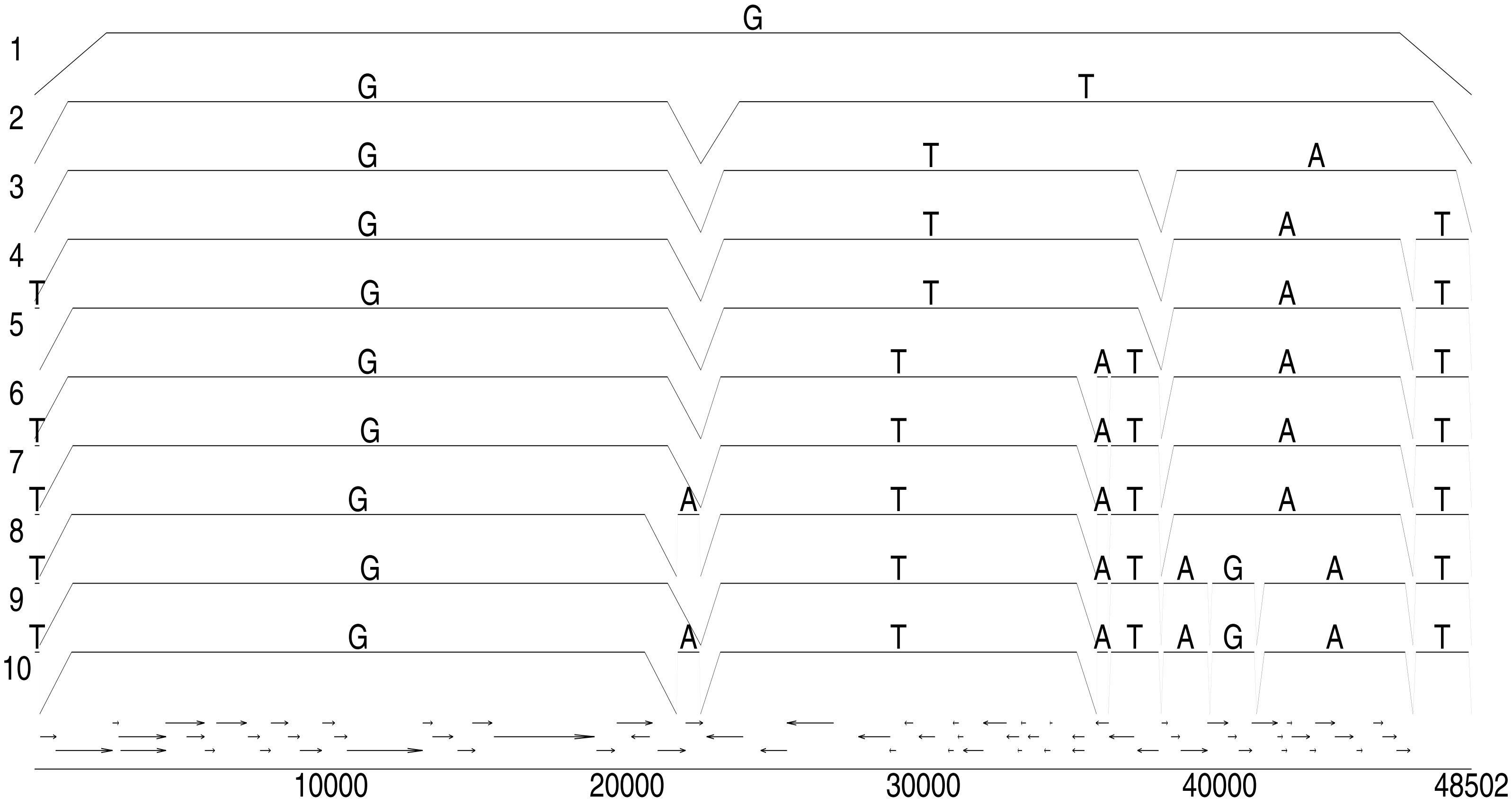


FIG. 5.9 – k-partitions successives optimales de la séquence génomique du bactériophage λ avec `L = *`. Le trait gradué représente la séquence (48502 bp) et les flèches au dessus de celui-ci les gènes dans leur sens de lecture.

limites des zones de même sens de lecture	césures extrêmes	nombre de césures
191	174–180	5
22557–22686	22408–22501	4
37940–38040	38026–38044	6
46427	46524–46536	3

5.3.2 Origines de réplication

Comme il a été montré dans [Lob96a], entre les parties lagging et les parties leading d'un génome, les proportions en C et G sont différentes. Il s'agit de mettre à profit cette différence pour, à notre tour, prédire les localisations des origines et terminaisons de réplication. De façon la plus naturelle, les prédicteurs admissibles seront `C` et `G`. Ainsi, en prenant comme lexique de prédiction `L = C G`, la « lutte » entre `C` et `G` va faire apparaître les classes où la différence de composition en ces lettres est la plus grande. Regardons le partitionnement en 10 classes produit sur la séquence de la bactérie *Haemophilus influenzae* (1830135 bp) (figure 5.11).

On peut remarquer que les césures de la 3-partition se trouvent aux voisinages de l'origine de et de la terminaison de réplication, la classe prédite par `G` correspondant bien à la zone de leading, et celles par `C` à la zone de lagging[7]. Mais, si on connait pas ces points particulières, rien ne permet sur cette figure d'estimer que les classes de la 3-partition sont celles qui nous intéressent, et que celles des autres k-partitions, pour $k = 5, 7, 9, \dots$, ne sont pas aussi valables. Pour cela, si on observe la courbe des coudes de prédiction (figure 5.10), on remarque très nettement que le choix de 3 classes est très favorisé comme critère d'arrêt.

Le tableau 5.3 présente pour différentes bactéries et archebactéries pour lesquelles la courbe des coudes de prédiction a un comportement aussi prononcé (en désignant comme nombre de classes 2 ou 3), les positions obtenues par partitionnement avec le lexique `L = C G`, en prenant le coude de la courbe d'incréments comme critère d'arrêt. Lorsqu'une position est 0, cela signifie que la partition relevée a deux classes.

Ces résultats montrent une plutôt bonne corrélation avec les connaissances biologiques. Les résultats sur *Pyrococcus horikoshii* et *Mycoplasma genitalium* sont intéressants car ils proposent de nouvelles valeurs. En particulier, le partage en deux de *Pyrococcus horikoshii* propose une terminaison à $43,6$ % de la séquence, ce qui est curieux, puisque l'on devrait s'attendre à ce qu'elle soit plutôt vers les 50 % (comme sur la figure 5.2) si la réplication progresse à la même vitesse des deux cotés de l'origine de réplication.

Mais sur d'autres génomes de bactéries, tels ceux de *Aquifex aeolicus*, *Synechocystis sp.*,*Thermotoga maritima*, ou d'archæa bactéries comme *Archaeoglobus fulgidus*, *Methanococcus jannaschii*, *Pyrococcus abyssi*, le partitionnement ne produit pas un tel comportement. Il faut remarquer que pour ceux-ci, il n'a pas été non plus mis en évidence de biais de proportions entre C et G. Plusieurs

[7]Le chromosome est circulaire, si bien que la partie ici à gauche de l'origine de réplication rejoint celle qui est ici à droite de la terminaison de réplication.

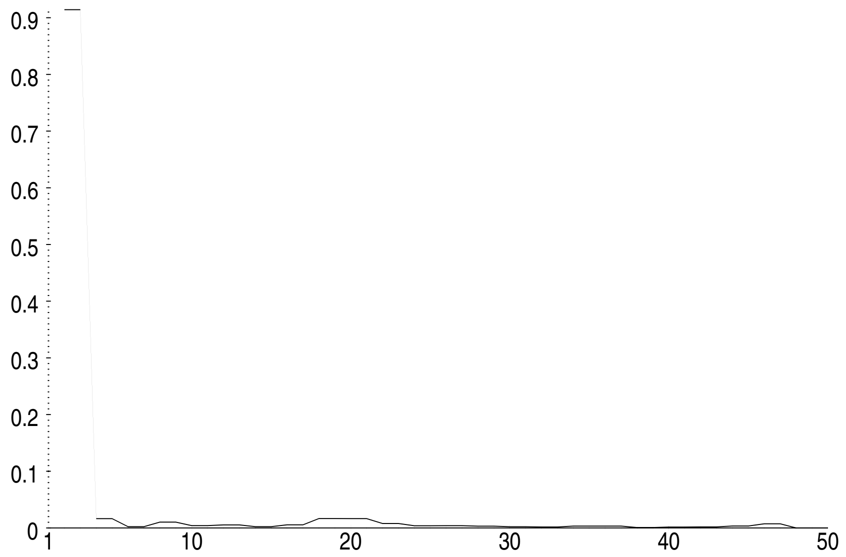


FIG. 5.10 – Courbe des coudes du partitionnement de *H. influenzae*, avec `L= C G`.

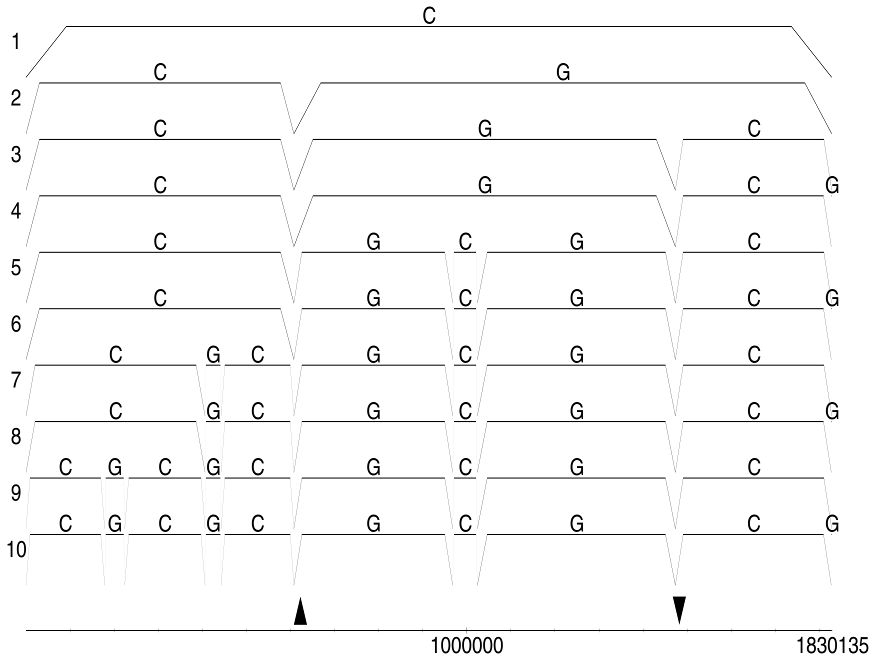


FIG. 5.11 – Partitionnement en 10 classes de *H. influenzae*, avec `L= C G`. Les triangles noirs représentent la position de l'origine (pointe en haut) et de la terminaison (pointe en bas) de réplication.

Organisme	Origine	Terminaison	Césure de `C` vers `G`	Césure de `G` vers `C`
B.burgdorferi	50.1	0	50.3	0
B.subtilis	0	47.8	0	46.1
C.trachomatis	69	19	69.1	18.7
E.coli	84.6	34.1	84.6	33.4
H.influenzae	32.9	80.1	33.2	80.1
H.pylori	96	47	98	48.8
M.tuberculosis	0	49	0	46.3
M.pneumoniae	25.1	73	25	74.1
M.genitalium	0	?	0	50.8
M.thermoautotrophicum	75	24	74.9	23.3
P.horikoshii	?	?	0	43.6
R.prowazekii	0	56	0	56.5
T.pallidum	0	48	0	48.6

TAB. 5.3 – Origines et terminaisons de réplication comparées aux césures trouvées par partitionnement.

hypothèses existent pour expliquer cela. Par exemple, dans *Pyrococcus abyssi*, des translocations entre la partie leading et la partie lagging insèrent des parties riches en G dans la zone normalement riche en C et inversement. Aussi, dans des séquences comme *Archaeoglobus fulgidus* et *Methanococcus jannaschii* il n'a pas encore été trouvé d'origine de réplication, et l'absence de biais peut aussi s'expliquer par le fait que peut-être il n'y pas d'origine de réplication (elle peut commencer n'importe où, ou alors le mécanisme de réplication y est différent), ou alors au contraire qu'il y a plusieurs origines de réplication, et le biais entre les zones de leading et de lagging est moins prononcé. Une autre explication serait que la réplication se fait selon un mécanisme de rolling circle, auquel cas le brin étudié est tout en leading ou lagging, et il n'y a pas de biais de proportions à déceler sur ce brin.

Si on regarde dans le détail le partitionnement maximalement prédictif de *M. pneumoniæ* (figure 5.12) sur les moments stables, en comparaison de la figure 5.6, on remarque que les zones où les inversions de biais GC apparaissent sont également mises en évidence. Les avantages de cette méthode-ci par rapport à l'étude de la courbe des variations de sommes de biais en GC sont que ces zones d'inversement de biais sont nettement délimitées, ordonnées selon leur importance, et que le fait de ne pas utiliser de fenêtre de lissage permet une bonne précision quant à leur délimitation.

5.3.3 Approche exploratoire

Nous avons expérimenté de façon exploratoire le partitionnement maximalement prédictif sur des séquences génétiques avec des lexiques divers et variés. En dehors des expérimentations précédentes qui corroborent des observations de biais de proportions, nous avons observé des comportements du partitionnement qui sont très intéressants, et éclairent l'intérêt de mettre en pratique

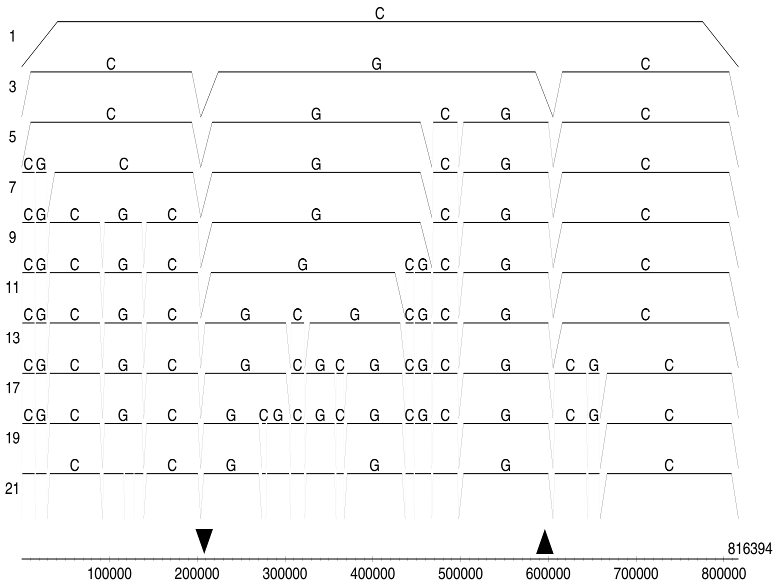


FIG. 5.12 – partitionnement en 21 classes de la séquence de *M. pneumoniæ* avec `L = C G`. Seules les partitions aux moments stables sont dessinées. Les triangles noirs représentent la position de l'origine (pointe en haut) et de la terminaison (pointe en bas) de réplication.

cette méthode dans des approches exploratoires automatisées, et/ou en utilisant les découvertes biologiques concernant des biais de proportions de bases, mots, motifs,...

Nous présentons trois expérimentations avec différents types de lexiques.

5.3.3.1 Partitionnement marqué

Un exemple simple est le partitionnement de la séquence de transcription de l'actine du rat, pratiqué avec le lexique `L = *`. Si on regarde la courbe de la fonction de coude (figure 5.13), on remarque que le partitionnement pertinent se démarque en cinq classes, avec une valeur d'angle de coude de 0.4 radian (23 degrés).

Pourtant, ce partitionnement ne révèle pas, à notre connaissance, de parties qui ont une explication biologique (figure 5.14), à ceci près que la zone qui est reconnue comme riche en C contient tous les exons. Bien entendu, aux niveaux inférieurs, elle est divisée, mais cela se passe après le moment stable le plus déterminant, et fait ainsi plus partie du processus normal de partage d'une partie quelconque.

Le même type d'observation a été fait sur d'autres types de séquences, avec différents lexiques. On note que le partitionnement extrait ainsi de nettes structures de séquences sur lesquelles, *a priori*, on ne l'attendrait pas, au même titre que des structures prévisibles furent construites sur d'autres séquences avec des biais connus (comme dans les exemples précédents).

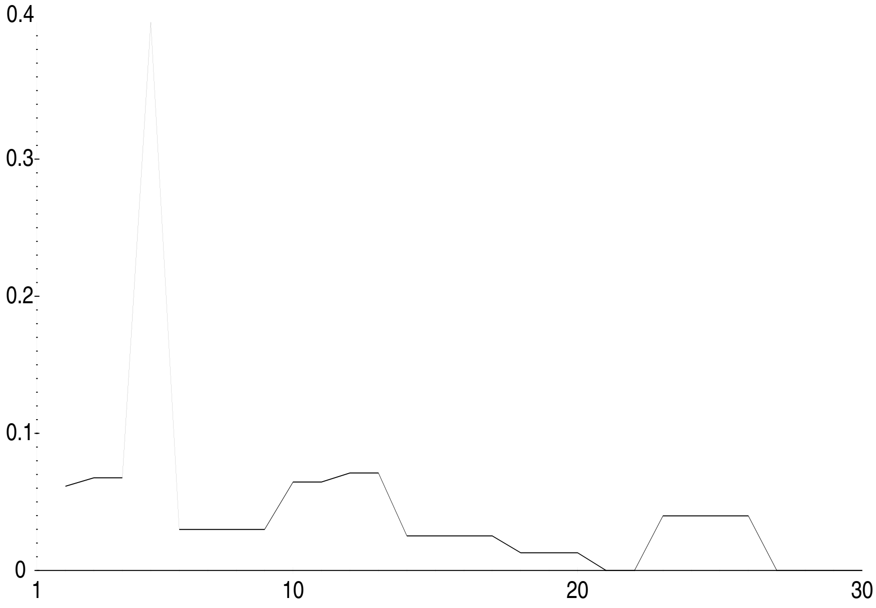


FIG. 5.13 – Courbe de la fonction de coude du partitionnement avec `L = *` de la zone de transcription de l'actine du rat.

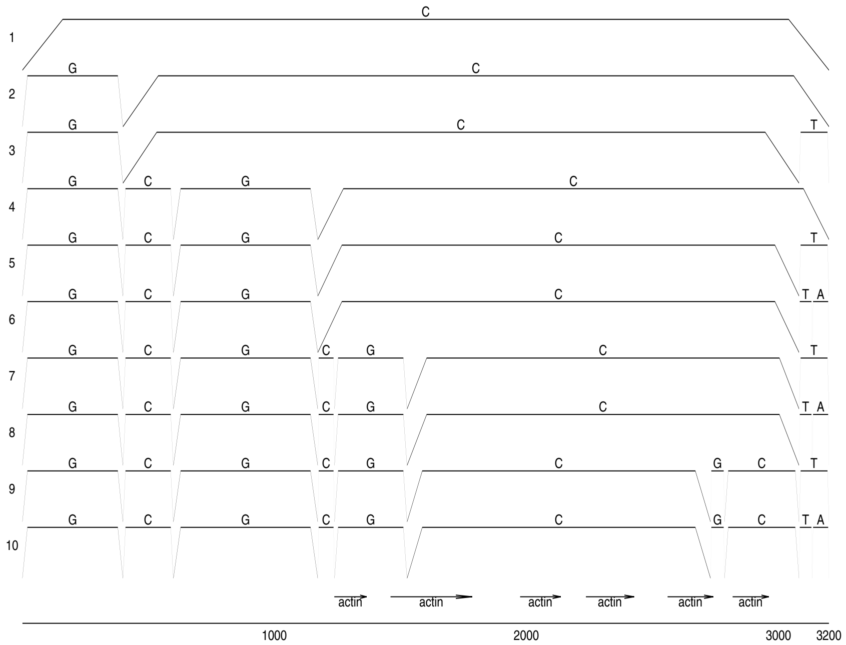


FIG. 5.14 – Partitionnement en dix classes avec `L = *` de la zone de transcription de l'actine du rat.

5.3.3.2 Phases de lecture

Une autre expérimentation a consisté à partager le génome mitochondrial de l'homme avec plusieurs lexiques. Cette approche peut donner une idée des relations qui existent entre des partitionnements sur une même séquence mais avec des lexiques proches.

En prenant comme lexique `L = *`, on observe sur la courbe de fonction de coude un maximum très net au niveau 2 (figure 5.15).

En fait, sur la figure 5.16, le partage en 2 parties correspond à la détection de la partie riche en A où sont codés les ARN ribosomiques, comme dans la figure 5.3, page 112. Le partitionnement en trois classes serait, biologiquement et en comparaison avec l'étude de CHURCHILL, plus probant. On remarque d'ailleurs que la partie à gauche de la classe prédite par A est vite trouvée, puis gardée, mais elle n'est pas mise en évidence car sa découverte est concurrencée par la classe riche en T trouvée au niveau 4 du partitionnement. On voit en outre peut-être une tendance du partitionnement à trouver des limites de gènes, mais cela n'a rien d'évident.

On a observé que pour le lexique `L = **` aussi le seul moment stable remarquable est la détection de zone riche en A. Hormis cette détection, le partitionnement avec des motifs de deux lettres ne semble pas ici présenter d'intérêt.

Par contre, en effectuant le partitionnement de cette séquence avec le lexique `L = ***`, on observe que le maximum de coude est très nettement au niveau 10 (figure 5.17). Si on observe le partitionnement jusqu'à la 10-partition (figure 5.18), on voit que les limites des classes correspondent assez précisément aux limites de longs gènes, qui sont des gènes codant les protéines. Le gène long[8] qui est entre les positions 14149 et 14643 dans le sens inverse se distingue très nettement. En outre, les prédicteurs des classes qui correspondent aux zones géniques dans le sens direct sont tous, à une permutation près, `ATC`. En fait lorsque l'on regarde plus en détail, on s'aperçoit que tous les gènes codant les protéines sont prédits par le prédicteur structuré `ATC` calé de telle sorte que `A` prédit les première, quatrième, septième... lettres de ce gène.

Par exemple, si on observe la classe qui englobe la position 10000. Son prédicteur est `ATC`, et elle commence en position 9234. Or parmi les cinq gènes qu'elle englobe, les trois plus longs commencent aux positions 9207, 10059 et 10470. Ces positions sont toutes à un multiple de 3 de la position de début de la classe, ce qui signifie que ces gènes sont en effet tous prédits par `ATC`. En revanche, les deux plus petits gènes commencent en 9991 et 10405, ce qui ne correspond pas à la même phase que les plus grands. Ces deux gènent codent pour des ARN de transfert, si bien qu'ils n'ont pas de cadre de lecture[9]. Par conséquent, ils n'ont pas de raison de répondre favorablement à une quelconque période de trois lettres. Dans cette séquence, tous les gènes qui sont nettement plus petits que les autres codent pour des ARN de transfert.

Ainsi, on peut estimer que le partitionnement met ici en relief les changements de phase entre des régions de gènes, et un prédicteur qui semble un

[8] NADH deshydrogénase sous-unité 6

[9] C'est-à-dire qu'ils ne sont pas concernés par le mécanisme de traduction, et ne sont donc pas structurés en codons.

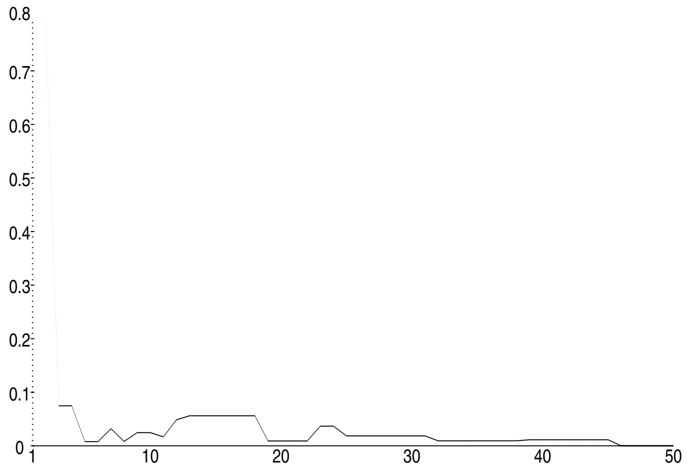


FIG. 5.15 – Courbe de la fonction de coude du partitionnement du génome mitochondrial de l'homme avec `L = *`.

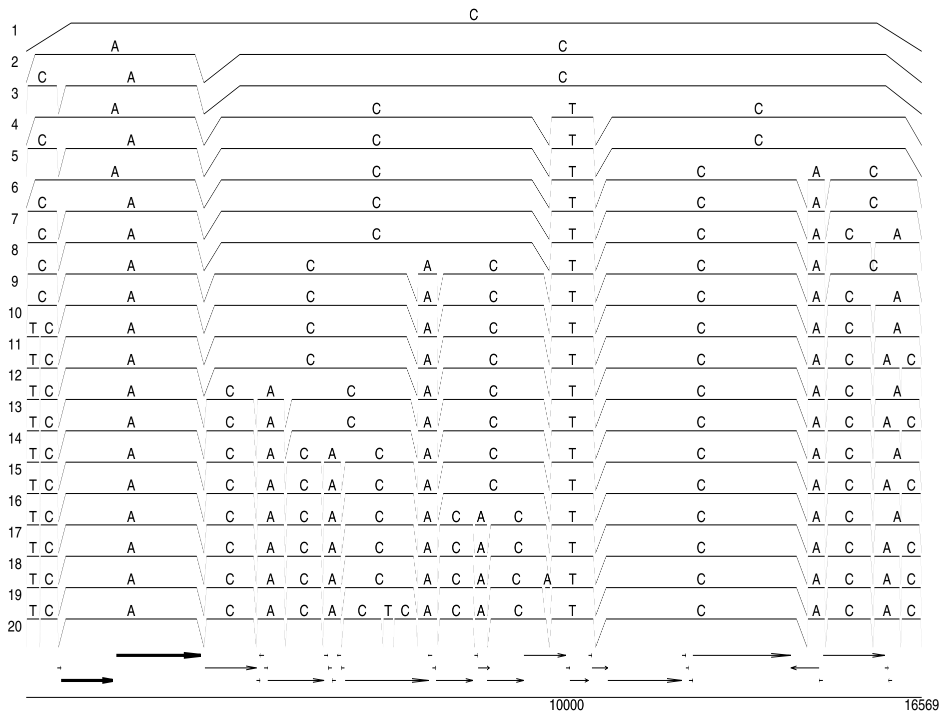


FIG. 5.16 – Partitionnement en vingt classes du génome mitochondrial de l'homme avec `L = *`. Les gènes sont tracés selon leur sens, et les deux gènes qui correspondent aux ARN ribosomiques $12S$ et $16S$ sont tracés en gras.

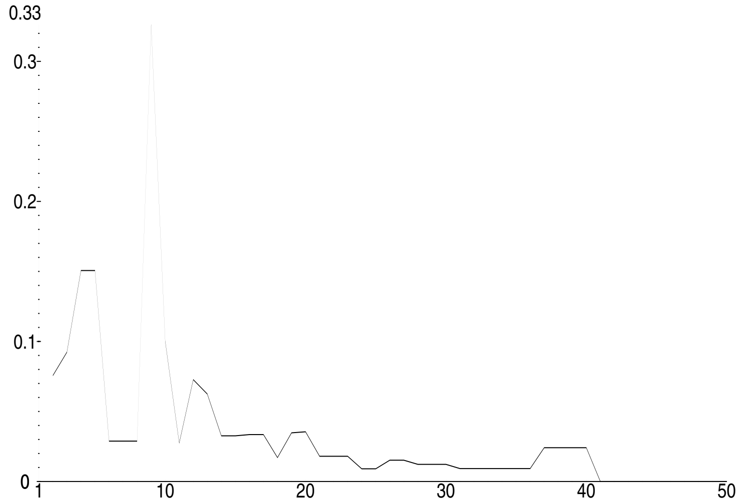


FIG. 5.17 – Courbe de la fonction de coude du partitionnement du génome mitochondrial de l'homme avec `L = ***`.

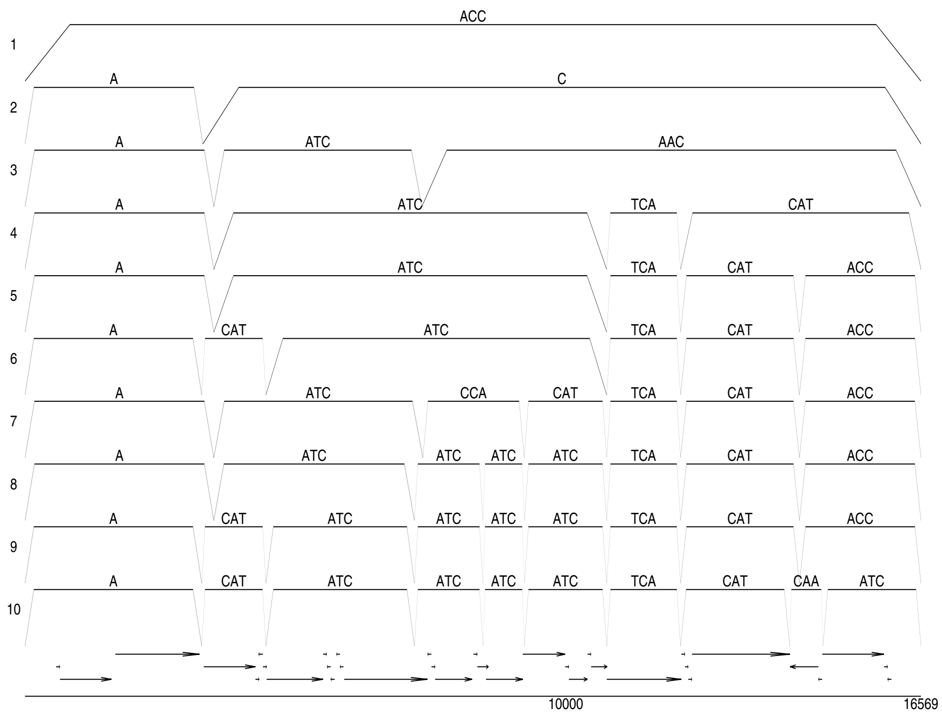


FIG. 5.18 – Partitionnement en dix classes du génome mitochondrial de l'homme avec `L = ***`. Les gènes sont tracés selon leur sens.

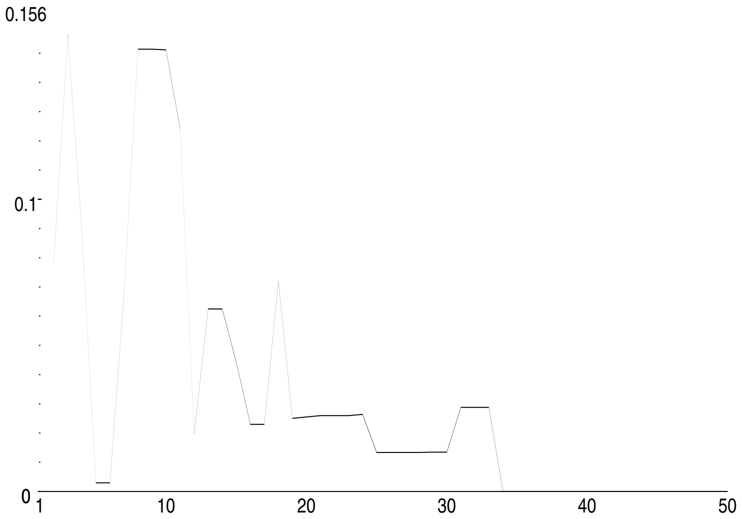


FIG. 5.19 – Courbe de la fonction de coude du partitionnement du génome mitochondrial de l'homme avec `L = -**`.

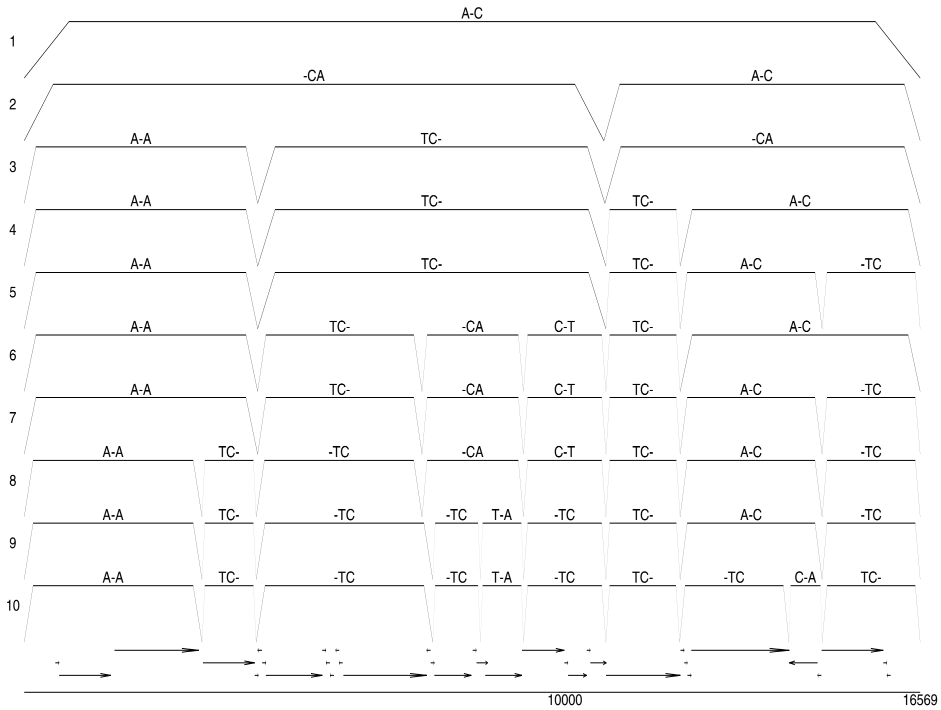


FIG. 5.20 – Partitionnement en dix classes du génome mitochondrial de l'homme avec `L = -**`. Les gènes sont tracés selon leur sens.

consensus accepté par l'ensemble des gènes codant les protéines. La forte influence du biais en A sur gauche de la figure est résolue dès le début du partitionnement et, en comparaison des autres biais qui sont découverts par la suite, elle est assez homogène pour ne pas apporter de perturbation à la découverte de biais dans la partie droite de la figure.

Considérer un lexique de motifs de trois lettres semble intéressant pour pratiquer le partitionnement sur cette séquence. On peut alors se demander s'il y a, dans les motifs, une lettre particulière (par exemple le `G` en troisième position de codon...) qui détermine la qualité du partitionnement selon des motifs de trois lettres.

On effectue le partitionnement selon le lexique `L = **-`. Comme il n'y a pas de lettre `-` dans la séquence, cela signifie que les prédicteurs sont des motifs de période trois mais dont seulement deux lettres servent à la prédiction. La courbe de la fonction de coude (figure 5.19) ne présente pas du tout la même apparence qu'avec les autres lexiques.

En particulier, il y a deux moments stables très nets, 3 et 10. Les maxima n'ont pas une valeur très élevée, mais ceci peut s'expliquer par le fait qu'ils sont deux, ce qui est certainement lié à la manifestation de deux biais conjointement : la zone riche en A en début de séquence, et les partitions des gènes selon les motifs de trois (figure 5.20). La partie riche en `A` est tout de suite détectée, et les classes de la 10-partition correspondent bien à des limites de gènes. Les classes où les gènes codant des protéines sont dans le sens direct ont comme prédicteur `TC-`, sauf un qui a pour prédicteur `T-A`. Les gènes codant des protéines concernés par les classes prédites par `TC-` sont tous en phase avec le prédicteur structuré `-TC`, et celui de la classe prédite par `AT-` est en phase avec `AT-`, ce qui signifie que sa première lettre est prédite par `A` et sa deuxième par `T`.

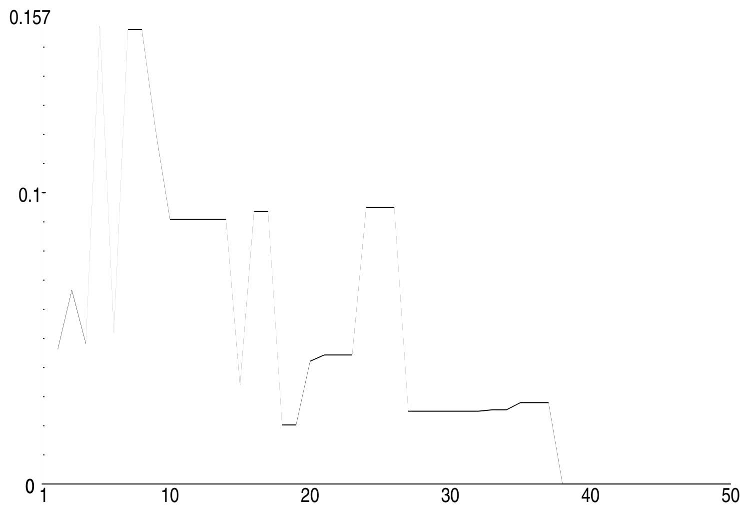


FIG. 5.21 – Courbe de la fonction de coude du partitionnement du génome mitochondrial de l'homme avec `L = -*-`.

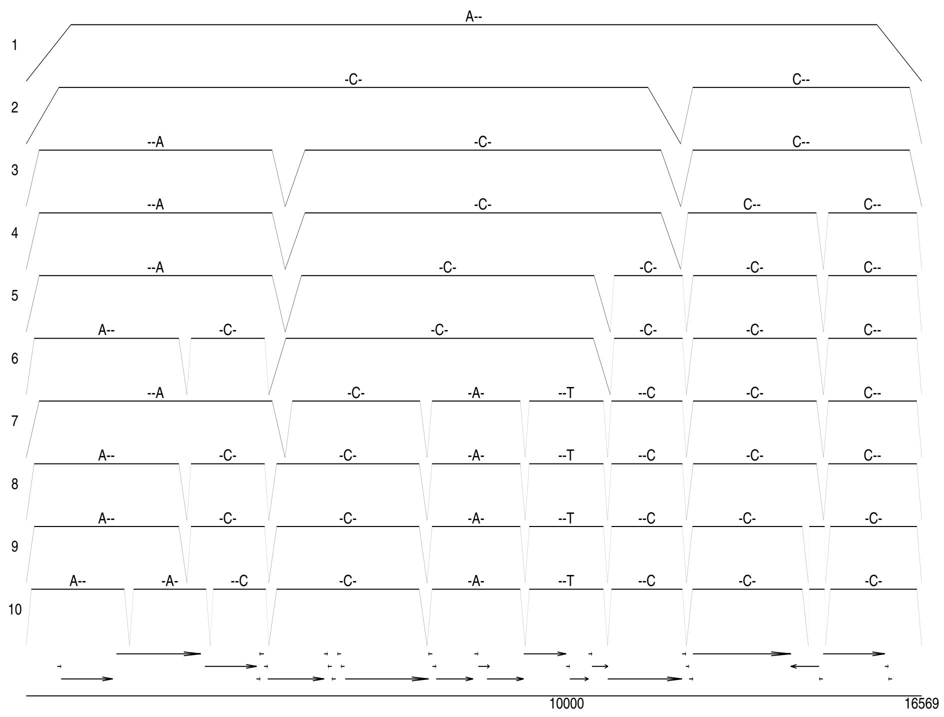


FIG. 5.22 – Partitionnement en dix classes du génome mitochondrial de l'homme avec `L = -*-`. Les gènes sont tracés selon leur sens.

À l'inverse du partitionnement précédent, selon les critères d'angle, la découverte de la zone riche en A est aussi importante que la découverte ultérieure des phases des gènes. L'explication de cela est que cette zone riche est très nettement confondue avec le gène voisin jusqu'au niveau huit. Dans le partitionnement avec `L = ***`, cette zone riche en A n'est pas vraiment sensible à une période de trois, puisque le prédicteur qui lui est associé est `A`. Ceci est cohérent puisque la zone riche en A correspond à l'expression des ARN ribosomiques, qui n'ont pas non plus de phase de lecture. Avec `L = -**`, l'influence sur la prédiction de la période de trois lettres en comparaison de l'influence de la proportion en lettres, est plus forte qu'avec `L = ***`. Par conséquent, la discrimination entre les deux zones, celle riche en A et celle qui correspond aux gènes, est déterminée à un plus grand nombre de classes.

On peut alors se demander ce que donnerait un partitionnement qui ne jouerait que sur les phases de prédicteurs. Avec le lexique `L = -*-`, une lettre sur trois est évaluée, si bien qu'une classe est d'autant plus pertinente qu'elle présente, dans une de ses trois phases, une périodicité de trois selon une lettre. La courbe de la fonction de coude 5.21 présente deux maxima (en 5 et 8) qui sont encore moins nets que pour la courbe 5.19. La valeur de l'angle du niveau 9 est la moyenne des valeurs d'angle en 8 et 10 niveau. On peut aussi considérer ce niveau. Si on regarde le partitionnement en neuf classes (figure 5.22), on note que des zones assez satisfaisantes sont trouvées.

Il est normal que la seule prédiction d'une lettre sur trois suffise à déterminer la phase, mais on remarque que cette détermination était plus nette dans les deux partitionnements précédents. La tendance est la même que ci-dessus, à savoir que le fait de ne plus considérer qu'une lettre sur trois amoindrit la facilité de détermination de la zone riche en A.

5.3.3.3 Critère d'espace entre mêmes bases

Dans [FIC82], FICKETT présente un biais de proportions entre parties codantes et non codantes. En particulier, il remarque que sur les parties codantes, les bases ont bien plus tendance à être semblables si elles sont distantes d'un multiple de trois positions que sur les parties non codantes. Sur une partie non codante, la proportion de positions où les lettres espacées de deux positions sont semblables est assez similaire à la proportion de positions où les bases espacées de trois lettres sont semblables.

Nous avons décidé d'utiliser un lexique qui utilise ce biais de proportion. Il s'agit de faire lutter un prédicteur qui signifie que la lettre considérée et celle qui est trois positions plus loin sont semblables, contre un prédicteur qui signifie que la lettre considérée et celle qui est quatre positions plus loin sont semblables. Les prédicteurs d'objets sont donc :

`["A??A""C??C""T??T""G??G"]`

et

`["A???A""C???C""T???T""G???G"]`

Sur la seconde moitié de la séquence du premier chromosome de *Leishmania major* (de 140000 à 268984 bp), le premier prédicteur prédit 1.19 fois plus que le second. Si, pour étudier la variation sur cette séquence par rapport à cette moyenne, on opère un partitionnement en 60 classes avec comme lexique

`L = ["A??A""C??C""T??T""G??G"] ["A???A""C???C""T???T""G???G"]`(1.19)

on obtient comme courbe de la fonction de coude la figure 5.23.

Cette fonction favorise très nettement la partition de la séquence en deux classes. Mais si on continue le partitionnement jusqu'au niveau 60, on obtient la figure 5.24, sur laquelle les gènes sont très nettement déterminés par le partitionnement. Cette détermination est due au fait que le biais que l'on exprime dans le lexique n'ait pas été « déséquilibré » par un biais à plus grande échelle. Le fait que le partage se soit nettement décidé au niveau deux aurait pu faire craindre cela.

Le fait de continuer à partager la séquence bien après le moment d'arrêt apparent replace le partitionnement maximalement prédictif dans une approche de redescription de séquences, plus que dans une approche d'extraction de structure tel que cela a été le cas dans les expérimentations précédentes.

Cette expérimentation révèle que le biais étudié est à peu près uniforme sur l'ensemble des gènes de la séquence, et est aussi à peu près uniforme sur chacun de ces gènes. En effet, ce n'est pas parce que l'on observe que des gènes – ou des parties quelconques – font preuve d'un biais de composition, que ce biais est uniforme sur chacun de ces gènes ; cela supposerait que chaque gène est

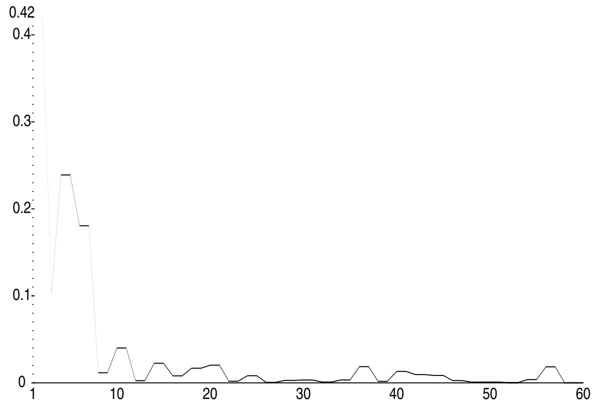


FIG. 5.23 – Courbe de la fonction de coude du partitionnement de la seconde moitié de la séquence du premier chromosome de *Leishmania major* avec `L = ["A??A""C??C""T??T""G??G"]["A???A""C???C""T???T""G???G"](1.19)`.

vis-à-vis de ce biais de composition à peu près uniforme sur toute sa longueur. Ainsi, ce n'est pas parce que l'on remarque que les gènes d'une séquence sont tous caractérisés par une plus forte proportion d'un certain oligonucléotide que les parties non codantes, que l'utilisation de ce biais pour le partitionnement va révéler ces gènes. Si ce biais n'est pas uniforme dans un gène, le partitionnement va créer une césure entre les parties de ce gène où le biais est fort et où il est faible, voire même inverse. Pourtant, dans cette expérimentation, beaucoup de gènes demeurent longtemps à peu près entiers dans les partitions.

La vérification de l'homogénéité sur de telles classes, *indépendamment de leur taille*, n'aurait pas été satisfaisante avec une fenêtre glissante, puisque cela aurait fixé un *a priori* sur les tailles de ces classes, alors que les gènes de cette séquence sont de taille très variable. En outre, considérer de toutes petites fenêtres n'extrait pas de zones d'homogénéité, puisque le bruit est trop important. C'est vraiment l'approche globale du partitionnement qui joue en sa faveur.

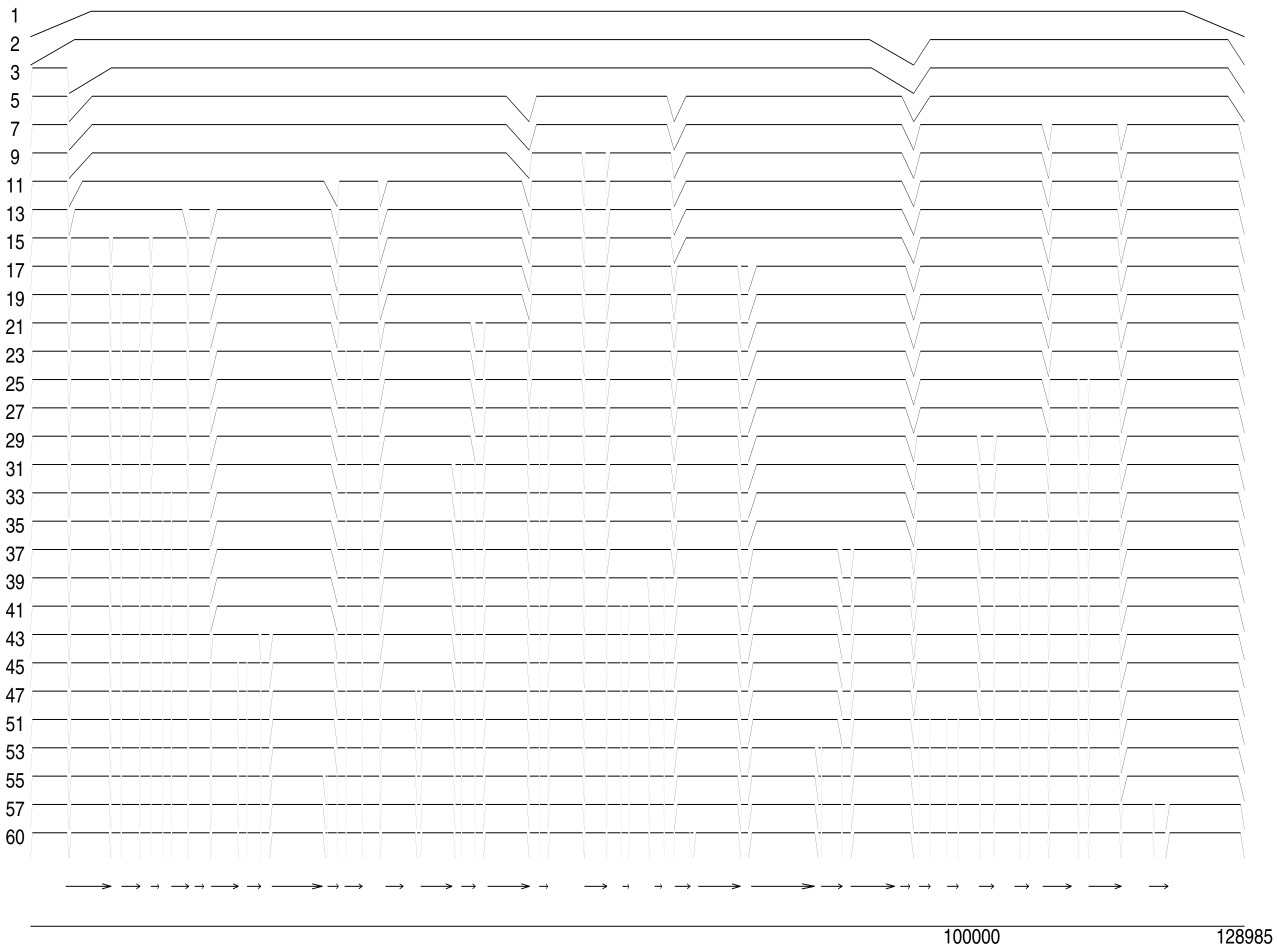


Fig. 5.24 – Partitionnement en 60 classes de la seconde moitié de la séquence du premier chromosome de *Leishmania major* (de 140000 à 268984 bp) avec `L = ["A??A""C??C""T??T""G??G"] ["A???A""C???C""T???T""G???G"](1.19)`. Les gènes sont figurés par des flèches dans leur sens de lecture.

Conclusion et perspectives

Sur une séquence d'objets qualitatifs et selon un ensemble de prédicteurs donné, effectuer un partitionnement maximalement prédictif permet d'obtenir (au moins) une partition prédictive de cette séquence en segments, de telle sorte que ces segments décrivent au mieux la structure de la séquence, et que cette partition décrive avec fiabilité le contenu de cette séquence. Dans le contexte de la classification, nous avons défini rigoureusement ces notions. Nous avons également exposé en quoi le partitionnement permettait de redécrire des séquences de lettres, mais ce n'est que l'aspect « description » de cette approche. L'autre aspect concerne le problème d'extraire une structure d'une séquence, et est lié au problème du nombre de classes, qui est classique en classification.

À la fois dans un but applicatif sur de grandes séquences et pour l'étude du comportement du partitionnement maximalement prédictif, notre travail consistait à concevoir un outil efficace pour réaliser ces partitionnements. Pour cela, nous avons conçu un algorithme, défini une relation d'équivalence sur les partitions, ainsi qu'un langage de génération des types de prédicteurs de classes qui peuvent servir à opérer ce partitionnement. Ces prédicteurs peuvent être à la fois ambigus, contextuels, structurés, et pondérés à convenance. Ceci nous a permis de concevoir un programme, `pampres`. Sur la donnée d'une séquence, d'un lexique et d'un nombre maximal de classes, ce programme réalise le partitionnement macimalement prédictif avec une complexité temporelle linéaire avec la taille de la séquence, le nombre de classes, et la taille de l'ensemble des descripteurs de classes ; il s'avère plutôt efficace puisque, par exemple, sur un ensemble de quatre prédicteurs, la construction d'une partition en cent classes d'une séquence de cent mille lettres prend environ quarante secondes, et l'ensemble des partitions en cent classes peut être obtenu en quatre-vingts secondes.

Grâce à un ensemble de menus, l'utilisation de `pampres` est aisée. Il permet de manipuler facilement séquences, partitions, partitionnements, et propose différentes sorties graphiques (en format *postscript*), ainsi que plusieurs fonctions d'évaluation des partitions. Plusieurs de ces fonctions ont été adaptées de critères connus en classification et peuvent servir de critères de détermination du nombre de classes des partitions.

La validité du partitionnement de séquences a été confortée par son comportement. En effet, la conservation des césures et prédicteurs des classes des partitions au cours du partitionnement convainc qu'une structure des séquences est effectivement extraite. Plus il y a de classes, plus la structure de la séquence est exprimée en détails, mais du fait de cette stabilité, la structure à plus grande échelle est toujours présente. Elle est juste cachée par les variations à plus petite échelle. Il ne faut pas oublier que cette révélation de structure est faite selon un ensemble de prédicteurs, et que d'un ensemble de prédicteurs à un autre, elle peut varier, et que sur certains ensembles, la séquence ne fait pas apparaître de structure remarquable.

Il faut alors évaluer si un ensemble de prédicteurs permet de révéler une structure dans une séquence, et dans ce cas à quel niveau du partitionnement cette structure est révélée. Nous avons mis en lumière que certains niveaux du partitionnement sont corrects pour partager la séquence : les moments stables. Une caractéristique importante de ces moments est qu'ils peuvent être déduits de la simple donnée de la fonction de prédiction, ce qui permet de les obtenir sans avoir à construire l'ensemble des k-partitions optimales, pour tous les k. Nous avons adapté un critère d'arrêt – le coude – pour exprimer l'intérêt relatif de ces moments, et ainsi révéler si une séquence, selon l'ensemble de prédicteurs qui a servi au partitionnement, admet un niveau de classes particulier qui soit vraiment révélateur de sa structure. Ce critère est très facile à calculer, ne prend en compte que la donnée de la prédiction, et possède quelques propriétés qui permettent d'estimer si un moment stable, ou des moments stables, sont « aptes » à être déclarés moments d'arrêt.

Pour commencer à juger de l'intérêt de cette approche dans le contexte de la biologie moléculaire, différentes méthodes qui concernent la recherche de zones dans des séquences génétiques ont été présentées. Certaines de ces méthodes peuvent être apparentées à la notre, car elles cherchent à exploiter des biais de proportions dans la constitution des séquences. Une bonne utilisation de ces biais peut nous laisser espérer, à l'avenir, des applications intéressantes du partitionnement maximalement prédictif.

En effet, comme nous l'avons vu dans le cinquième chapitre, il existe de nombreux biais de proportions en nucléotides, oligonucléotides, codons, expressions en acides aminés dans le contenu des séquences génétiques, et plusieurs de ces biais sont utilisés pour l'analyse de séquences génétiques. Nous mêmes en avons utilisés pour expérimenter le partitionnement maximalement prédictif. Bien sûr, il faudra à l'avenir chercher à appliquer le partitionnement maximalement prédictif sur toutes sortes de biais observés dans les séquences génétiques, d'autant plus que le partitionnement, parce qu'il partage au mieux selon un biais, fournit une évaluation de ce biais. Cette évaluation peut être faite par un critère d'évolution du partitionnement (comme le coude), ou comme un critère interne à une partition (comme l'interaction ou la fonction « between » de Gower). À l'inverse, si on sait qu'une séquence d'un certain type devrait normalement être assez homogène selon un critère, on peut étudier si dans cette séquence il n'y a pas d'anomalies, c'est-à-dire des segments qui ne répondent pas à ce critère. Ces anomalies pourraient être dues à un « accident », inversion, translocation... de cette séquence, ou à un élément exogène (zones de prophages par exemple).

Il faudra pratiquer ce partitionnement sur d'autres données que des séquences de nucléotides, comme les protéines. Une protéine a généralement une forme – structure tertiaire – stable qui détermine sa fonction. Un problème important dans la recherche biologique actuelle est de parvenir à prédire la structure tertiaire d'une protéine de la simple connaissance de la séquence d'acides aminés – structure primaire. En fait, dans les structures tertiaires des protéines on reconnaît plusieurs formes bien précises (hélice-α, feuillet-β), et la seule connaissance de la répartition de ces formes dans la protéine – structure secondaire – est intéressante. La structure d'une protéine dépend d'énormément de facteurs, aussi bien locaux (acides aminés petits ou gros, hydrophiles ou hydrophobes, polarisés ou neutres...) que globaux (par repliement de la molécule, deux acides aminés éloignés sur la séquence peuvent être proches dans la protéine). Beaucoup de travaux ont porté sur l'expression de zones et de régularités dans les protéines (par exemple [McL77, Ros78, KD82, LRS+86, CKM+87, GBBM87, LVHG+90, KBM+94]). Grâce à sa malléabilité en ce qui concerne les types de régularités exprimés, et grâce au fait que l'on peut intégrer dans un lexique des relations de similarités (positives et négatives) entre objets, il serait sûrement intéressant d'essayer le partitionnement maximalement prédictif sur des protéines.

En amont de la traduction, on peut aussi réaliser des partitionnements sur les gènes d'une séquence. Les gènes sont maintenus en place dans la séquence, mais toutes les parties non-codantes sont omises. À ce moment, on connaît la phase de lecture de tous les gènes, et on peut analyser si des critères tels que l'usage du code, ou des proportions en oligonucléotides en position donnée des codons, font apparaître une structure dans la séquence. Ceci pourrait révéler des modules dans ces séquences, ou bien des zones de translocation, et aider à appréhender une histoire phylogénétique. Il serait aussi possible d'arriver à détecter des biais à grande échelle dans la répartition de l'usage du code ou des acides aminés (au même titre que [McI98, RDV99]).

Il est aussi possible de construire des partitions prédictives sur des alignements de séquences, ce qui extrait un biais consensus aux séquences alignées. Si on a confiance en cet alignement, ce consensus permettrait d'extraire des caractéristiques communes.

À l'opposé, on peut aussi se demander si pratiquer des partitions sur un ensemble de séquences ne permet pas de faciliter l'alignement de ces séquences grâce aux résumés obtenus par partitionnement. Le partitionnement pourrait permettre de comparer plusieurs séquences d'une façon totalement duale d'algorithmes usuels comme BLAST et FASTA qui recherchent des occurences de grands mots. La coopération de ces dualités serait sûrement profitable.

Les biais que présentent les séquences génétiques se cumulent et compliquent énormément leur analyse. Comme nous l'avons brièvement exposé dans le cinquième chapitre, ces séquences sont au centre de plusieurs mécanismes génétiques. Elles font aussi partie de tout un complexe physico-chimique. Elles sont toutes issues d'une histoire phylogénétique. Si chacun de ces facteurs provoque un biais de constitution, on comprend très bien que tous ces biais s'accumulent. Sur la donnée d'un lexique qui exprime un biais, le partitionnement maximalement prédictif cherche à partager une séquence en segments homogènes vis-à-vis de ce biais. Sur une partition ainsi construite, on pourrait modifier l'ensemble de la séquence initiale segment par segment pour détruire le biais observé et pratiquer, avec un nouveau lexique, un nouveau partitionnement. De manière à ne pas modifier la structure de la séquence, cette suppression de biais serait juste une pondération des objets de chaque classe qui correspondent au prédicteur optimal de cette classes. Cependant, ce n'est pas parce que l'on supprime l'expression essentielle d'un biais sur l'ensemble d'une séquence que l'on supprime toute expression de ce biais. Certes, normalement, ce biais devrait alors être moins prononcé qu'auparavant, mais il faut recommencer le partitionnement avec le même lexique jusqu'à ce qu'il ne soit plus « trop » exprimé. Il faut bien sûr donner un sens précis à l'importance de cette expression ; elle est liée au problème de la détermination du nombre de classes.

Une autre possibilité d'évaluation de plusieurs biais sur une séquence serait de pratiquer plusieurs partitionnements selon différents lexiques, et d'étudier l'ensemble de ces résultats. On pourrait estimer la stabilité des bornes des classes en fonction des lexiques un peu comme nous l'avons brièvement fait pour le partitionnement de la séquence mitochondriale de l'homme (page 126), ou bien par l'évolution des prédictions en fonction de pondérations attribuées à des prédicteurs. Ceci nécessite sûrement une étude théorique assez ardue pour évaluer quelle est l'influence des différents lexiques sur un partitionnement et dresser des relations entre ces lexiques.

Une telle étude servirait si on compte intégrer le partitionnement maximalement prédictif dans une approche exploratoire. Partant d'une séquence, on cherche s'il existe des lexiques selon lesquels elle présente une structure nette. Ceci est possible parce que la rapidité du programme `pampres` permet de pratiquer beaucoup de partitionnements et qu'une large gamme de lexiques peut être utilisée. Évidemment, il faut d'abord déterminer s'il est possible, pour une séquence quelconque, qu'il existe un lexique de taille raisonnable[10] qui fasse apparaître une structure. Dans ce cas, il faut tempérer l'obtention d'un quelconque lexique qui exhibe une structure. Ou alors, il faut se fixer une gamme très précise des lexiques parmi lesquels rechercher. Par des méthodes d'apprentissage, l'élaboration de tels lexiques pourrait être automatisée, soit par une approche « naïve » qui cherche seulement sur des critères liés à la prédiction, soit à l'aide de connaissances spécifiques au champ d'étude (comme l'expression de biais dans des types de séquences génétiques). Par souci de rapidité, un tel processus exploratoire pourra se faire sur la seule donnée des partitionnements simples, avec un critère, comme le coude, qui n'a besoin que de la prédiction pour évaluer les différents moments stables susceptibles d'être déclarés niveaux d'arrêt. Si un lexique et un niveau intéressants sont trouvés, un partitionnement complexe pourra alors être effectué.

En ce qui concerne l'étude du partitionnement maximalement prédictif en lui-même, il serait intéressant de pouvoir, à la suite des résultats effectués dans le cas où il n'y a que deux prédicteurs admissibles, trouver des propriétés qui certifient la nature des moments stables dans le cas général. Il serait idéal d'obtenir une méthode qui permette de construire les partitions aux moments stables, sans avoir à passer par les niveaux intermédiaires.

De même, sur la donnée d'un lexique quelconque, il faudrait pouvoir estimer précisément pour quelle valeur d'angle de coude on peut considérer que la

[10]Un lexique constitué de deux mots qui correspondent chacun à une moitié de la séquence extraira une structure de cette séquence, mais ce n'est pas très très intéressant.

séquence admet une structure remarquable, ou bien plusieurs structures remarquables.

Toutes ces méthodes seraient aussi intégrées dans le programme `pampres`. Il faudra aussi que ce programme dispose d'une interface qui permette de visualiser dynamiquement telle ou telle donnée, et de manipuler directement leur représentation visuelle, de manière à rendre ce programme plus convivial.

Annexe A

Menus de pampres

Les critères qui ont guidé la réalisation de l'interface sont de permettre à la fois un accès facile par l'utilisateur à tous les objets qu'il souhaite pouvoir manipuler, ainsi qu'un large éventail de fonctionnalités sur ces objets.

L'interface s'exécute en mode texte, et concerne la gestion des objets dits principaux présentés en 3.1.1. Le menu principal a l'allure suivante :

```
---------------------------------------------
Choisir un objet de travail

Lexique:                                l
Genome:                                 g
Sequence initiale:                      s
Partition simple:                       p
Partitionnement simple:                 a
Ens. de Partitionnements simples:       e
Partition complexe:                     c
Partitionnement complexe:               o
-------
quitter:                                q
```

L'utilisateur communique son objectif par une lettre qui permet de travailler sur tel ou tel type d'objet. Répondre `q` termine l'exécution du programme. Lorsqu'un choix a été fait, il faut désigner par un nom l'objet voulu, pour le créer aussi bien pour y avoir de accès. Un nom d'objet est limité à trente caractères. À chaque fois que le nom d'un objet est demandé, entrer le « `?` » fournit la liste des noms de tous les objets similaires déjà créés. Le nom vide annule la procédure dans laquelle on s'est engagé.

Dans un quelconque menu, la réponse vide provoque la sortie de ce menu, et répondre une lettre non sollicitée ne provoque aucune action.

Lorsque l'on veut afficher le contenu d'un objet à l'écran, il suffit d'entrer le nom vide lorsque le programme demande un nom de fichier dans le processus qui correspond à la proposition :

```
ecriture:         e
```

En outre, ne rien entrer à une question du type :

`Etre ou ne pas etre (e/p)?(e)`

provoque la réponse par défaut, ici `etre`.

A.1 Lexique

La description de la classe `Clexique` qui correspond aux lexiques est en section 3.1.1.2.

La syntaxe du lexique a été présentée dans la partie 2.4. Un lexique peut être entré directement au clavier ou bien à partir d'un fichier après la question :

`fichier ou clavier(f/c)?(*)`

La réponse par défaut signifie que le lexique est `*`. Dans ce cas, de même que lorsqu'un symbole `*` est présent dans le lexique, comme toutes les lettres de la séquence doivent être considérées, la génération de l'ensemble des prédicteurs ne peut se faire qu'à partir de la donnée d'une séquence dont le nom est alors demandé. Dans ce cas, il faut qu'une séquence initiale ait déjà été créée.

À la syntaxe d'un fichier de lexique est ajoutée le règle que le point-virgule marque la fin de fichier. Ceci permet d'ajouter des commentaires dans le fichier après le lexique lui-même.

Une fois un lexique créé, le menu suivant est affiché :

```
reconstruction:         r
ecriture:               e
valeurs sur sequence:   v
```

En fonction de la réponse,

- `r` détruit le lexique entré précédemment et affiche le menu de construction de lexique ;
- `e` écrit le lexique ;
- `v` permet de calculer les prédictions des prédicteurs du lexique sur une séquence entière.

A.2 Génome

La description de la classe `Cgenome` qui correspond aux génomes est en section 3.1.1.

La construction d'un génome se fait à partir du fichier dont le nom est donné en réponse à la question

`nom du fichier d'origine?`

Le programme reconnaît pour les génomes deux extensions de fichiers : `.seq` et `.gbk` (voir A.13).

A.3 Séquence

La description de la classe `Cseq_prim` qui correspond aux séquences d'objets est en section 3.1.1.

Le menu pour construire une séquence est le suivant :

```
fichier:        f
genome:         g
partition:      p
melange:        m
```

En fonction de la réponse,

`f` la séquence sera lue dans un fichier de type `.seq` plus général que celui qui concerne les génomes (voir A.13) ;

`g` la séquence sera construite depuis un génome déjà entré ;

`p` la séquence sera extraite d'une autre séquence par la sélection du segment qui correspond à une classe d'une partition simple ou d'un partitionnement simple ; ces objets doivent déjà être créés ;

`m` la séquence sera générée aléatoirement à partir d'une séquence. Voir la partie A.12 qui traite du mélange des séquences.

Une fois la séquence construite, le menu qui est affiché est

```
reconstruction:         r
destruction:            d
ecriture:               e
valeurs sur lexique:    v
```

Répondre `v` permet de calculer les prédictions des prédicteurs d'un lexique sur cette séquence.

A.4 Partition simple

On appelle partition simple toute partition prédictive. La description de la classe `Csequence` qui correspond aux partitions simples est en section 3.1.1.4.

Les moyens de construire une partition simple sont proposés dans le menu :

```
lissage:     l
extremum:    e
extraction:  x
fichier:     f
```

En fonction de la réponse :

`l` permet de construire la partition par partitionnement maximalement prédictif (voir A.9) ;

`e` permet de construire la partition par l'optimisation d'une fonction sur un ensemble de partitions optimales (voir A.10) ;

`x` permet de construire la partition à partir d'un partitionnement déjà conçu ;

`f` permet de construire la partition à partir de la lecture d'un fichier de type `.part` (voir A.13).

Une fois qu'une partition est construite, parmi le menu

```
reconstruction: r
destruction:    d
calcul:         c
graphique:      g
ecriture:       e
```

choisir

`c` permet de calculer la valeur d'une fonction sur la partition (voir A.11) ;

`g` permet la génération d'un fichier en format *postscript* encapsulé (extension `.eps`) dans lequel seront dessinées les classes prédictives de cette partition et éventuellement les zones intéressantes d'un génome ;

`e` permet d'écrire la partition dans un fichier `.part` (voir A.13).

A.5 Partitionnement simple

On appelle partitionnement simple une suite de partitions prédictives en k classes, pour k allant de 1 jusqu'à un nombre choisi. La description de la classe `Cpartitionnement` qui correspond aux partitions simples est en section 3.1.1.

Un partitionnement peut se construire de trois manières différentes :

```
Construction par:
lissage:        l
extremum:       e
fichier:        f
```

En fonction de la réponse, le programme permet de construire le partitionnement simple

`l` par partitionnement maximalement prédictif (voir A.9) ;

`e` par optimisation d'une fonction sur une suite ensemble de partitions optimales (voir A.10) ;

`f` à partir de la lecture d'un fichier de type `.part` (voir A.13).

Une fois le partitionnement construit, le menu qui apparaît est

```
reconstruction: r
destruction:    d
graphique:      g
courbe:         c
ecriture:       e
extraction:     x
```

Répondre

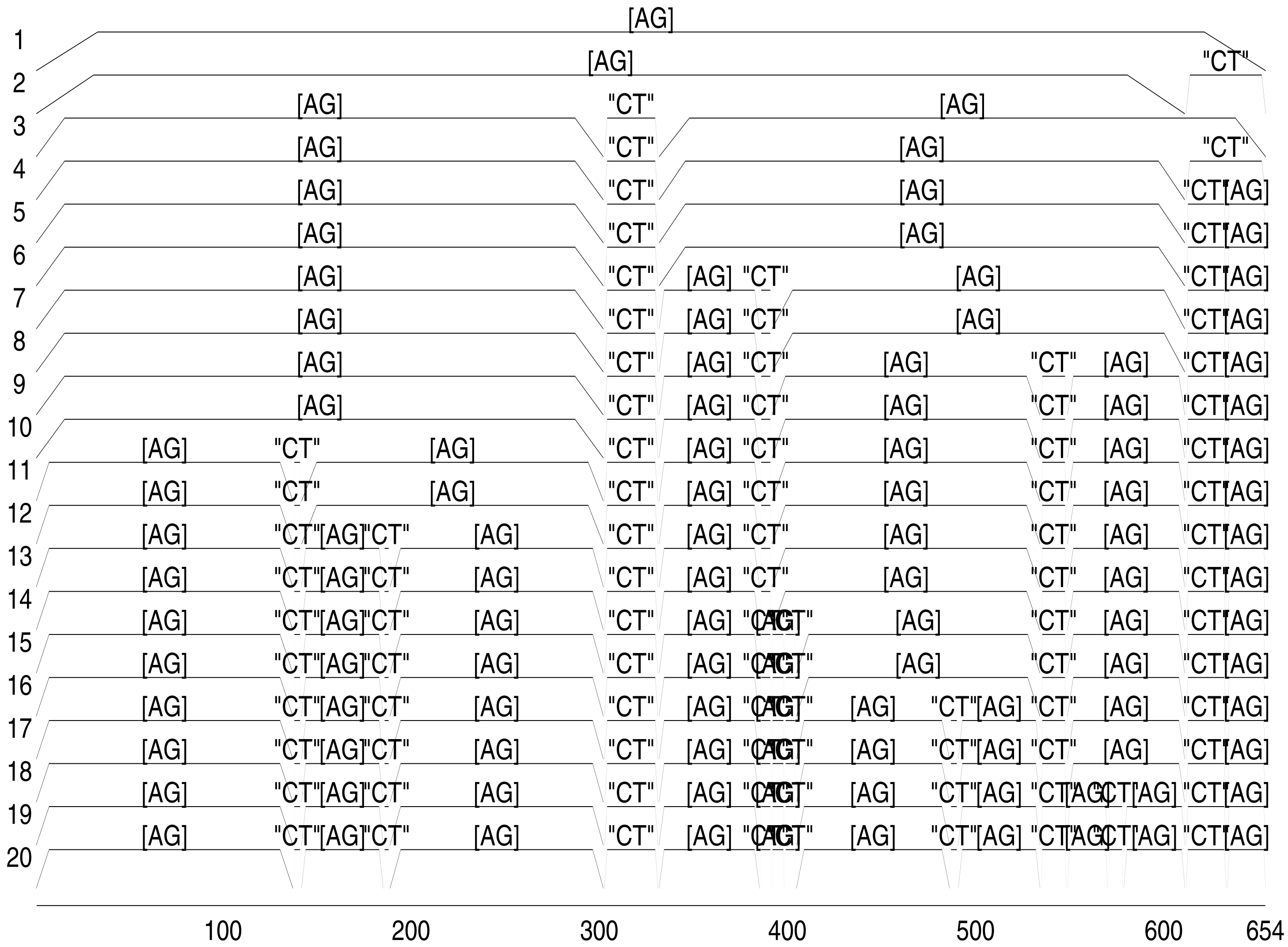


FIG. A.1 – Le trait horizontal gradué représente la séquence. De haut en bas sont dessinées les k-partitions pour k allant de 1 à 20. Chaque arc correspond à une classe. Le prédicteur de chaque classe prédictive est écrit par mot surmontant l'arc de la classe concernée (sans facteur par souci de clarté). Dans le cas de prédicteurs structurés, le mot est en phase avec le début de la classe, c'est-à-dire que s'il est écrit `ABC`, ceci signifie que le premier objet de la classe est prédit par `A`, le deuxième par `B`, etc.

`g` permet la génération d'un fichier en format *postscript* encapsulé (extension `.eps`) dans lequel figureront les partitions du partitionnement. Le dessin produit ressemble à la figure A.1. La question est posée si les prédicteurs des classes doivent être affichés. Il se peut que le programme demande également si on veut en outre que les zones intéressantes d'un génome soient dessinées, et dans ce cas si les noms de ces zones doivent être écrits. Si le nombre de niveaux du partitionnement est supérieur à 20, seuls les niveaux qui correspondent aux moments stables (voir page 86) sont affichés.

`c` permet de générer un fichier *postscript* qui affiche la courbe d'une fonction des partitions successives. Le programme demande les extrema du nombre de classes entre lesquels sera dessinée la courbe. Répondre zéro fait que la borne en question sera l'extrême possible. On peut tracer le graphe d'une fonction présentée en A.11, ou bien, en répondant `o` à la question `difference(o/n)?(n)`, celle de la variation d'une fonction des partitions simples (voir A.11) ou celle de la fonction coude (voir 1.4.3.6 et 4.3.2).

`e` permet d'écire le partitionnement dans un fichier de type `.part` (voir A.13).

`x` permet d'accéder à une partition donnée du partitionnement, accessible par sa place dans ce dernier.

A.6 Ensemble de partitionnements

Un ensemble de partitionnements est une population de partitionnements simples créés par partitionnement maximalement prédictif sur une population de séquences générée par mélange d'une séquence initiale. La description de la classe `Cpopulation` qui correspond aux ensembles de partitionnements simples est en section 3.1.1.

La construction d'un tel objet se fait donc à partir d'une séquence et d'un lexique qui doivent déjà avoir été construits dans le programme. En outre, le programme demande le nombre de classes des partitionnements, le nombre d'individus de la population, et quel type de mélange pratiquer sur la séquence (voir A.12). Ensuite, les partitionnements simples sont construits comme présenté en A.9.

Une fois un ensemble de partitionnements simples construit, par ce menu

```
reconstruction: r
destruction:    d
extraction:     x
courbe:         c
```

on peut

`x` extraire un partitionnement simple accessible par son numéro dans population des partitionnements ;

`c` tracer des courbes, en format postscript encapsulé, qui correspondent à la prédiction des membres de l'ensemble. Trois courbes sont tracées : la moyenne, les 5-percentiles supérieurs et les 5-percentiles inférieurs. Un partitionnement simple (qui doit déjà être construit) peut être donné. Ou bien sa courbe pour la fonction va être tracée en plus de celles de l'ensemble, ou bien ce sont les différences entre cette fonction et celles des membres de l'ensemble de partitionnements qui seront tracées.

A.7 Partition complexe

Il s'agit d'une représentation d'un ensemble de partitions optimales ex æquo. La description de la classe `Cseq_equ` qui correspond aux partitions complexes est en section 3.1.1.4.

La construction d'une partition complexe se fait depuis ce menu

```
Construction par:
lissage:                l
exhaustivite:           e
extraction:             x
partition simple:       p
fichier:                f
```

En fonction de la réponse, la construction de la partition complexe se fait

`l` par partitionnement maximalement prédictif (voir A.9) ;

`e` à partir d'une autre partition complexe, en construisant l'ensemble des représentants des classes d'équivalence dont sont membres les partitions de cette partition complexe (voir 2.3.1) ;

`x` par copie d'une partition complexe issue d'un partitionnement complexe et accessible par sa position dans celui-ci ;

`p` par copie d'une partition simple ;

`f` par la lecture d'un fichier `.part` (voir A.13).

Une fois la partition complexe construite, le menu suivant apparaît :

```
reconstruction: r
destruction:    d
calcul:         c
ecriture:       e
graphique:      g
```

Selon la réponse,

`c` calcule le résultat d'une fonction sur la partition complexe (voir A.11) ;

`e` écrit la partition complexe sous le format `.part` (voir A.13) ;

`g` génère un fichier en format postscript encapsulé (extension `.eps`) dans lequel figurera le graphe des partitions ex æquo qui constituent cette partition complexe. La figure produite a l'allure de la figure A.2.

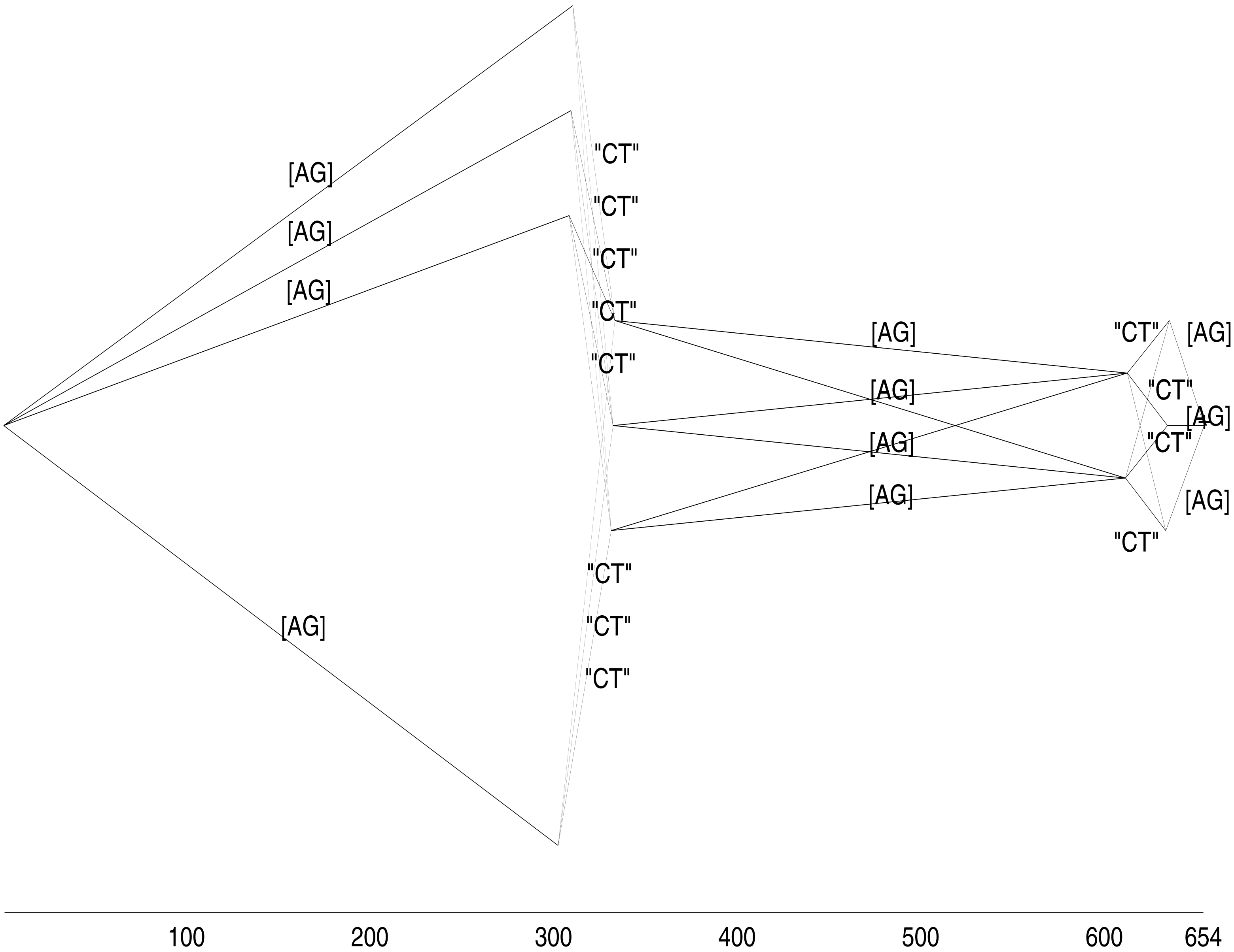


FIG. A.2 – La séquence est représentée par le trait gradué horizontal. Une classe prédictive est représentée par un segment du graphe, le mot situé auprès d'elle est son prédicteur optimal, calé de manière à ce que la première lettre du mot prédise la première position de la classe. La jonction entre deux classes se situe à la verticale de sa position dans la séquence. Une partition prédictive s'obtient en parcourant le graphe dans un même sens, en choisissant successivement aux nœuds du graphe de quelles classes prédictives elle se compose.

A.8 Partitionnement complexe

Il s'agit d'une suite de partitions complexes en k classes, pour k allant de 1 à un nombre choisi. La description de la classe `Cparti_comp` qui correspond aux partitions simples est en section 3.1.1.

Le menu de construction est le suivant :

```
Construction par:
lissage:        l
exhaustivite:   e
fichier:        f
```

Et correspond aux possibilités suivantes :

`l` par partitionnement maximalement prédictif (voir A.9) ;

`e` à partir d'un autre partitionnement complexe, par la construction de l'ensemble des réprésentants des classes d'équivalence dont sont membres les partitions des partitions complexes d'un autre partitionnement complexe (voir 2.3.1) ;

`f` par la lecture d'un fichier `.part` (voir A.13).

Après la construction, les opérations possibles sont

```
reconstruction: r
destruction:    d
ecriture:       e
courbe:         c
extraction:     x
```

`c` tracer une courbe qui, en format postscript encapsulé, affiche la courbe d'une fonction des partitions complexes successives. Le programme demande les extrema du nombre de classes entre lesquels sera dessinée la courbe. Répondre `0` fait que la borne en question sera l'extrême possible. On peut tracer le graphe d'une des fonctions présentées en A.11, ou bien, en répondant `o` à la question `difference(o/n)?(n)`, une courbe des différences entre partitions complexes successives. Les possibilités sont alors de tracer

```
fonction:               f
coude:                  c
nouvelles cesures:      n
```

`f` la variation d'une fonction sur les partitions complexes (voir A.11) ;

`c` la fonction coude (voir 1.4.3.6 et 4.3.2) ;

`n` le nombre de césures créées entre deux nombres de classes successifs.

`x` extraire une partition complexe accessible par son numéro.

A.9 Partitionnement maximalement prédictif

Bien sûr, le partitionnement maximalement prédictif ne peut se faire qu'à partir d'une séquence et d'un lexique, qui sont demandés, et qui doivent déjà être entrés dans le programme. En outre, un nombre maximum de classes k est demandé.

• **Partitions et partitionnements simples**

Le menu

```
Critere de selection des partitions:
quelconque:     q
fonction:       f
```

permet, en fonction de la réponse,

`q` d'opérer un partitionnement maximalement prédictif dans lequel juste une partition est gérée. Ainsi, pour chaque k, la k-partition obtenue sera une représentante quelconque des k-partitions optimales ;

`f` d'opérer un partitionnement maximalement prédictif dans lequel toutes les partitions optimales sont gérées. Pour chaque k, la k-partition obtenue sera parmi les k-partitions celle qui optimise une certaine fonction (voir A.10).

• **Partitions et partitionnements complexes**

La question

```
lissage complet ou par classes d'equivalence(c/e)?(e)
```

permet, en entrant `c`, de faire le partitionnement qui génère toutes les partitions ex æquo, ou bien, en entrant `e`, de faire celui qui ne génère que les classes d'équivalence (voir 2.3.1).

A.10 Choix parmi les ex æquo

Le choix parmi un ensemble de partitions optimales se fait par l'optimisation d'une fonction. Les fonctions disponibles sont celles présentées dans la section A.11, hormis bien entendu la prédiction. Si deux partitions optimisent ce critère, une quelconque des deux est choisie. Le critère d'optimisation est au choix de l'utilisateur, maximisation ou minimisation.

A.11 Calculs sur les partitions

• **Partitions et partitionnements simples**

Le menu suivant :

```
Quelle fonction?
--prediction:   p
--compacite:    c
--between:      b
--interaction:  i
--Gower:        g
```

permet de calculer les valeurs des fonctions suivantes sur les partitions simples :

`p` la prédiction ;

`c` la moyenne des densités de prédiction sur les classes (voir 1.4.3.4) ;

`b` la fonction between de Gower définie en 1.4.2. Pour pouvoir faire ce calcul, il faut entrer le nom d'une séquence déjà représentée dans le programme ;

`i` la fonction d'interaction, c'est-à-dire la fonction between de Gower réduite, pour chaque classe, à ses classes voisines, selon la formule donnée en 1.4.3.5. Pour pouvoir faire ce calcul, il faut entrer le nom d'une séquence déjà en programme ;

`g` La prédiction moins la fonction calculée dans la fonction de Gower réduite au voisinage (choix `i` ci-dessus).

• **Partitions et partitionnements complexes**

Le menu suivant

```
Quelle fonction?
-prediction:                p
-moyenne des differences
        de prediction:      m
-variance des bornes:       v
-nombre de cesures:         C
-moyenne du
        nombre de cesures:  c
-nombre de segments:        S
-moyenne du
        nombre de segments: s
```

permet de calculer les valeurs des fonctions suivantes sur les partitions complexes :

`p` la prédiction ;

`m` la moyenne, sur la taille de la séquence, du nombre de prédictions différentes par objet ;

`v` la somme, sur les paquets de classes (voir A.13), de la variance des positions des césures des classes de chaque paquet ;

`C` le nombre de césures du graphe de la partition complexe ;

`c` le nombre de césures du graphe de la partition complexe divisé par le nombre de paquets de cette partition complexe ;

`S` le nombre de classes ;

`s` le nombre de classes divisé par le nombre de paquets de cette partition complexe.

A.12 Mélanges

Les mélanges sont effectués sur les lettres des séquences, et non sur les objets eux-mêmes, si bien que même si les objets d'une séquence ont des poids différents de 1, tous les objets sont de poids 1 dans la séquence qui est le résultat de ce mélange.

En outre, le mélange peut être fait sur la donnée d'une partition simple, c'est-à-dire que le mélange est fait au sein de chaque classe de cette partition, qui est alors considérée comme une séquence à part entière, puis les classes qui sont le résultat du mélange fait *indépendamment* sur chaque classe sont concaténées pour construire la séquence voulue, de même taille que la séquence initiale. Dans ce cas, il faut que la partition simple soit déjà créée dans le programme.

Les trois méthodes de mélange de séquences de lettres accessibles sont

```
melange pur:    p
mono brassage:  m
bi brassage:    b
```

`p` un mélange uniforme de toutes les lettres de la séquence, avec leur nombre exact d'occurences.

`m` un mélange uniforme des lettres de la séquence de telle manière que le χ^2 entre le nombre de d'occurences de chaque lettre sur la nouvelle séquence et la séquence initiale pour l'ensemble des lettres soit inférieur à 0.01.

`b` un mélange uniforme des mots de deux lettres de la séquence de telle manière que le χ^2 entre le nombre de d'occurences de chaque mot dilitère sur la nouvelle séquence et la séquence initiale pour l'ensemble de ces mots dilitères soit inférieur à 0.01.

Si le test de χ^2 persiste à échouer après cent tentatives de mélange, l'opération est abandonnée.

A.13 Formats des fichiers utilisés

A.13.1 Extension `.gbk`

L'extension `.gbk` correspond au format des fichiers de la banque de données GenBank [BBL+99]. Cette banque permet l'accès à une énorme quantité de séquences génétiques publiques annotées[1]. Les informations concernant cette banque sont accessibles à l'adresse
`http://www.ncbi.nlm.nih.gov/Genbank/GenbankOverview.html`

A.13.2 Extension `.seq`

L'extension `.seq` est propre au programme et correspond à l'écriture des objets génome et séquence en sortie du programme. Pour un génome, un fichier `.seq` s'écrit de cette manière :

[1]En août 1999, environ 3.4×10^9 bases dans 4.61×10^6 séquences.

```
654 TAACCCTAAC CCTAACCCTG ACCCTAACCC TAACCCTAAC CCTAACCCTA
ACCAGTACAC GCGTACACGT ACAAGCACCC GTACCCCCAG TATACTTGGA
CACCCGTACT CAGTTATCCT TTTTATTAGT GTACCCGCCT CTTGCACGCA
TGCCACAGTT CTTCAGCAGA AGAACACGCA CAATGCTCTT TGATAAACGT
GCGGACATGA AAAAAAGGGA AAAACGCAGC TACGTGTGCT GTCGTTGGTT
ACACAGCGTC AAGCCGCGTC GGTGTACCAA AGAGGAGGTG ACCCATCGAG
TACTCGCACC CTCTAGCTCT CCTTTTCTGC CTCGTATTAT ACACGTTGAT
CGGAAAACAG GGTAGGCACT AGGGTGGTGG TTGCGTCTCT CTTGCAGATT
CTTCTTGTAG TTGTCGCCGC GTCTCGAATT GTGTGCTTGC GAAAATCAAG
TCGTGTGTGT TTGCACGCCT TCCGTGATCG CTGTGGCTAC TGCGTGACTC
GGCTGCAGAC ACCCCTTTCC GTCGATTATA TATACCTTTC TTTCGCCCTT
GTTTTGTGCA TTTATTTTTG TTTCTACCTG GATCGCTTTT TATTTATTTT
GTACATCCGC GTCTCTTCCT GTCCCTCACT CCCTCGGTGC GTCTGGTGGT
GGTT.
gene1(50..100) gene2(590..400)
```

Le premier nombre est la longueur de la séquence. Puis suit la séquence, qui ne contient que les lettres `A`, `C`, `G`, `T`, puis, après un point (`.`) – facultatif, mais c'est plus sûr si jamais le premier chiffre n'est pas vraiment la longueur de la séquence...– les parties remarquables de cette séquence, avec leur sens. Si seule la partie qui va de la lettre 100 à la lettre 222 nous intéresse, il suffit de changer le début en mettant `100 222` au lieu de `654`. Si le début est `(100) 222`, ceci signifie que les 123 *premières* lettres sont prises en compte, mais que la séquence va en réalité de la position 100 à la position 222.

De façon plus générale, pour construire des séquences d'objets, un fichier `.seq` admet que des facteurs soient assignés aux lettres entrées. Par exemple :

```
57 V(0.91)L(1.22)S(-0.55)P(-0.49)A(-0.4)D(-1.31)K(-0.67)
T(-0.28)N(-0.92)V(0.91)K(-0.67)A(-0.4)A(-0.4)W(0.5)
G(-0.67)K(-0.67)V(0.91)G(-0.67)A(-0.4)H(-0.64)A(-0.4)
G(-0.67)E(-1.22)Y(1.67)G(-0.67)A(-0.4)E(-1.22)A(-0.4)
L(1.22)P(-0.49)A(-0.4)E(-1.22)F(1.92)T(-0.28)P(-0.49)
A(-0.4)V(0.91)H(-0.64)A(-0.4)S(-0.55)L(1.22)D(-1.31)
K(-0.67)F(1.92)L(1.22)A(-0.4)S(-0.55)V(0.91)S(-0.55)
T(-0.28)V(0.91)L(1.22)T(-0.28)S(-0.55)K(-0.67)Y(1.67)R(-0.59)
```

ce qui signifie que chaque objet de la séquence aura un poids semblable à la valeur entre les parenthèses qui le suivent (l'absence de parenthèses après une lettre attribue le poids 1).

A.13.3 Extension `.part`

Cette extension correspond aux partitions simples et partitionnements simples, ainsi qu'aux partitions complexes et aux partitionnements complexes.

• **Partitions et partitionnements simples**

Une partition prédictive s'écrit de cette manière :

```
<1-302>144[AG] XXX <303-331>16"CT"(2) XXX <332-611>110[AG] XXX
<612-633>12"CT"(2) XXX <634-654>10[AG] ---->292
```

Les classes prédictives sont décrites entre les `XXX`. Les deux nombres entre `<` et `>` correspondent aux positions extrêmes de la classe. La valeur qui suit `>` est la prédiction sur cette classe du prédicteur écrit après. Ainsi, `<1-302>144[AG]` correspond à une classe prédictive qui va de l'objet en position 1 jusqu'à l'objet en position 302, tous deux inclus, sur laquelle le prédicteur est `[AG]` et la prédiction de ce prédicteur vaut 144. Dans le cas où le prédicteur de la classe prédictive est un motif, par exemple `ABC`, l'ecriture `<1-302>144ABC` signifie que le motif est calé de telle manière que `A` prédit l'objet en position 1, `B` celui en position 2, et ainsi de suite.

Après la flèche `---->` est la valeur de la prédiction sur cette partition.

Un partitionnement prédictif s'écrit comme une suite de partitions :

```
<1-654>271[AG] ---->271
<1-611>259[AG] XXX <612-654>14"CT"(2) ---->273
<1-302>144[AG] XXX <303-331>16"CT"(2) XXX
<332-654>122[AG] ---->282
<1-302>144[AG] XXX <303-331>16"CT"(2) XXX <332-611>110[AG] XXX
<612-654>14"CT"(2) ---->284
<1-302>144[AG] XXX <303-331>16"CT"(2) XXX <332-611>110[AG] XXX
<612-633>12"CT"(2) XXX <634-654>10[AG] ---->292
```

• **Partitions et partitionnements complexes**

Une partition complexe en 5 classes est écrite ainsi :

```
<1-310>146[AG] | <1-309>146[AG] | <1-308>146[AG] | <1-302>144[AG]
XXX  <311-333>14"CT"(2) | <310-333>14"CT"(2) | <309-333>14"CT"(2)
| <303-333>16"CT"(2) | <311-332>14"CT"(2) | <310-332>14"CT"(2) |
<309-332>14"CT"(2) | <303-332>16"CT"(2) | <311-331>14"CT"(2) |
<310-331>14"CT"(2) | <309-331>14"CT"(2) | <303-331>16"CT"(2) XXX
<334-612>110[AG] | <333-612>110[AG] | <332-612>110[AG] |
<334-611>110[AG] | <333-611>110[AG] | <332-611>110[AG] XXX
<613-633>12"CT"(2) | <612-633>12"CT"(2) | <613-634>12"CT"(2) |
<612-634>12"CT"(2) | <613-635>12"CT"(2) | <612-635>12"CT"(2) XXX
<634-654>10[AG] | <635-654>10[AG] | <636-654>10[AG] ---->292
```

où chaque classe est écrite comme cela a été présenté ci-dessus. Les classes avec les plus petites positions de l'ensemble des 5-partitions sont écrites avant le premier `XXX`, et séparées par `|`. L'ensemble de ces classes est le premier **paquet** de classes de la partition complexe. Toutes les classes qui, dans l'ordre de leurs positions, succèdent à une première classe dans l'ensemble des 5-partitions optimales sont ensuite écrites (ce qui forme le deuxième paquet), et ainsi de suite. Une 5-partition optimale peut être reconstruite en prenant successivement dans ces paquets des classes adjacentes. Comme les classes stockées sont extraites de 5-partitions optimales, on est assuré de reconstruire ainsi une 5-partition optimale.

Un partitionnement de tels ensembles de partitions optimales est, de façon similaire à l'écriture d'un partitionnement, écrit comme une suite de telles écritures de partitions complexes.

Bibliographie

[AdCB+96] A. Arneodo, Y. d'Aubenton Carafa, E. Bacry, P.V. Graves, J.F. Muzy, et C. Thermes. Wavelet based fractal analysis of DNA sequences. *Physica D 96*, pages 291–320, 1996.

[AGM+90] S.F. Altschul, W. Gich, W. Miller, E.W. Myers, et D.J. Lipman. Basic local alignment search tool. *J. Mol. Biol.*, 215 :403–410, 1990.

[AK90] S.G. Andersson et C.G. Kurland. Codon preferences in free-living microorganisms. *Microbiol Rev*, 54(2) :198–210, 1990.

[Arc84] J.W. Archie. A new look at thepredictive value of numerical classifications. *Systematic Zoology*, 33 :30–51, 1984.

[BBL+99] D.A. Benson, M.S. Boguski, D.J. Lipman, B.F.F. Ouellette, B.A. Rapp, et D.L. Wheeler. GenBank. *Nucleic Acids Research*, 27(1) :12–17, 1999.

[BCK92] C. Burge, A.M. Campbell, et S. Karlin. Over- and under-representation of short oligonucleotides in DNA sequences. *Proc. Natl. Acad. Sci. USA*, 89 :1358–1362, 1992.

[BD82] T.A. Bailey et R. Dubes. Cluster validity profiles. *Pattern Recognition*, 15(2) :61–83, 1982.

[BD90] P. Bertrand et E. Diday. Une généralisation des arbres hiérarchiques : les représentations pyramidales. *Revue de Statistique Appliquée*, 38(3) :53–78, 1990.

[BF98] V. Batagelj et A. Ferligoj. Constrained Clustering Problems. In A. Rizzi, M. Vichi, et H. Bock, éditeurs, *Advances in Data Science and Classification*, pages 137–144. IFCS, Springer Verlag, July 1998.

[BH82] J.L. Bennetzen et B.D. Hall. Codon selection in yeast. *The Journal of Biological Chemistry*, 257(6) :3026–3031, 1982.

[BJEG98] A. Brazma, I. Jonassen, I. Eidhammer, et D. Gilbert. Approaches to the automatic discovery of patterns in biosequences. *Journal of Computational Biology*, 5(2) :279–305, 1998.

[BKČK94] V. Batagelj, S. Korenjak-Černe, et S. Klavžar. Dynamic programming and convex clustering. *Algorithmica*, 11 :93–103, 1994.

[BM93] M. BORODOVSKY et J. MCININCH. GENMARK : Parallel gene recognition for both DNA strands. *Computers Chem.*, 17(2) :123–133, 1993.

[BM98] G. BEL MUFTI. *Validation d'une classe par estimation de sa stabilité*. Thèse, Paris IX–Dauphine, 1998.

[Boc85] H.H. BOCK. On some significance tests in cluster analysis. *Journal of Classification*, 2 :77–108, 1985.

[BW77] T.R. BEMENT et M.S. WATERMAN. Locating maximum variance segments in sequential data. *Mathematical Geology*, 9(1) :55–61, 1977.

[CB76] N. CHRISTOFIDES et P. BROOKER. The optimal partitioning of graphs. *SIAM Jounal of Applied Mathematics*, 30(1) :55–69, January 1976.

[CG78] C.J. CASELDINE et A.D. GORDON. Numerical analysis of surface pollen spectra from Bankhead Moss, Fife. *New Phytologist*, 80 :435–453, 1978.

[CH74] T. CALIŃSKY et J. HARABASZ. A dendrite method for cluster analysis. *Communications in statistics*, 3(1) :1–27, 1974.

[Chu89] G.A. CHURCHILL. Stochastic models for heterogenous DNA sequences. *Bulletin of Mathematical Biology*, 51(1) :79–94, 1989.

[Chu92] G.A. CHURCHILL. Hidden Markov chains and the analysis of genome structure. *Computers Chem.*, 16(2) :107–115, 1992.

[CKM+87] J.L. CORNETTE, B.C. KEMP, H. MARGALIT, J.L. SPOUGE, J.A. BERZOFSKY, et C. DELISI. Hydrophobicity scales and computational techniques for detecting amphiâthic structures in proteins. *J. Mol. Biol.*, 195 :659–685, 1987.

[Col84] D.H. COLLESS. A method for hierarchical clustering based on predictivity. *Systematic Zoology*, 33(1) :64–68, 1984.

[Col86] D.H. COLLESS. *L'ordre et la diversité du vivant*, chapitre Classification et information, pages 121–142. Fayard, 1986.

[Dal85] M.B. DALE. On the comparison of conceptual clustering and numerical taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(2) :241–244, 1985.

[DEKM98] R. DURBIN, S. EDDY, A. KROGH, et G. MITCHISON. *Biological sequence analysis*. Cambridge University Press, 1998.

[DM84] W.S. DESARBO et V. MAHAJAN. Constrained classification : the use of a priori information in cluster analysis. *Psychometrika*, 49(2) :187–215, June 1984.

[ECS65] A.W.F. EDWARDS et L.L. CAVALLI-SFORZA. A method for cluster analysis. *Biometrics*, 21 :362–375, June 1965.

[FB82] A. FERLIGOJ et V. BATAGELJ. Clustering with relational constraints. *Psychometrika*, 47(4) :413–426, December 1982.

[Fic82] J.W. Fickett. Recognition of protein coding region in DNA sequences. *Nucleic Acids Research*, 10(17) :5303–5318, 1982.

[Fic84] J.W. Ficket. Fast optimal alignment. *Nucleic Acids research*, 12 :175–179, 1984.

[Fis58] W.D. Fisher. On grouping for maximal homogeneity. *Journal of the American Statistical Association*, 53 :789–798, 1958.

[Fis87] D.H. Fisher. Conceptual clustering, learning from examples, and inference. In *International Workshop on Machine Learning*, pages 38–49, Irvine, California, June 1987.

[FK97] H. Frigui et R. Krishnapuram. Clustering by competitive agglomeration. *Pattern Recognition*, 30(7) :1109–1119, 1997.

[FL86] D. Fisher et P. Langley. Conceptual clustering and its relation to numerical taxonomy. *Artificial Intelligence and Statistics*, 1986.

[FVN71] L. Fisher et J. Van Ness. Admissible clustering procedures. *Biometrika*, 58(1) :91–104, 1971.

[GA99] O. Gascuel et J.M. Alain. Classification par arbre : la forme de l'arbre inféré dépend du schéma algorithmique retenu. In *Actes des septièmes journées de la Société Francophone de Classification*, pages 239–243, Nancy, 1999.

[GAM82] J. Gabarro-Arpa et F. Michel. The hierarchical approach to the DNA stability problem. I. - Patterns in non-equilibrium denaturation and renaturation. *Biochimie*, 64 :99–112, 1982.

[GB72] A.D. Gordon et H.J.B. Birks. Numerical methods in Quaternary palæoecology I. Zonation of pollen diagrams. *New Phytologist*, 71 :961–979, 1972.

[GB74] A.D. Gordon et H.J.B. Birks. Numerical methods in Quaternary palæoecology II. Comparison of pollen diagrams. *New Phytologist*, 73 :221–249, 1974.

[GBBM87] C. Gaboriaud, V. Bissery, T. Benchetrit, et J.P. Mornon. Hydrophobic cluster analysis : an efficient new way to compare and analyse amino acid sequences. *FEBS Letters*, 224(1) :149–155, 1987.

[GF85] A.D. Gordon et C.R. Finden. Classification of spatially-located data. *Computational Statistics Quarterly*, 4 :315–328, 1985.

[GGG80] R. Grantham, C. Gautier, et M. Gouy. Codon frequencies in 119 individual genes confirm consistent choices of degenerate bases according to genome type. *Nucleic Acids Research*, 8(9) :1893–1912, 1980.

[GGG+81] R. Grantham, C. Gautier, M. Gouy, M. Jacobzone, et R. Mercier. Codon catalog usage is a genome strategy modulated for gene expressivity. *Nucleic Acids Research*, 9(1) :r43–r74, 1981.

[GH61] R.E. Gomory et T.C. Hu. Multi-terminal network flows. *J. Soc. Indust. Appl. Math.*, 9(4) :551–570, December 1961.

[GIL37] J.S.L. GILMOUR. A taxonomic problem. *Nature*, 134 :1040–1042, 1937.

[GIL40] J.S.L. GILMOUR. Taxonomy and phylosophy. In HUXLEY J.S., éditeur, *The New Systematics*, pages 461–474. Clarendon Press, Oxford, 1940.

[GIL51] J.S.L. GILMOUR. The development of taxonomic theory since 1851. *Nature*, 168 :400–402, 1951.

[GKL77] R. GNANADESIKAN, J.R. KETTERING, et J.M. LANDWEHR. Interpreting and assessing the results of cluster analyses. *Bulletin of the International Statistical Institute*, 47(2) :451–463, 1977.

[GOR96A] A.D. GORDON. Cluster Validation. In C. HAYASHI, N. OHSUMI, K. YAJIMA, Y. TANAKA, H.H. BOCK, et Y. BABA, éditeurs, *Studies in Classification, Data Analysis, and Knowledge Organization : Data Science, Classification, and Related Methods*, pages 22–39, Kobe, March 1996. IFCS, Springer-Verlag. http ://www-solar.dcs.st-and.ac.uk/∼allan/.

[GOR96B] A.D. GORDON. How many clusters ? An investigation of five procedures for detecting nested cluster structure. In C. HAYASHI, N. OHSUMI, K. YAJIMA, Y. TANAKA, H.H. BOCK, et Y. BABA, éditeurs, *Studies in Classification, Data Analysis, and Knowledge Organization : Data Science, Classification, and Related Methods*, pages 109–116, Kobe, March 1996. IFCS, Springer-Verlag.

[GOR96C] A.D. GORDON. A survey of constrained classification. *Computational Statistics and Data Analysis*, 21 :17–29, 1996.

[GOW74] J.C. GOWER. Maximal predictive classification. *Biometrics*, 30 :643–654, December 1974.

[GRI98] A. GRIGORIEV. Analyzing genomes with cumulative skew diagrams. *Nucleic Acids Research*, 26(10) :2286–2290, 1998.

[GS69] K.R. GABRIEL et R.R. SOKAL. A new statistical approach to geographic variation analysis. *Systematic Zoology*, 18 :259–278, 1969.

[GV99] L. GUÉGUEN et R. VIGNES. Descriptions structurées en classification sous contraintes internes. In *Actes des septièmes journées de la Société Francophone de Classification*, pages 255–260, Nancy, 1999.

[GVL98] L. GUÉGUEN, R. VIGNES, et J. LEBBE. Maximal predictive clustering with order constraint : a linear and optimal algorithm. In A. RIZZI, M. VICHI, et H. BOCK, éditeurs, *Advances in Data Science and Classification*, pages 137–144. IFCS, Springer Verlag, July 1998.

[HBF92] D.G. HIGGINS, A.J. BLEASBY, et R. FUCHS. CLUSTAL V : improved software for multiple sequence alignment. *Comput. Appl. Biosci.*, 8 :189–191, 1992.

[HD75] R.M. HARALICK et I. DINSTEIN. A spatial clustering procedure for multi-image data. *IEEE Transactions on Circuits and Systems*, 22(5) :440–450, May 1975.

[HD99] A. Hardy et J.F. Deschamps. Apport du critère des hypervolumes à la validation en classification. In *Actes des septièmes journées de la Société Francophone de Classification*, pages 201–207, Nancy, 1999.

[HM73] D.M. Hawkins et D.F. Merriam. Optimal zonation of digitized sequential data. *Mathematical Geology*, 5(4) :389–395, 1973.

[HS76] L. Hubert et J. Schultz. Quadratic assignement as a general data analysis strategy. *British Journal of Mathematical and Statistical Psychology*, 29 :190–241, 1976.

[HS83] R.I.C. Hansell et P.H.A. Sneath. Algorithms optimizing the taxonomic information of classifications. In J. Felsenstein, éditeur, *Numerical Taxonomy*, volume G1 de *NATO ASI Series*, pages 97–101. Springer-Verlag Berlin Heidelberg, 1983.

[HSY85] F.K. Hwang, J. Sun, et E.Y. Yao. Optimal set partitioning. *SIAM J. Alg. Disc. Meth.*, 6(1) :163–170, January 1985.

[Hub74] L.J. Hubert. Some applications of graph theory to clustering. *Psychometrika*, 39(3), 1974.

[Hwa81] F.K. Hwang. Optimal Partitions. *Journal of Optimization Theory and Applications*, 34(1) :1–10, May 1981.

[Jen68] R.E. Jensen. A dynamic programming algorithm for cluster analysis. *J. Op. Res. Soc. Am.*, 7 :1034–1057, 1968.

[JS68] N. Jardine et R. Sibson. The construction of hierarchic and non-hierarchic classifications. *The Computer Journal*, 11 :177–184, 1968.

[KBM+94] A. Krogh, M. Brown, I.S. Mian, K. Sjölander, et D. Haussler. Hidden Markov models in computational biology. Applications to protein modeling. *J. Mol. Biol.*, 235 :1501–1531, 1994.

[KC94] S. Karlin et L.R. Cardon. Computational DNA sequence analysis. *Annu. Rev. Microbiol.*, 48 :619–654, 1994.

[KD82] J. Kyte et R.F. Doolittle. A simple method for displaying the hydropathic character of a protein. *J. Mol. Biol.*, 157 :105–132, 1982.

[KM91] S. Karlin et C. Macken. Some statistical problems in the assessment of inhomogeneities of DNA sequence data. *Journal of the American Statistical Association*, 86(413) :27–35, 1991.

[KMD94] A. Krogh, I.S. Mian, et Haussler D. A hidden Markov model that finds genes in *E.coli* DNA. *Nucleic Acids Research*, 22(22) :4768–4778, 1994.

[LDL85] P. Legendre, S. Dallot, et L. Legendre. Succession of species within a community : chronological clustering, with applications to marine and freshwater zooplankton. *The American Naturalist*, 125(2) :257–288, February 1985.

[Lob96a] J.R. Lobry. Asymmetric substitution patterns in the two DNA strands of bacteria. *Mol. Biol. Evol.*, 13(5) :660–665, 1996.

[Lob96b] J.R. Lobry. Origin of Replication of *Mycoplasma genitalium*. *Science*, 272 :745–746, 1996.

[LRS+86] A.M. Liquori, A. Ripamonti, C. Sadun, S. Ottani, et D. Braga. Pattern recognition of sequence similarities in globular proteins by Fourier analysis : A novel approach to molecular evolution. *J. Mol. Evol.*, 23 :80–87, 1986.

[LV94] J. Lebbe et R. Vignes. Classification maximalement prédictive sous contrainte d'ordre total. In *Actes de la société française de classification*, pages 127–128, Tours, 1994.

[LV95] J. Lebbe et R. Vignes. Optimal hierarchical clustering with order constraint. In E. Diday, Y. Lechevallier, et O. Optiz, éditeurs, *Ordinal and Symbolic Data Analysis*, pages 265–276, Paris, June 1995.

[LVHG+90] L. Lemesle-Varloot, B. Henrissat, C. Gaboriaud, V. Bissery, A. Morgat, et J.P. Mornon. Hydrophobic cluster analysis : procedures to derive structural and functional information from 2-D-representation of protein sequences. *Biochimie*, 72 :555–574, 1990.

[LW65] G.N. Lance et W.T. Williams. Computer programs for monothetic classification ("Association analysis"). *The Computer Journal*, 8 :246–249, 1965.

[LW66] G.N. Lance et W.T. Williams. Computer programs for hierarchical polythetic classification ("similarity analyses"). *The Computer Journal*, 9 :60–64, 1966.

[MC85] G.W. Milligan et M.C. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50(2) :159–179, June 1985.

[McI98] J.O. McInerney. Replicational and transcriptional selection on codon usage in *Borrelia burgdorferi*. *Proc. Natl. Acad. Sci. USA*, 95 :10698–10703, September 1998.

[McL77] A.D. McLachlan. Analysis of periodic patterns in amino acid sequences : collagen. *Biopolymers*, 16 :1271–1297, 1977.

[MM89] E.W. Myers et W. Miller. Approximate matching of regular expressions. *Bulletin of Mathematical Biology*, 51(1) :5–37, 1989.

[MS81] R.S. Michalsky et R. Stepp. Revealing conceptual structure in data by inductive inference. In J.E. Hayes-Michie, D. Michie, et Y.H. Pao, éditeurs, *Machine Intelligence*, volume 10, pages 173–196. Chichister : Ellis Horwood, New York : Halsted Press (John Wiley), 1981.

[MSE81] R.S. Michalsky, R. Stepp, et Diday E. A recent advance in data analysis : Clustering objects into classes characterized by conjunctive concepts. *Progress in Pattern Recognition*, pages 33–56, 1981.

[Mur85] F. Murtagh. A survey of algorithms for contiguity-constrained clustering and related problems. *The Computer Journal*, 28(1) :82–88, 1985.

[NG80] P.M. Narendra et M. Goldberg. Image segmentation with directed trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-2(2) :185–191, March 1980.

[NM89] J. Ninio et E. Mizraji. String analysis and energy minimization in the partition of DNA sequences. *Journal of Molecular Biology*, 207 :585–596, 1989.

[NMW97] C.G. Nevill-Manning et I.H. Witten. Identifying hierarchical structure in sequences : a linear-time algorithm. *Journal of Artificial Intelligence Research*, 7 :67–82, September 1997.

[Nus84] R. Nussinov. Doublet frequencies in evolutionary distinct groups. *Nucleic Acids Research*, 12(3) :1749–1763, 1984.

[Oll90] E. Ollivier. *Analyse de séquences biologiques : développement d'une approche indépendante d'un système de référence par traitement du signal.* Thèse, Université Paris 6, 1990.

[Pan83] R.J. Pankhurst. An improved algorithm for finding diagnostic taxonomic descriptions. *Mathematical Biosciences*, 65 :209–218, 1983.

[Pea90] W.R. Pearson. Rapid and sensitive sequence comparison with FASTP and FASTA. *Meth. Enzymol.*, 183 :63–98, 1990.

[Pea91] W.R. Pearson. Searching protein sequence libraries : comparison of the sensitivity and selectivity of the Smith-Waterman and FASTA algorithms. *Genomics*, 11 :635–650, 1991.

[RDV99] E.P.C. Rocha, A. Danchin, et A. Viari. Universal replication biases in bacteria. *Molecular Microbiology*, 32(1) :11–16, 1999.

[Ros78] G.D. Rose. Prediction of chain turns in globular proteins on a hydrophobic basis. *Nature*, 272 :586–590, 1978.

[Rus69] E.H. Ruspini. A new approach to clustering. *Information and Control*, 15 :22–32, 1969.

[RVD98] E.P.C. Rocha, A. Viari, et A. Danchin. Oligonucleotide bias in *Bacillus subtilis* : general trends and taxonomic comparisons. *Nucleic Acids Research*, 26(2) :2971–2980, 1998.

[Sag96] M.F. Sagot. *Ressemblance lexicale et structurale entre macromolécules. Formalisation et approches combinatoires.* Thèse, Université de Marne-la-Vallée, 1996.

[SAL91] G.D. Schuler, S.F. Altschul, et D.J. Lipman. A workbench for multiple alignment construction and analysis. *Proteins*, 9 :180–190, 1991.

[San75] D. Sankoff. Minimal mutation trees of sequences. *SIAM J. Appl. Math.*, 28 :35–42, 1975.

[SCL76] D. Sankoff, R.J. Cedergren, et G. Lapalme. Frequency of insertion-deletion, transversion, and transition in the evolution of 5S ribosomal RNA. *J. Mol. Evol.*, 7 :133–149, 1976.

[SH85] P.H.A. Sneath et R.I.C. Hansell. Naturalness and predictivity of classifications. *Biological Journal of the Linnean Society*, 24 :217–231, 1985.

[Sha48] C.E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27 :379–423, 623–656, July, October 1948.

[Spo91] J.L. Spouge. Fast optimal alignment. *Comput. Appl. Biosci.*, 7 :1–7, 1991.

[SS92] A.M.J. Skulimowski et B.F. Schmid. Redundance-free description of partitioned complex systems. *Mathl. Comput. Modelling*, 16(10) :71–92, 1992.

[SSKT98] S.L. Salzberg, A.J. Salzberg, A.R. Kerlavage, et J.F. Tomb. Skewed oligomers and origins of replication. *Gene*, 217 :57–67, 1998.

[Sta90a] R. Staden. Finding protein coding regions in genomic sequences. *Methods in Enzymology*, 183 :193–211, 1990.

[Sta90b] R. Staden. Searching for patterns in protein and nucleic acid sequences. *Methods in Enzymology*, 183 :193–211, 1990.

[Str97] B. Stroustrup. *The C++ Programming Language. Third edition.* ADDISON-WESLEY, 1997.

[SW81] T.F. Smith et M.S. Waterman. Identification of common molecular subsequences. *J. Mol. Biol.*, 147 :195–197, 1981.

[SWS83] T.F. Smith, M.S. Waterman, et J.R. Sadler. Statistical characterization of nucleic acid sequence functional domains. *Nucleic Acids Research*, 11(7) :2205–2220, 1983.

[Wat83] M.S. Waterman. Sequence alignments in the neighborhood of the optimum with general application to dynamic programming. *Proc. Natl. Acad. Sci. USA*, 80 :3123–3124, 1983.

[WB68] C.S. Wallace et D.M. Boulton. An information measure for classification. *The Computer Journal*, 11 :185–194, 1968.

[WC53] J.D. Watson et F.H.C. Crick. A structure for deoxyribose nucleic acid. *Nature*, 171 :737, 1953.

[WF74] R.A. Wagner et M.J. Fischer. The string-to-string correstion problem. *J. ACM*, 21 :168–178, 1974.

[Wit83] A.P. Witkin. scale-space filtering. In *International Joint Conference on Artificial Intelligence*, Karlsruhe, 1983.

[WLDC71] W.T. Williams, G.N. Lance, M.B. Dale, et H.T. Clifford. Controversy concerning the criteria for taxonometric strategies. *The Computer Journal*, 14(2) :162–165, 1971.

[WP84] M.S. Waterman et M. Perlwitz. Line geometries for sequence comparisons. *Bull. Math. Biol.*, 46 :567–577, 1984.

[Wri90] F. Wright. The 'effective number of codons' used in a gene. *Gene*, 87 :23–29, 1990.

[Zah71] C.T. Zahn. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on computers*, C-20(1) :68–86, January 1971.

[Zuk91] M. Zuker. Suboptimal sequence alignment in molecular biology. Alignement with error analysis. *J. Mol. Biol.*, 221 :403–420, 1991.